CHAPTER 1 GENERAL

Page

1.1 Descriptive Expressions Used in This Manual 1-1
1.2 Sample Program Format 1-4
1.3 How to Read Subroutine Lists 1-6

APPENDIX Sample List 1-7

As various number systems are in use, such as binary, decimal, hexadecimal, etc., the base or radix of a number system is placed as a subscript to the lower right of a number to identify whether the number is decimal, hexadecimal, octal, or binary as shown in Fig. 1-3.

1001_2	(Binary 1001)
10_8	(Octal 10)
10_{10}	(Decimal 10)
10_{16}	(Hexadecimal 10)

Fig. 1-3 Number Notation (with Base m)

Unless otherwise specified, the hexadecimal notation is used throughout this manual to express the contents of memory and registers which are represented by binary numbers. Characters represented by ASCII codes may be enclosed by single quotation marks (ex: 'ABCD'). Symbol Δ represents a space.

The MCU is provided with the internal registers shown in the following Table. The registers are abbreviated as follows.

Table 1-1 Internal Registers

Address	Register	Abbreviation
00	Port 1 Data direction register	DDR1
01	Port 2 Data direction register	DDR2
02	Port 1 Data register	PORT 1
03	Port 2 Data register	PORT 2
04	Port 3 Data direction register	DDR3
05	Port 4 Data direction register	DDR4
06	Port 3 Data register	PORT 3
07	Port 4 Data register	PORT 4
08	Timer control and status register	TCSR
09	Counter (high-order bytes)	FRC
0A	Counter (low-order bytes)	
0B	Output compare register (high-order bytes)	OCR
0C	Output compare register (low-order bytes)	
0D	Input capture register (high-order bytes)	ICR
0E	Input capture register (low-order bytes)	
0F	Port 3 control and status register	
10	Rate and mode control register	RMCR
11	Transmit/receive control and status register	TRCSR
12	Receive data register	RDR
13	Transmit data register	TDR
14	RAM control register	
15 to 1F	Reserved	

CHAPTER 1 GENERAL

1.1 Descriptive Expressions Used in This Manual

The HX-20 incorporates two HD6301 microprocessors. One of the microprocessors has a 64K-byte memory area to control the entire HX-20 components and is called the master MCU (Micro Computer Unit). The other plays an auxiliary role. Namely, it controls I/O devices such as the microprinter, cassettes, etc., and is called the slave MCU. Each MCU has a CPU, a ROM, a RAM, a serial I/O port, a parallel I/O port, and timer function. Fig. 1-1 shows the arrangement of the registers in the CPU.

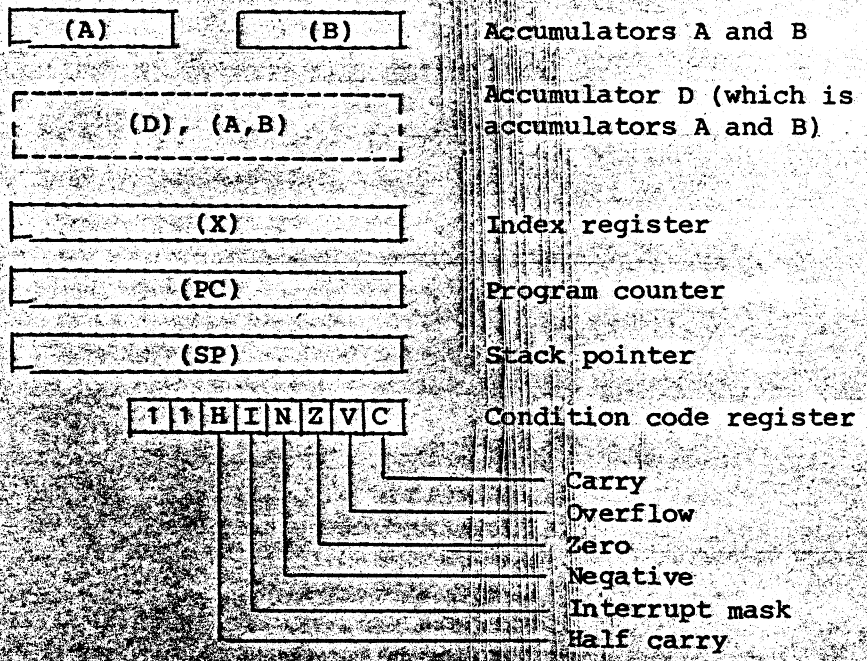


Fig. 1-1 Arrangement of CPU Registers

The registers are identified by symbols: (A) for accumulator A, (B) for accumulator B, (D) or (A,B) for accumulator D, (X) for the index register, (PC) for the program counter, and (SP) for the stack pointer. For the condition code register, (H), (I), (N), (Z), (V), and (C) are used to indicate the respective bits as shown in Fig. 1-1. As shown in Fig. 1-2, bit positions are indicated from the bit lowest place with the lowest place value (or weighting) at the extreme right such as bit 0, bit 1 .... This bit with the lowest place value is called LSB (Least Significant Bit), while the bit with the highest place value (at the extreme left) is called MSB (Most Significant Bit).

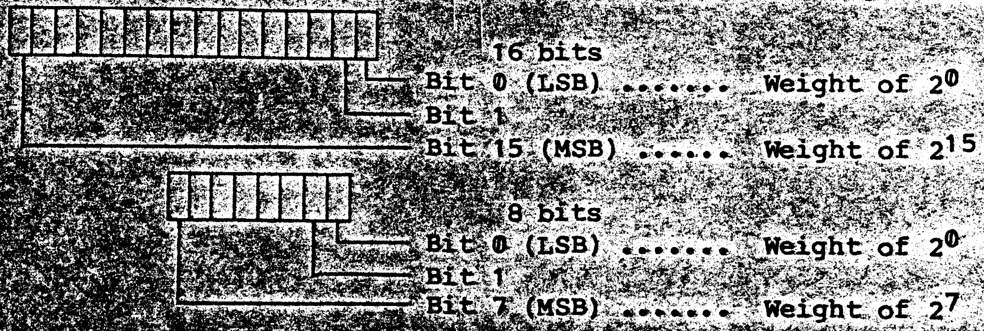

16 bits
Bit 0 (LSB) .... Weight of 2⁰
Bit 1
Bit 15 (MSB) ...... Weight of 2¹⁵
8 bits
Bit 0 (LSB) .... Weight of 2⁰
Bit 1
Bit 7 (MSB) .... Weight of 2⁷

Fig. 1-2 Bit Positions

1.2 Sample Program Format

Table 1-3 shows the standard format of a sample program.

Table 1-3 Standard Format of Sample Program

Column number	Description	
1 to 5	Error message flag	
6 to 10	Source line number (decimal)	
11	Location counter section flag (see Note 1.)	
13 to 16	Location counter value (hexadecimal)	
18 to 19	Operation code (hexadecimal)	
21 to 28 Part which depends on instructions	Non-branch instructions	
	21 to 22	First byte of operand
	23 to 24	Second byte of operand
	26	Section flag of operand (See Note 2.)
	Branch instructions	
	21 to 22	Relative address of branch destination
	24 to 27	Location address of branch destination
	28	Branch error warning flag ("*" is displayed if relative address is in the range $70 to $90.)
30 to 31	M1, M2, M3, M4: Macroexpansion statements (The digit following "M" indicates the nesting level.) IF: Statement skipped by the, IF control instruction	
35 to 40	Label field	
42 to 47	Operation field	
49 to 56	Operand field	
58 to last column	Comment field	

Table 2 lists the abbreviations for the respective bits of each internal register.

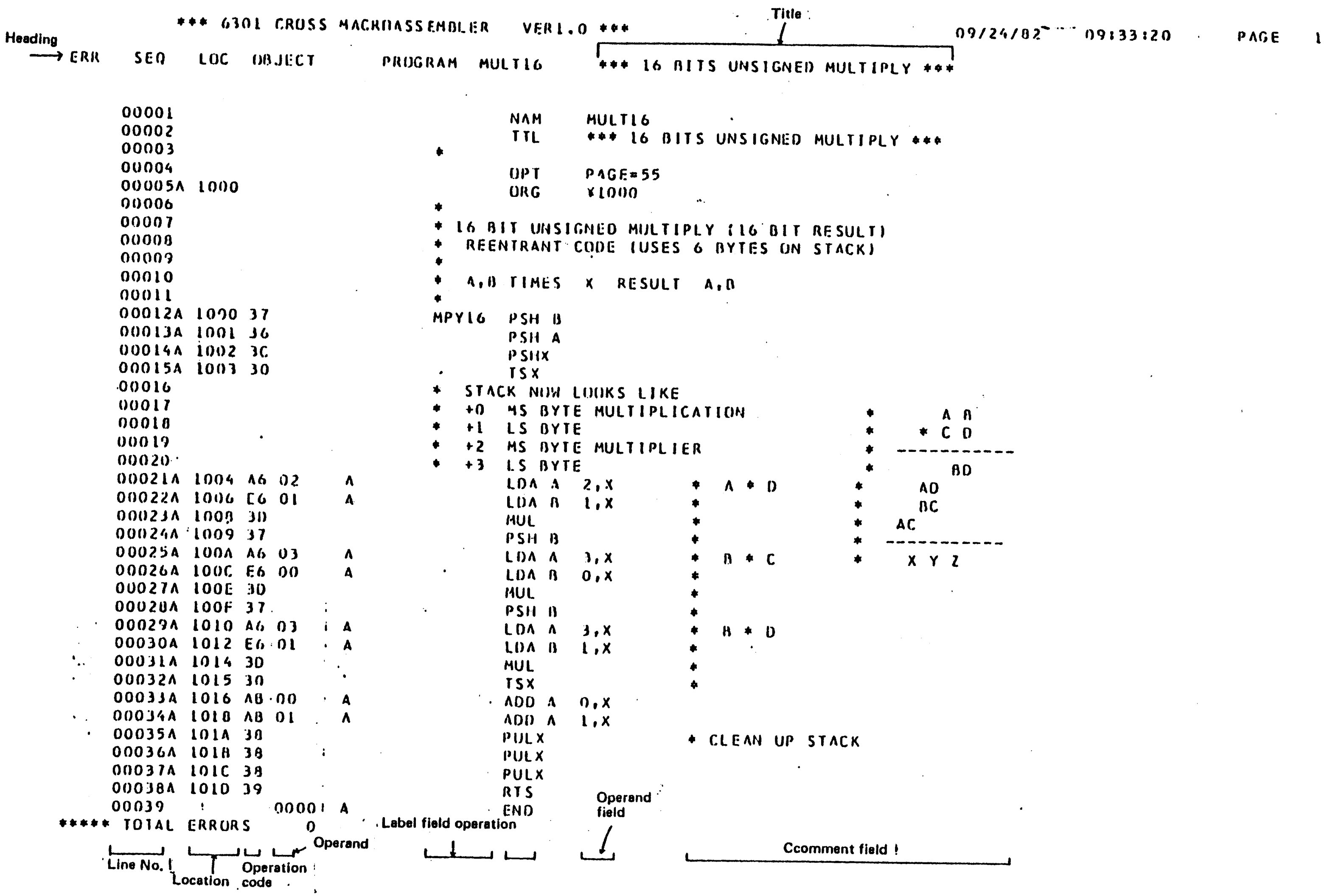
Table 2 Bits of Internal Registers and Their Abbreviations

Address register	Bit	Abbreviation
02 Port 1	0	P10
	1	P11
	2	P12
	3	P13
	4	P14
	5	P15
	6	P16
	7	P17
03 Port 2	0	P20
	1	P21
06 Port 3	0	P30
	1	P31
	2	P32
	3	P33
	4	P34
	5	P35
	6	P36
	7	P37
07 Port 4	0	P40
	1	P41
	2	P42
	3	P43
	4	P44
	5	P45
	6	P46
	7	P47
08 TCSR	0 (Output level)	OLVL
	1 (Input edge)	IEDG
	2 (Enable timer overflow interrupt)	ETOI
	3 (Enable output compare interrupt)	EOCI
	4 (Enable input capture interrupt)	EICI
	5 (Timer overflow flag)	TOF
	6 (Output compare flag)	OCF
	7 (Input capture flag)	ICF
11 TRCSR	0 (Wake up)	WU
	1 (Transmit enable)	TE
	2 (Transmit interrupt enable)	TIE
	3 (Receive enable)	RE
	4 (Receive interrupt enable)	RIE
	5 (Transmit data register empty)	TDRE
	6 (Overrun framing error)	ORFE
	7 (Receive data register full)	

1.3 How to Read Subroutine Lists

The subroutine lists in each chapter contain the subroutine names, entry points, descriptions of subroutines, and parameters. The parameters shown are divided into those to be output for subroutine call and those to be input for subroutine return. In describing the CPU registers, symbols are used: (A) for accumulator A, (B) for accumulator B, and (X) for the index register. For the condition code register, (C) stands for carry, (N) for a negative flag, and (Z) for a zero flag. Details of registers are listed under "Registers retained". "Subroutines referenced" lists the subroutines called in the course of execution. The I/O routines normally use addresses 0050 to 0077 as a work area. The actual locations used are represented as variables. (See Chapter 13.)

"(C): Abnormal I/O flag" appears quite often in the description of parameters at the time of subroutine return. This indicates that the I/O operation has not been performed correctly due to a drop in voltage, the power switch being turned OFF, or the BREAK key being pressed. (C)=1 indicates abnormal I/O operation and (C)=0 indicates normal I/O operation.



```
*** 6301 CROSS MACROASSEMBLER   VER1.0 ***                               09/24/82      09:33:20      PAGE   1
ERR   SEQ    LOC  OBJECT      PROGRAM  MULT16     *** 16 BITS UNSIGNED MULTIPLY ***

      00001                              NAM    MULT16
      00002                              TTL    *** 16 BITS UNSIGNED MULTIPLY ***
      00003                     *
      00004                              OPT    PAGE=55
      00005A 1000                        ORG    $1000
      00006                     *
      00007                     * 16 BIT UNSIGNED MULTIPLY (16 BIT RESULT)
      00008                     *  REENTRANT CODE (USES 6 BYTES ON STACK)
      00009                     *
      00010                     *  A,B TIMES  X  RESULT  A,B
      00011                     *
      00012A 1000 37            MPY16    PSH B
      00013A 1001 36                     PSH A
      00014A 1002 3C                     PSHX
      00015A 1003 30                     TSX
      00016                     *  STACK NOW LOOKS LIKE
      00017                     *  +0  MS BYTE MULTIPLICATION                    *      A B
      00018                     *  +1  LS BYTE                                   *    * C D
      00019                     *  +2  MS BYTE MULTIPLIER                        *  -----------
      00020                     *  +3  LS BYTE                                   *       BD
      00021A 1004 A6 02     A            LDA A  2,X           *  A * D           *     AD
      00022A 1006 E6 01     A            LDA B  1,X           *                  *     BC
      00023A 1008 3D                     MUL                  *                  *   AC
      00024A 1009 37                     PSH B                *                  *  -----------
      00025A 100A A6 03     A            LDA A  3,X           *  B * C           *     X Y Z
      00026A 100C E6 00     A            LDA B  0,X           *
      00027A 100E 3D                     MUL                  *
      00028A 100F 37                     PSH B                *
      00029A 1010 A6 03     A            LDA A  3,X           *  B * D
      00030A 1012 E6 01     A            LDA B  1,X           *
      00031A 1014 3D                     MUL                  *
      00032A 1015 30                     TSX                  *
      00033A 1016 AB 00     A            ADD A  0,X
      00034A 1018 AB 01     A            ADD A  1,X
      00035A 101A 38                     PULX                 * CLEAN UP STACK
      00036A 101B 38                     PULX
      00037A 101C 38                     PULX
      00038A 101D 39                     RTS
      00039            00001 A           END
***** TOTAL ERRORS      0
```

Fig. 4 Example of Program List Format

```
          *** 6301 CROSS MACROASSEMBLER   VER1.0 ***                                          10/21/82    09:41:24
ERR   SEQ    LOC  OBJECT        PROGRAM  MULT16      --- 16 BITS UNSIGNED MULTIPLY ---

      00001                                       NAM     MULT16
      00002                                       TTL     --- 16 BITS UNSIGNED MULTIPLY ---
      00003                                *
      00004                                * FILE NAME  'EXS3'  BY K.A
      00005                                       OPT     PAGE=55
      00006                                       OPT     LOAD
      00007                                *
      00008A 1000                                 ORG     $1000
      00009                                *
      00010                                * 16 BIT UNSIGNED MULTIPLY (16 BIT RESULT)
      00011                                *  REENTRANT CODE (USES 6 BYTES ON STACK)
      00012                                *
      00013                                *  A,B TIMES  X  RESULT  A,B
      00014                                *
      00015A 1000 37                       MPY16  PSH B
      00016A 1001 36                              PSH A
      00017A 1002 3C                              PSHX
      00018A 1003 30                              TSX
      00019                                *  STACK NOW LOOKS LIKE
      00020                                *  +0  MS BYTE MULTIPLICATION              *      A B
      00021                                *  +1  LS BYTE                             *    * C D
      00022                                *  +2  MS BYTE MULTIPLIER                  *  -----------
      00023                                *  +3  LS BYTE                             *       BD
      00024A 1004 A6 02      A                    LDA A  2,X       *  A * D          *     AD
      00025A 1006 E6 01      A                    LDA B  1,X       *                 *     BC
      00026A 1008 3D                              MUL              *                 *   AC
      00027A 1009 37                              PSH B            *                 *  -----------
      00028A 100A A6 03      A                    LDA A  3,X       *  B * C          *    X Y Z
      00029A 100C E6 00      A                    LDA B  0,X       *
      00030A 100E 3D                              MUL              *
      00031A 100F 37                              PSH B            *
      00032A 1010 A6 03      A                    LDA A  3,X       *  B * D
      00033A 1012 E6 01      A                    LDA B  1,X       *
      00034A 1014 3D                              MUL              *
      00035A 1015 30                              TSX              *
      00036A 1016 AB 00      A                    ADD A  0,X
      00037A 1018 AB 01      A                    ADD A  1,X
      00038A 101A 38                              PULX             * CLEAN UP STACK
      00039A 101B 38                              PULX
      00040A 101C 38                              PULX
      00041A 101D 39                              RTS
      00042             0000  A                   END
***** TOTAL ERRORS      0
```

CHAPTER 2 INPUT FROM KEYBOARD

Page

2.1 General 2-1
2.2 I/O Ports for Keyboard Input 2-1
2.3 Key Scan 2-2
2.4 Keyboard input interrupt 2-3
2.5 Timing of Key Input Process.................................. 2-3
2.6 Automatic Key Input at Power ON 2-4
2.7 Key Input Subroutines 2-4
2.8 Sleep Function 2-4
2.9 Special Keys.. 2-4
2.10 Key Input Modes 2-5
2.11 Changing Constants 2-5
2.12 Tables 2-6
2.13 Keyboard Input Subroutines 2-8
2.14 Keyboard Work Area 2-10

2.1 General

The keyboard, connected to the master MCU, has 8 lines each of which fetches 1Ø data. In other words, the keyboard is an 8 x 1Ø matrix structure. The pressed key can be found by inputting the data for each line. Interrupt IRQ1 occurs each time a key is pressed. The keyboard matrix incorporates the Printer ON/OFF and DIP switches in addition to the alphanumeric keys.

Key input is processed by interrupts and input data is stored in the 8-byte key stack. A power ON key stack, which stores data to be input automatically from the keyboard when the power is turned ON, is also provided. The contents of the power ON key stack are first fetched and the data in the key stack is input when the power ON key stack becomes empty. The contents of the power ON key stack can be restored by turning the power ON (reset).

2.2 I/O Ports for Keyboard Input

Table 2.1 shows the I/O ports related to the keyboard input.

Table 2.1 I/O Ports Related to Keyboard Input

Address	Bit position	Definition
2Ø	Ø	Output Specifies scanning of LØ line. Ø: Scanning enabled. 1: Scanning is not performed.
	1	Output L1
	2	Output L2
	3	Output L3
	4	Output L4
	5	Output L5
	6	Output L6
	7	Output L7
22	Ø	Input Scan result DØ Ø: ON 1: OFF
	1	Input Scan result D1 Ø: ON 1: OFF
	2	Input Scan result D2 Ø: ON 1: OFF
	3	Input Scan result D3 Ø: ON 1: OFF
	4	Input Scan result D4 Ø: ON 1: OFF
	5	Input Scan result D5 Ø: ON 1: OFF
	6	Input Scan result D6 Ø: ON 1: OFF
	7	Input Scan result D7 Ø: ON 1: OFF
26	4	Output Key input interrupt mask Ø: Mask 1: Mask open
28	Ø	Input Scan result D8
	1	Input Scan result D9
P15		Input Key input interrupt flag Ø: A keyboard input interrupt has occurred. 1: No keyboard input interrupt has occurred.

2.3 Key Scan

As shown in Fig. 2-1, ten data can be input from each of the eight lines connected to the keyboard. Line L0 inputs data from keys 0, 1, 2, 3, 4, 5, 6, 7, the PF1 key and DIP switch 1. In the same way, data from keys @, A, B, C, D, E, F, G, the PF3 key and DIP switch 3 are input through line L2. This means that to input all of the data from the keyboard, lines L0 to L7 must be selected in turn and the data fetch operation repeated eight times.

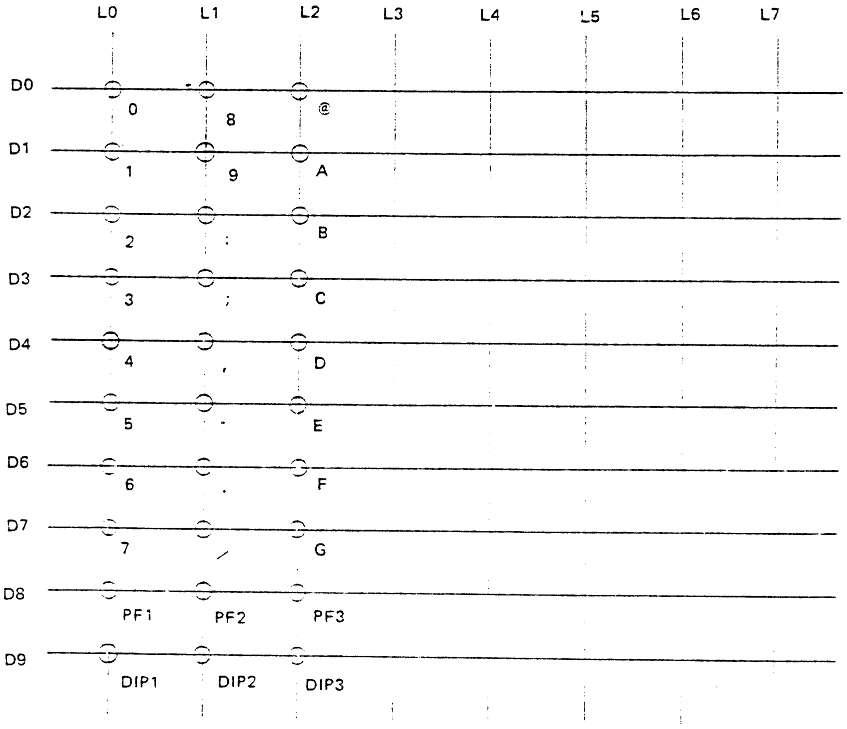


Fig. 2-1 Key Matrix

In some cases, data may not be input correctly from the keyboard due to this circuit configuration. For example, when keys 1, 8 and 9 are pressed, the circuit recognizes key 0 as having been pressed. That is, if key 9 is pressed while holding down keys 1 and 8, the input is recognized as 1, 8, 0. There are several such combinations which will cause incorrect data to be received. Key scan is performed by the following procedure.

(1) Close key input interrupt mask
P264 (bit 4 at address 26) is an IRQ1 key input interrupt mask. As an interrupt occurs if the key is pressed (i.e., the line to scan is specified and the key on the line is pressed) if this mask is open, the interrupt is disabled.

(2) Specify line to scan
There are 8 lines, L0 to L7, and any line can be specified for input. When line L0 is specified, the data on the line L0 can be input. If all lines L0 to L7 are specified, any key can be detected. The bit 0 at address 20 specifies line L0. When the

value of this bit is '0', scan is enabled and when '1', scan is not performed. The value FE (line L0 is scanned and the other lines are not scanned) is output first to address 20.

(3) Fetch data

When the contents of the address 22 are input, the data in D0 through D7 can be obtained. When the contents of address 28 are input, the data in D8 and D9 (bit 0, bit 1) can be obtained. (There is a wait of several tens of microseconds to obtain correct data after the line is specified.) Now, input from keys 0 to 7, PF1 and DIP switch 1 is enabled.

(4) Scan lines

Lines L1 through L7 are sequentially scanned and procedures 2 and 3 above are repeated. In this way all the data from the keyboard as well as the DIP switch values can be input. Table 2.2 shows the arrangement of the key matrix. Table 2.3 shows the arrangement of the keyboard matrix.

2.4 Keyboard input interrupt

An IRQ1 interrupt is enabled when the keyboard data is input. The following procedure is used to issue an IRQ1 interrupt.

(1) Specify the key line

The line where an interrupt occurs when a key is pressed is specified. Set '0' in address 20 to specify the key scan line. Once '0' is set, an interrupt occurs when any key is pressed. Note that the keys and switches on D9 such as DIP switches 1 to 4, shift key, control keys, and Printer ON/OFF switch are excluded from the keyboard input interrupt.

(2) Open the keyboard input interrupt mask

Write '1' to bit 5 of address 26 (P265) where the keyboard input interrupt mask is performed.

(3) Open the CPU interrupt mask

The CPU interrupt mask is opened by a CLI command.

(4) Confirm the keyboard interrupt

If the P15 is '0' when an IRQ1 interrupt occurs, it indicates the occurrence of the key input interrupt.

2.5 Timing of Key Input Process

An IRQ1 interrupt occurs when a key is input. Sampling (OCR interrupt) is performed using the MCU free running counter.

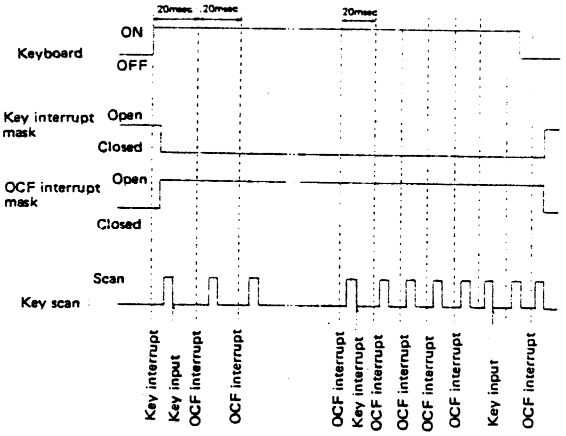


Fig. 2-2 Timing of Key Input

After a key is pressed as shown in Fig. 2-2, the Output Compare Register (OCR) issues interrupts at intervals of 20 msec (the key interrupt mask is closed) and auto repeat process is performed by key scan. If the same key is pressed continuously for a certain number of key scans after issuance of an OCF interrupt, it is assumed that the key has been newly pressed. The received data from the keyboard is stored in the First In, First Out (FIFO) key stack.

2.6 Automatic Key Input at Power ON

The 18-byte variable KYISTK contains key input data that can be specified by a monitor K command during reset (refer to Memory Map). When the value of the variable KYISFL is 0A, the KYISTK contains key input data. When the value is 0B, key input data is currently being fetched from the KYISTK. If the value of the KYISFL is 0B when the subroutine KEYIN (to fetch the key input data from the key stack) is called, the value obtained from the KYISTK is used as the key input data.

2.7 Key Input Subroutines

The following subroutines for key input are provided.
(1) INITKY: Initializes the keyboard and sets the default value.
(2) KEYSCN: Performs the key scan operation and obtains input data from the pressed key.
(3) KEYIN: Fetches one character from the key input buffer.
(4) KEYSTS: Obtains the number of characters in the key input buffer.
(5) KYSSTK: Specifies the automatic input key data.

2.8 Sleep Function

The MCU is provided with a sleep function to reduce power consumption when it is not functioning. The sleep mode is entered during execution of the KEYIN subroutine to wait for key input when the key input buffer is empty.

2.9 Special Keys

(1) BREAK key
When the BREAK key is pressed, the data is not taken into the key stack, but the I/O operation is cancelled. (Subroutine BREKIO is called.) Then, the break process is performed to the slave MCU and bit 7 of variables MIOSTS (address 007D) and SIOSTS (address 007C) become ON. The data input from the power ON key stack is cancelled. If bit 7 of the variable RUNMOD (address 007B) is '1', control returns from the key input interrupt. When bit 7 is '0', the subroutine is called starting at the address specified by address (0120, 0121). The default values of address (0120, 0121) is (FF, B2). The subroutine RSTRIO (re-start of I/O operation) is executed at the entry point of the address FFB2 and control jumps to the menu routine. In addition to the BREAK key, the specified subroutines are executed when the MENU, PAUSE, CTRL/PF3, CTRL/PF4 and CTRL/PF5 keys are pressed. Any addresses can be specified.

(2) MENU key
When the MENU key is pressed and bit 7 of the variable RUNMOD is '1', the code FC is input to the key stack and control returns from the interrupt. When bit 7 is '0', control returns from the interrupt after executing the subroutine starting at the address specified by address (0122, 0123).

(3) PAUSE key
When the PAUSE key is pressed, bit 6 of the variable MIOSTS becomes '1'. Then, control returns from the interrupt if bit 7 of the variable RUNMOD is '1'. If bit 7 is '0', control returns from the interrupt after executing the subroutine starting at the address specified by address (0124, 0125).

(4) CTRL/PF3, CTRL/PF4 and CTRL/PF5 keys
When the CTRL/PF3, CTRL/PF4 or CTRL/PF5 keys are pressed, control returns from the interrupt after executing the corresponding subroutine starting at the address specified by the address (0126, 0127), (0128, 0129) and (012A, 012B). If, for example, (FF, 10) (the entry point of Monitor) is written to the address (0126, 0127), the Monitor will be executed when the CTRL/PF3 keys are pressed.

2.10 Key Input Modes

The current key input mode (numeric and uppercase, shift, etc.) is indicated by the 1-byte variable KEYMOD. The address of this variable is (FFE4, FFE5). The current data in this address is 0169. Referring to the contents in the address, the current mode, (in this case, the keyboard mode) can be recognized. To force-set a certain mode, change the contents of the current address to those of the mode to be set. The following three modes are available.

Bit 1: Numeric mode
Bit 2: CAPS mode (lowercase letter mode is assumed when bit
2 is '1'.)
Bit 4: Graphic mode

Only one of these bits may be '1' or all of them may be '0'. The current mode is indicated by the bit which is '1'.

2.11 Changing Constants

The constants on the RAM are the following.
Key stack size, time interval until the second key input is accepted for auto repeat, time interval until the third or subsequent key input is accepted for auto repeat, sampling interval of key scan and power ON key stack. The default values for these constants are set when the keyboard is initialized. Values set after the default values have been set are used. (For details, refer to Memory Map).

Key scan

The keyboard value read by key scan is assigned to variable NEWKTB (10 bytes, starting address: FFD0, FFD1). Fig. 2-3 shows the format of the keyboard values read by key scan.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
NEWKTB-0	D7(L0)	D6(L0)	D5(L0)	D4(L0)	D3(L0)	D2(L0)	D1(L0)	D0(L0)
NEWKTB-1	D7(L1)	D6(L1)	…………………………					D0(L1)
NEWKTB-2	D7(L2)	D6(L2)	…………………………					D0(L2)
NEWKTB-3	D7(L3)	D6(L3)	…………………………					D0(L3)
NEWKTB-4	D7(L4)	D6(L4)	…………………………					D0(L4)
NEWKTB-5	D7(L5)	D6(L5)	…………………………					D0(L5)
NEWKTB-6	D7(L6)	D6(L6)	…………………………					D0(L6)
NEWKTB-7	D7(L7)	D6(L7)	…………………………					D0(L7)
NEWKTB-8	D8(L7)	D8(L6)	D8(L5)	D8(L4)	D8(L3)	D8(L2)	D8(L1)	D8(L0)
NEWKTB-9	D9(L7)	D9(L6)	D9(L5)	D9(L4)	D9(L3)	D9(L2)	D9(L1)	D9(L0)

Fig. 2-3 Key Scan Values

In this case, the DIP switches and the PRINTER ON/0FF switch are set according to the values at address 7F. (In other words, software specification takes precedence over the key scan specification.)

2.12 Tables

Table 2.2 Key Matrix

Line / Return data	L 0	L 1	L 2	L 3	L 4	L 5	L 6	L 7
D 0	0 0 0	0 8 8	1 0 @	1 8 H	2 0 P	2 8 X	3 0 RET	3 8 CLR
D 1	0 1 1	0 9 9	1 1 A	1 9 I	2 1 Q	2 9 Y	3 1 SPACE	3 9 SCRN
D 2	0 2 2	0 A :	1 2 B	1 A J	2 2 R	2 A Z	3 2 TAB	3 A BREAK
D 3	0 3 3	0 B ;	1 3 C	1 B K	2 3 S	2 B [		3 B PAUSE
D 4	0 4 4	0 C ,	1 4 D	1 C L	2 4 T	2 C]		3 C DEL
D 5	0 5 5	0 D –	1 5 E	1 D M	2 5 U	2 D ¥	3 5 NUM	3 D MENU
D 6	0 6 6	0 E .	1 6 F	1 E N	2 6 V	2 E →		
D 7	0 7 7	0 F	1 7 G	1 F O	2 7 W	2 F ←	3 7 CAPS	
D 8	4 0 PF1	4 1 PF2	4 2 PF3	4 3 PF4	4 4 PF5	4 5 PAPER		
D 9	4 8 DIP1	4 9 DIP2	4 A DIP3	4 B DIP4		4 D SHIFT	4 E CTRL	4 F PRINT

Note: The hexadecimal values in the above Table indicate the positions of the key layout on the keyboard matrix table.

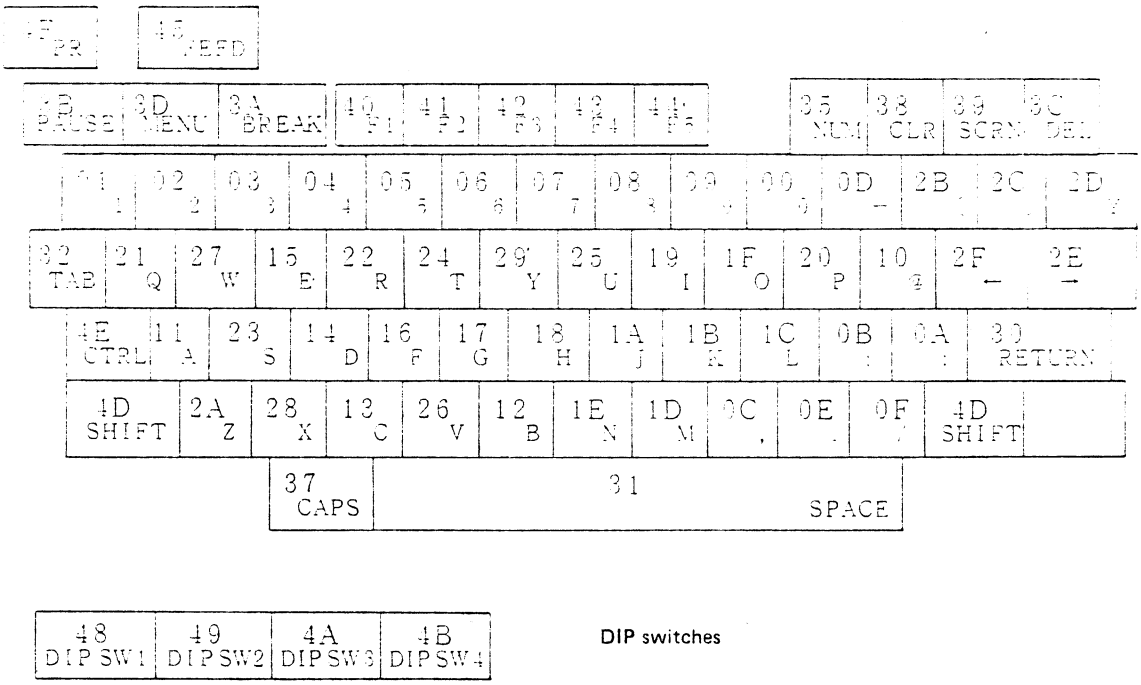


Fig. 2-4 Arrangement of the Keyboard Matrix

Note: The hexadecimal values in the above Table indicate the positions of the keys on the key table.

2.13 Keyboard Input Subroutines

Name of Subroutine	Entry Point	Description
INITKY	FFA0	Initializes key input. Sets the initial values (including default values). Specifies the vectors jumped to when the BREAK, MENU, PAUSE, CTRL/PF3, CTRL/PF4, and CTRL/PF5 keys are pressed. Specifies default values for timing such as sampling time, etc. Specifies the number of key stack data, and the key stack used when the power is turned ON.
		°Parameters: At Entry None At Return None °Registers retained None °Subroutines referenced None °Variables used None
KEYSTS	FF9D	Inputs the number of key stack data (Excluding the key stack used when the power is turned ON.)
		°Parameters: At Entry None At Return (C): Abnormal I/O flag (A): Number of key stack data (in bytes) (Z): According to the value of (A) °Registers retained (B) and (X) °Subroutines referenced None °Variables used None Note: When a PF key is pressed, the number of stack data increases by 2.
KEYIN	FF9A	Inputs one character from the key stack. If no data exists in the key stack, this subroutine lets the MCU sleep and waits until data is input from the keyboard. If data exists in the key stack when the power is turned ON, this data is recognized as input data. If both key stack data and keyboard data are available, the key stack data take precedence over keyboard data.
		°Parameters: At Entry None At Return (C): Abnormal I/O flag (A,B): Character code. 1-byte codes are stored in (A) and 2-byte codes ((A)=FE) are stored in (A,B).

Name of Subroutine	Entry Point	Description
		°Registers retained (X) °Subroutines referenced None °Variables used None
KEYSCN	FF6A	Scans the key matrix. The result of the key scan is stored in NEWKTB (1Ø bytes). Note that the DIP switches and Printer ON/OFF switch are set according to the value of the variable SDIPS2. The contents of NEWKTB are: Bit 7 Bit 6...Bit Ø NEWKTB+Ø: D7 D6 .... DØ...... LØ NEWKTB+1: D7 D6 .... DØ...... L1 . . . . NEWKTB+8: L7 L6 .... L1...... D8 NEWKTB+9: L7 L6 .... L1...... D9
		°Parameters: At Entry None At Return None °Registers retained None °Subroutines referenced None °Variables used KØ and K1 (The values of these variables are retained.)
KYSSTK	FF22	Inputs data to the key stack when the power is turned ON. The size of the key stack is 18 bytes. If more than 18 bytes of data are input, the excess data is ignored.
		°Parameters: At Entry (X): Starting address of character strings (B): Number of characters (Ø to 18: the key stack is cleared when the number is Ø.) °Parameters: At Return None °Registers retained None °Subroutines referenced None °Variables used None

2.14 Keyboard Work Area

Address (from)	(to)	Variable name	Bytes	Description
0140	0140	KSTKSZ	1	The size of the key stack. The default value is 8. May be specified in the range 1 to 15. If the value is '1', input of PF keys is not accepted.
0141	0141	KICNT1	1	Time until the first key input is accepted for auto repeat. The unit depends on sampling time. The default value for sampling time is 20 msec. The default value is 40 (800 msec).
0142	0142	KICNT2	1	Time until the second or subsequent key input is accepted for auto repeat. The unit are the same as those of KICNT1. The default value is 6 (120 msec).
0143	0144	KICNTM	2	Sampling time. The unit 1 equals approx. 1.6 µsec. The default value is 12,288 (20 msec).
0145	014E	NEWKTB	10	Value of the key scan matrix. The status after the key scan is stored in this area. '1' denotes the ON condition. Bit 0 at the first address of the work area corresponds to 00 of the key matrix table and bit 7 corresponds to 07. In this manner, bit 7 of the last address corresponds to 4F.
014F	0158	OLDKTB	10	The value of the previous key scan. The previous value of NEWKTB is stored in this variable.
0159	0162	CHKKTB	10	Stores the data for the position of the newly pressed key after key scan.
0163	0164	KYISAD	2	Address of automatic key input when power is turned ON. Set to 016F at reset.
0165	0165	KYISFL	1	Flag indicating whether or not data exists in the key stack when power is turned ON. When this flag is 0A, data exists in the key stack but the fetch operation ends. When the flag is 0B, data is currently being fetched from the stack. If the flag is other than 0A and 0B, no data exists in the key stack.
0166	0166	KYISCN	1	The number of data in the automatic input key stack. Value is in the range 0 to 255.
0167	0167	KYISPN	1	The number of data input from the automatic key input. The number is in the range 0 to the value specified by KYISCN.
0168	0168	STKCNT	1	The number of data in the input key stack. The number is in the range 0 to the value specified by KSTKSZ.
0169	0169	KEYMOD	1	Input key modes. This address indicates the uppercase, numeric modes, etc.

Address (from)(to)	Variable name	Bytes	Description
			Bit 1: Numeric mode when this bit is '1'. Bit 2: Lowercase mode when this bit is '1'. Bit 3: Unused Bit 4: Graphic mode when this bit is '1'. Bit 5: SHIFT mode when this bit is '1'. Bit 6: The CTRL key has been pressed when this bit is '1'. Bit 7: Indicates that a special keys such as the BREAK, PAUSE, MENU or one of the PF keys has been pressed when this bit is '1'. One of bits 0 through 4 must be ON or all bits must be OFF.
016A 016A	ONKFLG	1	Indicates the key input status. 00: Inhibits key reception. Waits until the pressed key is released. FF: Key input enabled. 01: Auto repeat function
016B 016B	KPRFLG	1	For auto repeat, this variable indicates the number of times the same key input has been received. When this value equals that of KICNT1 or KICNT2, the pressed character is taken to be input once.
016C 016C	KEYRPT	1	Indicates the auto repeat key position on the matrix. Refer to the Matrix Table.
016D 016E	CKEYRD	2	Input key code A PF key is 2 bytes.
016F 0180	KYISTK	18	Work area for the power ON key stack
0181 0188	CHRSTK	8	Work area for the key stack
0189 018F		7	This area is secured for expansion of the key stack.

CHAPTER 3 LIQUID CRYSTAL DISPLAY (LCD)

Page

3.1 General 3-1
3.2 Functions of LCD Controllers 3-1
3.3 I/O Ports for Display and Input 3-2
3.4 Data Display Procedure 3-2
3.5 Input of Display Data 3-3
3.6 Display Subroutines 3-3
3.7 Coordinates on the LCD 3-3
3.8 LCD Subroutines 3-4
3.9 Screen Routine Work Area 3-6

APPENDIX Sample Lists 3-8

3.1 General

The Liquid Crystal Display (LCD) has a resolution of 120 horizontal dots and 32 vertical dots and LCD controllers which enable the specification of data for each dot.

6 LCD controllers together control the LCD, each of which controls an area 40 horizontal by 16 vertical dots.

Normally, a single character is displayed in a matrix of 6 horizontal by 8 vertical dots. Alphanumeric characters, however, are actually formed in a matrix 5 by 7 dots as spaces are left between characters on the screen.

3.2 Functions of LCD Controllers

As abovementioned, the 6 controllers together control the LCD. The dot areas controlled by each controller are shown in Table 3-1.

Table 3-1 Dot Area Controlled by Each LCD Controller

	0 … 39	40 … 79	80 … 119
0 7	Controller 1 (bank 0)	Controller 2 (bank 0)	Controller 3 (bank 0)
8 15	Controller 1 (bank 1)	Controller 2 (bank 1)	Controller 3 (bank 1)
16 23	Controller 4 (bank 1)	Controller 5 (bank 0)	Controller 6 (bank 0)
24 31	Controller 4 (bank 1)	Controller 5 (bank 1)	Controller 6 (bank 1)

As shown in the table, each controller is responsible for an area of above of 40 by 16 dots.

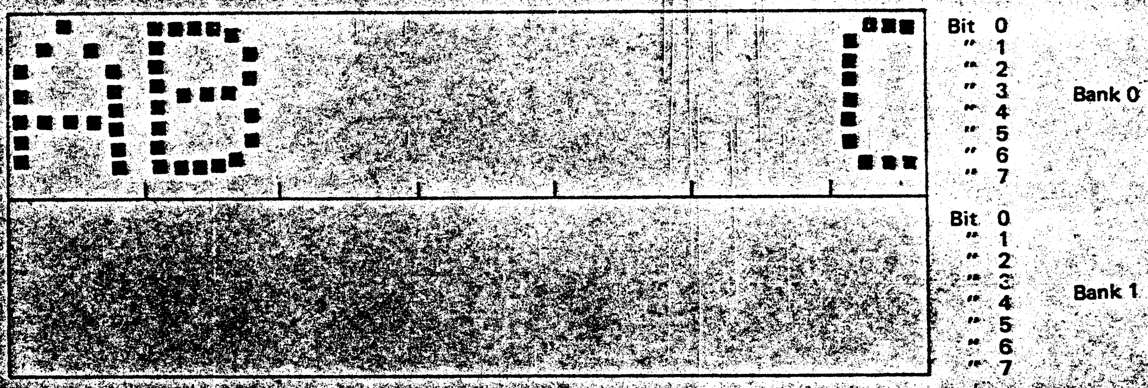


Fig. 3-1 Displayed Contents of Each LCD Controller

Each controller has data addresses 0 to 27_{16} in the row direction. Data consists of 8 bits (Fig. 3-1).

3.3 I/O Ports for Display and Input

Table 3-2 I/O Ports Related to LCD Controllers

	Address	Bit position	Description
Master MCU	26	0 1 2	Output Selection of LCD driver 1-6: Controllers 1 to 6 selected. 0: No controller is selected.
		3	Output Selection of data or command for LCD driver 0: Data 1: Command
	28	7	Input BUSY signal of LCD controller 0: Busy
	2A		Output Serial clock to LCD controller
	2B		Output Serial clock to LCD controller
	2A	0	Output Output data to LCD controller
		1	Output Output data to LCD controller
		2	Output Output data to LCD controller
		3	Output Output data to LCD controller
		4	Output Output data to LCD controller
		5	Output Output data to LCD controller
		6	Output Output data to LCD controller
		7	Output Output data to LCD controller

3.4 Data Display Procedure

Data is displayed on the LCD by the following procedure.

(1) Selection of controller

One of the 6 controllers is selected by specifying an appropriate value in the bits 0 to 2 of address 26 using subroutine WRTP26. If 0 is specified, no controller is selected. System default is 0 for power conservation.

(2) Command setting

The bit 3 of address 26 is a bit used to select either data or command for the selected controller. When this bit is set to "1", a command is selected. This data/command selection may be performed simultaneously with the controller selection described in (1) above. Set a command in address 2A and confirm that the LCD controller is ready (when the bit 7 of address 28 is "1"). Then apply 8 serial clock pulses to the controller. (Address 2A is read 8 times. Since address 2B is also for serial clock input, 8 serial clock pulses are given to the controller upon execution of "LDD$2A" 4 times.)

(3) Data

When the bit 3 of address 26 is set to "0", data is selected. The data setting procedure is the same as the command setting described above. Depending on the type of command, data must be continuously output for display.

3.5 Input of Display Data

The bit indicating that the controller is busy (i.e., bit 7 of address 28) becomes the input data for display.

3.6 Display Subroutines

The HX-20 has the following three subroutines for character display:

(1) DSPLCN: Displays n characters of data (ASCII code) on the physical screen.
(2) DSPLCH: Displays one character of data (ASCII code) on the physical screen.
(3) DISPIT: Displays one character of data (ASCII code) on the physical screen.

3.7 Coordinates on the LCD

(x,y) indicates the coordinates on the LCD. x is the coordinate in the horizontal directions (columns) and y is the coordinate in the vertical direction (rows). (0,0) indicates that the positions of both the vertical and horizontal coordinates are the upper left edge of the LCD. The values of x,y coordinates on the text screen must be in the range shown below.

$0 \leq x \leq 19$ and $0 \leq y \leq 3$

The values of x,y coordinates on the graphic screen must be in the range shown below.

$0 \leq x \leq 119$ and $0 \leq y \leq 31$

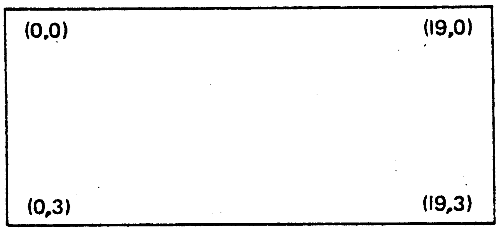
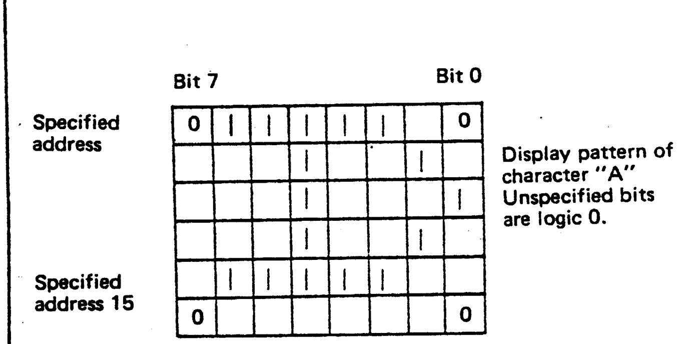


Fig. 3-2 Coordinates on the LCD (Text Screen)

3.8 LCD Subroutines

Subroutine name	Entry point	Description
DSPLCN	FF49	Displays or clears n characters on the physical screen.
		Parameters: At Entry (B): Number of characters displayed The screen is cleared when (B) is 0. (x): Starting address of data packet This parameter need not be specified when (B) is 0. Data packet Byte 0: x coordinate (0 to 19_{10}) of the display position of the first character. Byte 1: y coordinate (0 to 3_{10}) of the display position of the first character. Byte 2: Character code (ASCII) Byte 3: Character code (ASCII) Byte n+1: Character code (ASCII) At Return None Registers retained None Subroutines referenced 'DSPLCH' Variables used None
DSPLCH	FF4C	Displays one character on the physical screen. The display data is first written into the screen buffer PSBUF.
		Parameters: At Entry (A): ASCII character code (X): Display position on LCD (high, low)=x coordinate (0 to 19), y coordinate (0 to 3) At Return None Registers retained (A), (B), and (X) Subroutines referenced CHRGEN, LCDMOD and DATMOD Variables used None
DISPIT	FF5B	Displays one character on the physical screen. The display data is not written into the screen buffer (PSBUF).
		Parameters at entry and return, registers retained, subroutines referenced and variables used are the same as those for subroutine DSPLCH.

Subroutine name	Entry point	Description
CHRGEN	FF67	Generates the character pattern. A character pattern (6x8 dots) is provided for display of the character specified by the ASCII code on the LCD. Certain character patterns may change according to the value set by the DIP switch for different countries.
		Parameters: At entry (A): Character code (X): Starting address where 6-byte character display pattern is stored At Return Character display pattern (specified address)  Bit 7 Bit 0 Specified address Specified address 15 Display pattern of character "A" Unspecified bits are logic 0. Registers retained None Subroutines referenced None Variables used None Others Re-entrant

3.9 Screen Routine Work Area

Address	Variable name	No. of bytes	Description
220 26F	PSBUF	80	Positions of data (ASCII codes) displayed on the physical screen represented in (column, row) format as follows: (0,0), (1,0) ... (19_{10}, 0) (0,1) ... (19_{10}, 3)
270 271	SCRTOP	2	Starting address of the virtual screen buffer
272 273	SCRBOT	2	Ending address of the virtual screen buffer
274 275	DISTOP	2	Starting position of the physical screen on the virtual screen The address of position (0,0) of the physical screen in the physical screen buffer
276 276	VSCRX	1	Virtual screen width indicated as the maximum values of x coordinate.
277 277	VSCRY	1	Virtual screen length indicated as the maximum value of y coordinate.
278 278	CURX	1	x coordinate of the cursor position (on the physical screen)
279 279	CURY	1	y coordinate of the cursor position (on the physical screen)
27A 27A	LRMODE	1	Scroll step x (left and right)
27B 27B	UDMOD	1	Scroll step y (up and down)
27C 27C	CURMRG	1	Scroll margin (1 through 10)
27D 27D	SSPEED	1	Vertical scrolling speed (0 to 9) When the scrolling speed is set at 8, there is a 130-msec wait between each scroll. This wait is increased in 130-msec increments for each setting: 7,6,5 ... 0. When a scrolling speed of 9 is specified, there is no wait between vertical scrolls.
27E 27E	DISPX	1	x coordinate (0 to 19_{10}) of the character to be displayed on the physical screen by a virtual screen routine
27F 27F	DISPY	1	y coordinate (0 to 3) of the character to be displayed on the physical screen by a virtual screen routine
280 280	DISSTS	1	Indicates the display status. Bit 0: Indicates whether or not left and right scrolling is permitted. 1: Scrolling disabled 0: Scrolling enabled Bit 1: Not used Bit 2: Not used Bit 3: Not used Bit 4: Indicates whether or not there is a wait in vertical scrolling. 1: Wait executed

Address	Variable name	No. of bytes	Description
			Bit 5: Cursor ON/OFF switch Determines whether the cursor ('_' below the character) will be displayed on the physical screen. 1: Cursor ON 0: Cursor OFF Bit 6: Indicates cursor ON/OFF status. 1: Cursor ON 0: Cursor OFF Bit 7: Flag to indicate whether or not the entire screen is to be rewritten 0: Display for only one character 1: Rewrites for entire screen. Note: All screen data must be checked and rewritten even if only one character is to be displayed. However, since doing this adversely affects operating speed, this switch is used to reduce the amount of screen data checked and rewritten. Bits 5 and 6 The following two operations are required to display a character at the cursor position and then move the cursor to the next position: 1. Display of the character at the cursor position The cursor is turned OFF and the character is displayed at the cursor position. 2. Cursor movement A space character is displayed where the cursor is ON. Bit 5 is used to control cursor ON/OFF condition and bit 6 determines whether the cursor will be displayed on the screen.
281 285		5	Used as temporary work area.
286 28B	CHRPTN	6	Contains the character font (result of subroutine CHRGEN).

```
    00056                        *  REGISTER PRESERVE (X) <--- (A,B)
    00057                        *
    00058           1000  A      DPCHEK EQU    *
    00059A 1000 72 017D  A              OIM    #$1,MIOSTS * SET FLAG   LCD IS ON WRITING
    00060A 1003 37                      PSH B
    00061A 1004 36                      PSH A
    00062A 1005 3C                      PSHX
    00063A 1006 CE 0286  A              LDX    #CHRPTL    * SET CHARACTER PATTERN TO 'CHRPTL'
    00064A 1009 BD FF67  A              JSR    CHRGEN
    00065A 100C 32                      PUL A
    00066A 100D 33                      PUL B
    00067A 100E 38                      PULX              * NOTE.  (X)  <---> (A,B)
    00068A 100F 81 13    A              CMP A  #19        * IS COLUMN LIMIT OUT OF RANGE ?
    00069A 1011 25 02 1015              BCS    NONOVX     * NO.
    00070A 1013 86 13    A              LDA A  #19        * YES. LIMIT= 19
    00071A 1015 C4 03    A       NONOVX AND B  #3
    00072                        *
    00073A 1017 39                      RTS
    00074                        *
    00075                        **
    00076                        * DISPLAY ONE CHARACTER TO REAL SCREEN WITHOUT WRITING TO SCREEN BUFFER
    00077                        * ON ENTRY
    00078                        *   (A): CHARACTER (ASCII CODE)
    00079                        *   (X): DISPLAY ADDRESS (HIGH:CLOLUMN, LOW:LINE)
    00080                        *
    00081                        *  ENTRY POINT
    00082           1018  A      DISPIT EQU    *
    00083A 1018 8D E6 1000              BSR    DPCHEK
    00084A 101A 3C                      PSHX              * SAVE VALUE OF (A,B)
    00085A 101B 20 15 1032              BRA    NONSET
    00086                        *  ENTRY POINT
    00087           101D  A      DSPLCH EQU    *
    00088A 101D 8D E1 1000              BSR    DPCHEK
    00089                        *
    00090A 101F 3C                      PSHX              * SAVE (A,B)
    00091                        *
    00092A 1020 37                      PSH B
    00093A 1021 36                      PSH A
    00094A 1022 86 14    A              LDA A  #20
    00095A 1024 3D                      MUL
    00096A 1025 30                      TSX
    00097A 1026 EB 00    A              ADD B  0,X        * ADDRESS OFFSET <--- (B)*WIDTH + (A)
    00098A 1028 A6 02    A              LDA A  2,X
    00099A 102A CE 0220  A              LDX    #PSBUF     * (X) <--- PHISICAL SCREEN BUFFER ADDRESS
    00100A 102D 3A                      ABX
    00101A 102E A7 00    A              STA A  0,X
    00102A 1030 32                      PUL A
    00103A 1031 33                      PUL B
    00104                        *CALCULATE ADDRESS IN LCD DRIVER
    00105           1032  A      NONSET EQU    *
    00106A 1032 37                      PSH B             * SAVE LOCATION POINTER (X,Y)
    00107A 1033 36                      PSH A
    00108                        *                          * ALREADY SAVE FOUR BYTES
    00109                        *                          *  STACK+0: COLUMN,   STACK+1:LINE
    00110                        *                          *  STACK+2: (A)       STACK+3:(B)
```

```
ERR   SEQ    LOC   OBJECT        PROGRAM  LCD          -----LCD DRIVER ROUTINE-----

     00001                                        NAM     LCD
     00002                                *
     00003                                *  LCD DRIVER'S ROUTINE
     C0004                                        TTL     -----LCD DRIVER ROUTINE-----
     00005                                * FILE NAME     'EXS9'   BY   K.A
     00006                                        OPT     LOAD
     00007                                        OPT     PAGE=55
     00008                                *
     00009                                * I/O PORT
     00010                                *  $28
     00011                                *        7:R LCD DRIVER BUSY (0:BUSY 1:READY)
     00012                                *
     00013                                *  $26
     00014                                *        0:W LCD COMMAND/DATA 1
     00015                                *        1:W LCD COMMAND/DATA 2
     00016                                *        2:W LCD COMMAND/DATA 4
     00017                                *        3:W LCD COMMAND/DATA SELCTION (0:DATA 1:COMMAND)
     00018                                *        4:W KEY BOARD INTERRUPT MASK (0:CLOSE 1:OPEN)
     00019                                *        5:W PERIPHERAL CONTROL (TO SERIAL)
     00020                                *        6:W TO PLUG IN 1
     00021                                *        7:W TO PLUG IN 2 AND SLAVE P40
     00022                                *
     00023                                * SUBROUTINE ENTRY POINT
     C0024            FED4   A            WRTP26 EQU     $FED4
     C0025            FF67   A            CHRGEN EQU     $FF67
     00026                                * WORK AREA
     00027            007D   A            MIOSTS EQU     $7D        * MAIN I/O STASTUS
     00028                                *                        * BIT 0: ON READ/WRITE TO LCD   1:READING/WRITING
     00029            0286   A            CHRPTL EQU     $286       * WORK AREA TO STORE CHARACTER PATTERN
     00030            0220   A            PSBUF  EQU     $220       * CHARACTER CODES ON PHISICAL SCREEN.
     00031            0280   A            DISSTS EQU     $280       * DISPLAY STATUS
     00032                                *                         *   , BIT 5: CURSOR ON WITH CHARACTER PATTERN FLA
     00033                                *                         *          (1:ON)
     C0034                                *                         *
     00035                                *                         *
     00036A 1000                                  ORG     $1000
     00037                                *
     00038                                * DISPLAY ONE CHARACTER TO REAL LCD SCREEN
     00039                                *  ROUTINE 'DISPIT': DISPLAY 1 CHARACTER TO LCD WITHOUT BUFFER
     C0040                                *         'DISPCH': DISPLAY ONE CHARACTER TO LCD AND WRITE TO BUFFER
     00041                                * ON ENTRY
     00042                                *   (A): ASCII CHARACTER CODE
     00043                                *   (X): LCD POSITION  (HIGH:COLUMN, LOW:LINE)
     C0044                                * ON EXIT
     00045                                *   (X): NEXT DATA POSITION (IF ILLEGAL ADDRESSING, CHANGE TO LEGAL)
     00046                                * REGISTER PRESERVE     A,B
     00047                                *
     00048                                *
     00049                                *
     00050                                * CHECK DISPLAY POSITION AND GENERATE CHARACTER FONT
     00051                                *  ON ENTRY
     00052                                *   (X):DISPLAY ADDREDSS (HIGH BYTE:COLUMN,  LOW BYTE:LINE)
     00053                                *   (A):DISPLAYED CHARACTER
     00054                                *  ON EXIT
     00055                                *   (A,B):MODEFIED POSITION
```

```
    00166A 1082 20 D3 1057                 BRA    WRTLOP    *
    00167                           *
    00168                           *
    00169A 1084 CC 0F08  A          ENDDIC LDD    #$0FD8    * CHIP SELECT OFF. COMMAND MODE
    00170A 1087 8D 53 10DC                 BSR    DATMOD    *
    00171A 1089 31                         INS              * RECOVER STACK POINTER  (+5)
    00172A 108A 38                         PULX
    00173A 108B 38                         PULX
    00174A 108C 32                         PUL A            * RESTORE POSITION ON LCD
    00175A 108D 33                         PUL B
    00176A 108E 4C                         INC A
    00177A 108F 81 14     A                CMP A  #20       * NEXT POINTER.
    00178A 1091 26 04 1097                 BNE    DIC100
    00179A 1093 4F                         CLR A
    00180A 1094 5C                         INC B
    00181A 1095 C4 03     A                AND B  #$03
    00182            1097 A         DIC100 EQU    *
    00183A 1097 71 FE7D  A                 AIM    #$FF-$1,MIOSTS * LCD FLAG   LCD NOT BUSY
    00184A 109A 38                         PULX
    00185A 109B 18                         XGDX            *RETURN NEXT DISPLAY POINT.
    00186A 109C 39                         RTS
    00187                           *
    00188                           *
    00189                           *
    00190                           * SELECT LCD DRIVER AND CALCULATE BANK AND ADDRESS POINTER
    00191                           * NOTE. SET TO $26 DIRVER ADDRESS, BUT NOT SET TO LCD DRIVER, ONLY RETUR
    00192                           *        LCD ADDRESS TO (A).
    00193                           * ON ENTRY
    00194                           *   (A): DOT LINE COLUMN POSITION (00 - 119)
    00195                           *   (B): LINE (0 - 3)
    00196                           * ON EXIT
    00197                           *   (A): DOT POINTER IN THE LCD DRIVER
    00198                           *   (B): CHIP NO. (BIT3=1)
    00199                           * REGISTER PRESERVE X
    00200                           *
    00201            109D A         LCADDR EQU    *
    00202A 109D 3C                         PSHX             * SAVE (X)
    00203A 109E 37                         PSH B            * STACK VALUE OF (B)
    00204A 109F 30                         TSX
    00205A 10A0 5F                         CLR B
    00206A 10A1 80 28     A         LCAD10 SUB A  #40       * (A) <--- ADDRESS AND BANK
    00207A 10A3 5C                         INC B            * (B) <--- CHIP NO.
    00208A 10A4 24 FB 10A1                 BCC    LCAD10
    00209A 10A6 8B 28     A                ADD A  #40       * GET START ADDRESS.   (B): 1-3
    00210A 10A8 6B 0100  A                 TIM    #$1,0,X   * CHECK BANK (ODD LINE NO. = BANK 1)
    00211A 10AB 27 02 10AF                 BEQ    LCAD20
    00212A 10AD 8A 40     A                ORA A  #$40
    00213            10AF A         LCAD20 EQU    *
    00214A 10AF 6B 0200  A                 TIM    #$2,0,X   * CHECK DRIVER CHIP (LINE >=2,  4-6)
    00215A 10B2 27 02 10B6                 BEQ    LCAD30
    00216A 10B4 CB 03     A                ADD B  #3        * CHIP IS 4 , 5 OR 6
    00217A 10B6 31                  LCAD30 INS
    00218A 10B7 38                         PULX
    00219A 10B8 36                         PSH A
    00220A 10B9 CA 08     A                ORA B  #$08      * BIT3= DATA MODE BIT (1:COMMAND)
```

```
     00111A 1034 37                     PSH B
     00112A 1035 48                     ASL A              * (A) <--- (A) * 6  (DOT COLUMN)
     00113A 1036 16                     TAB
     00114A 1037 48                     ASL A
     00115A 1038 1B                     ABA
     00116A 1039 33                     PUL B
     00117                        *
     00118                        *  WORK USE  STACK
     00119                        *     STACK     00: DOT COUNTER (1 CHARACTER = 6 DOT LINES)
     00120                        *               01,02: CHARACTER FONT ADDRESS
     00121                        *               03,04: DOT COLUMN, LINE
     00122                        *
     00123A 103A 37                     PSH B
     00124A 103B 36                     PSH A
     00125                        *
     00126                        * ***** LCD DRIVE RUOTINE ******  (X):CHARACTER PATTERN TOP ADDRESS
     00127                        * SET CHARCTER TO DRIVER
     00128A 103C CE 0286   A            LDX     #CHRPTL
     00129A 103F 3C                     PSHX
     00130A 1040 5F                     CLR B
     00131A 1041 37                     PSH B
     00132A 1042 30                     TSX
     00133           1043  A      DISCHL EQU     *          * SET COUNTER.
     00134A 1043 EC 03     A            LDD     3,X
     00135A 1045 8D 56 109D             BSR     LCADDR     * GET CHIP NO. & DATA ADDRESS.(DATADD,CHIPNO)
     00136A 1047 16                     TAB                * SAVE LCD ADDRESS TO (B)
     00137A 1048 86 64     A            LDA A   #$64       * SELECT WRITE MODE.
     00138A 104A 8D 76 10C2*            BSR     LCDMOD     * SET COMMAND
     00139A 104C 17                     TBA
     00140A 104D 8A 80     A            ORA A   #$80       * SET DATA ADDR TO LCD DRIVER.(AUTO INCREMENT)
     00141A 104F 8D 71 10C2*            BSR     LCDMOD
     00142A 1051 CC 0800   A            LDD     #$0800     * SET DATA MODE CODE FOR 'WRTP26' ROUTINE
     00143A 1054 BD 10DC   A            JSR     DATMOD     * LCD DRIVER: ENTER DATA MODE (NOT COMMAND)
     00144A 1057 E6 00     A      WRTLOP LDA B   0,X        * GET 8 BITS PATTERN
     00145A 1059 EE 01     A            LDX     1,X        * (B): DOT POSITION (0 - 5)
     00146A 105B 3A                     ABX
     00147A 105C A6 00     A            LDA A   0,X
     00148A 105E C1 05     A            CMP B   #5         * LAST DOT (6 TH): WITHOUT CURSOR
     00149A 1060 27 09 106B             BEQ     DISC20
     00150A 1062 F6 0280   A            LDA B   DISSTS     * CURSOR ON ?
     00151A 1065 C5 20     A            BIT B   #$20
     00152A 1067 27 02 106B             BEQ     DISC20
     00153A 1069 8A 80     A            ORA A   #$80
     00154A 106B 8D 55 10C2       DISC20 BSR     WITDAT     * WRITE ONE BYTE BIT PATTERN
     00155A 106D 30                     TSX
     00156A 106E 6C 00     A            INC     0,X        * COMPLETE TO WRITE 6 BYTES ?
     00157A 1070 A6 00     A            LDA A   0,X
     00158A 1072 81 06     A            CMP A   #6
     00159A 1074 27 0E 1084             BEQ     ENDDIC     * YES . END
     00160A 1076 6C 03     A            INC     3,X
     00161A 1078 A6 03     A            LDA A   3,X        * INCREMENT DATA ADDRESS
     00162A 107A 81 28     A            CMP A   #40        * BOUNDARY OF DRIVER = 40.
     00163A 107C 27 C5 1043             BEQ     DISCHL
     00164A 107E 81 50     A            CMP A   #80
     00165A 1080 27 C1 1043             BEQ     DISCHL     * CHIP LAST ADD ?
     00276                            *
     00277A 10F3 8D 18 1100                  BSR    LCDCLR    * CLEAR SCREEN .
     00278                            *
     00279A 10F5 86 09     A                 LDA A  #$09      * DISPLAY ON COMMAND.
     00280                            *
     00281                            *
     00282                            ******* SET COMMAND ALL DRIVERES.
     00283                            * SET COMMAND TO LCD DRIVER
     00284                            * ON ENTRY
     00285                            *   (A): COMMAND FOR LCD DRIVER
     00286                            * ON EXIT
     00287                            * REGISTER PRESERVE  X
     00288                            * NOTE. THIS ROUTINE MUST BE CALL ONLY 'INITLC.
     00289                            *      BECAUSE COMMAND WILL BE CHANGED.
     00290                            *
     00291A 10F7 5F                  STRALL CLR B             * (B): DRIVER NUMBER
     00292A 10F8 5C                  STRA10 INC B
     00293A 10F9 37                         PSH B
     00294A 10FA CA 08     A                ORA B  #$8
     00295A 10FC 36                         PSH A
     00296A 10FD 86 0F     A                LDA A  #$0F       * SELECT DRIVER AND COMMAND MODE
     00297A 10FF BD FED4   A                JSR    WRTP26
     00298A 1102 32                         PUL A
     00299A 1103 8D BD 10C2                 BSR    LCDMOD     * SET COMMAND TO DRIVER
     00300A 1105 84 FD     A                AND A  #$FF-$2    * TO CHANGE '1E' (SMM COMMAND) COMMAND TO '1C'
     00301A 1107 33                         PUL B             *  OTHER COMMANS ARE $10, $08, $09.(NOT EFFECT
     00302A 1108 C1 06     A                CMP B  #6
     00303A 110A 26 EC 10F8                 BNE    STRA10
     00304A 110C 39                         RTS
     00305                            *
     00306                            *
     00307                            * CLEAR LCD SCREEN
     00308                            * ON ENTRY
     00309                            *   PARAMETER NONE
     00310                            * ON EXIT
     00311                            *   (X):0
     00312                            *  REGISTER PRESERVE NONE
     00313                            *
     00314             110D  A       LCDCLR EQU    *
     00315A 110D CE 0000   A                LDX    #0         * POINTER SET.
     00316A 1110 86 20     A         LCDC10 LDA A  #$20       * SET SPACE CODE.
     00317A 1112 BD 101D   A                JSR    DSPLCH
     00318A 1115 08                         INX               * IX HAS DISPLAY POINTER.
     00319A 1116 09                         DEX
     00320A 1117 26 F7 1110                 BNE    LCDC10     * NOT END.
     00321A 1119 39                         RTS
     00322                            *
     00323                            *
     00324                            * DISPLAY CHARACTER STRING TO LCD
     00325                            * ON ENTRY
     00326                            *   (B): NEMBER OF CHARCTER STRING (0 - 80)
     00327                            *   (X): ADDRESS OF DATA PACKET
     00328                            *      DATA PAKET:(ADDRESS X), (ADDRESS Y), DATA1,.....
     00329                            * ON EXIT
     00330                            *   PARAMETER  NONE
```

```
00221A 10BB 86 0F      A              LDA A  #$0F      * SET CHIP NO.
00222A 10BD BD FED4    A              JSR    WRTP26    * SELECTED DRIVER CHIP, AND ENTER COMMAND MODE
00223                          *
00224A 10C0 32                        PUL A            * SET DATA ADDRESS TO DRIVER
00225A 10C1 39                        RTS
00226                          *
00227                          *
00228            10C2  A       WITDAT EQU    *
00229            10C2  A       LCDMOD EQU    *
00230A 10C2 37                        PSH B
00231A 10C3 16                        TAB
00232A 10C4 36                        PSH A
00233A 10C5 07                        TPA
00234A 10C6 0F                        SEI
00235A 10C7 3C                        PSHX
00236A 10C8 D7 2A      A              STA B  $2A       * SET ADD OR MODE.
00237A 10CA 7D 0028    A       LCDM10 TST    $28       * 7 BIT IS LCD BUSY FLAG.
00238A 10CD 2A FB 10CA                BPL    LCDM10    * WAIT.
00239A 10CF DE 2A      A              LDX    $2A       * LDD SEND 2 PULSES ,SO 2 BIT SHIFT
00240A 10D1 DE 2A      A              LDX    $2A
00241A 10D3 DE 2A      A              LDX    $2A
00242A 10D5 DF 2A      A              LDX    $2A
00243A 10D7 38                        PULX
00244A 10D8 06                        TAP
00245A 10D9 32                        PUL A
00246A 10DA 33                        PUL B
00247A 10DB 39                        RTS
00248                          *
00249                          *
00250                          * AFTER CHECK LCD BUSY, CALL 'WRTP26'
00251                          * ON ENTRY, (SAME AS 'WRTP26')
00252                          *  (A):TARGET BIT POSITION
00253                          *  (B): DATA
00254                          *
00255            10DC  A       DATMOD EQU    *
00256A 10DC 7D 0028    A              TST    $28       * 7 BIT IS LCD BUSY FLAG.
00257A 10DF 2A FB 10DC                BPL    DATMOD    * WAIT.
00258A 10E1 7F FED4    A       DTMD10 JMP    WRTP26    * SET INTERRUPT MASK.
00259                          *
00260                          *
00261                          **
00262                          *
00263                          * INITIALIZE LCD ROUTINE
00264                          *    DRIVER INITIALIZE AND CLEAR CURSOR ON FLAG
00265                          * ON ENTRY PARAMETER NONE
00266                          *
00267A 10E4 86 10      A       INITLC LDA A  #$10      * SFF (SET FRAME FREQUENCY) COMMAND
00268A 10E6 8D 0F 10F7                BSR    STRALL    * SET IT FOR EACH DRIVER.
00269A 10E8 86 1E      A              LDA A  #$1E      * ACCA : SMM (SET MULTIPLEXING MODE) COMMAND
00270A 10EA 8D 0B 10F7                BSR    STRALL    *    NOTE. 1 ST DRIVER: SMM VALUE=$1E
00271                          *                                   2ND - 6 TH DRIVER: SMM=$1C
00272A 10EC 86 08      A              LDA A  #$08      * ACCA : DISP OFF COMMAND. ACCB : CHIP NO.
00273A 10EE 8D 07 10F7                BSR    STRALL
00274                          *
00275A 10F0 7F 0280    A              CLR    DISSTS    * CLEAR DISPLAY STATUS FOR NON CURSOR CLEAR.
```

CHAPTER 4 SERIAL COMMUNICATION

Page

4.1 General.. 4-1
4.2 I/O Ports.. 4-2
4.3 Serial Communication Procedure 4-3
4.4 Control Lines.. 4-4
4.5 High-Speed Serial Communication................................ 4-5
4.6 Subroutines for Serial Communication........................... 4-9
4.7 High-Speed Serial Subroutines.................................. 4-10
4.8 High-Speed Serial Communication Work Areas..................... 4-12

APPENDIXES Serial Communication Protocol............................ 4-13
Sample List.. 4-25

```
     00331                          * REGISTER PRESERVE NONE
     00332                          * ENABLE REENTRANT
     00333                          *   WORK USE  : STACK   STACK +  0,1 :LOCATION OF CHARACTER ON LCD
     00334                          *                                 2,3 :ADDRESS OF STORED CHARACTER
     00335                          *                                 4   :DISPLAYED CHARACTER NUMBER
     00336A 111A 5D                 DSPLCN TST B              * IF (B)=0, CLEAR SCREEN.
     00337A 111B 27 F0 110D                BEQ     LCDCLR
     00338A 111D 37                        PSH B
     00339A 111E 3C                        PSHX
     00340A 111F EE 00      A              LDX     0,X        * GET LOCATION OF DISPLAY
     00341A 1121 3C                        PSHX               *
     00342A 1122 5F                        CLR B              * COUNTER OF DISPLAYED CHARACTER
     00343A 1123 30                        TSX
     00344A 1124 EE 02      A       DSPL10 LDX     2,X
     00345A 1126 3A                        ABX                *
     00346A 1127 A6 02      A              LDA A   2,X        * (A) <--- DISPLAYED CHARACTER
     00347A 1129 38                        PULX               * (X): LOCATION ON LCD
     00348A 112A BD 101D    A              JSR     DSPLCH     * DISPLAY ONE CHARACTER TO SCREEN.
     00349A 112D 3C                        PSHX
     00350A 112E 5C                        INC B
     00351A 112F 30                        TSX
     00352A 1130 E1 04      A              CMP B   4,X        * FINISHED ?
     00353A 1132 26 F0 1124                BNE     DSPL10
     00354A 1134 38                        PULX
     00355A 1135 38                        PULX
     00356A 1136 33                        PUL B
     00357A 1137 39                        RTS
     00358                          *
     00359          0000   A               END
***** TOTAL ERRORS       0
```

4.1 General

Serial communication is performed by the start-stop synchronous transmission system. In start-stop transmission, the signal is logic 1 while no data is being sent and becomes Ø to show the start of data. (See Fig. 4-1.) This first bit is called the start bit. Bits are then transmitted sequentially from the lowest order bit. A stop bit (logic 1) is always included to end transmission of the character. Stop bits may be 1 or 2 bits long. The word length (data bits) must be 5 to 8 bits. Bit time varies according to the bit rate. At 3ØØ BPS (bits per second), the duration of a single bit is 3.3 msec.

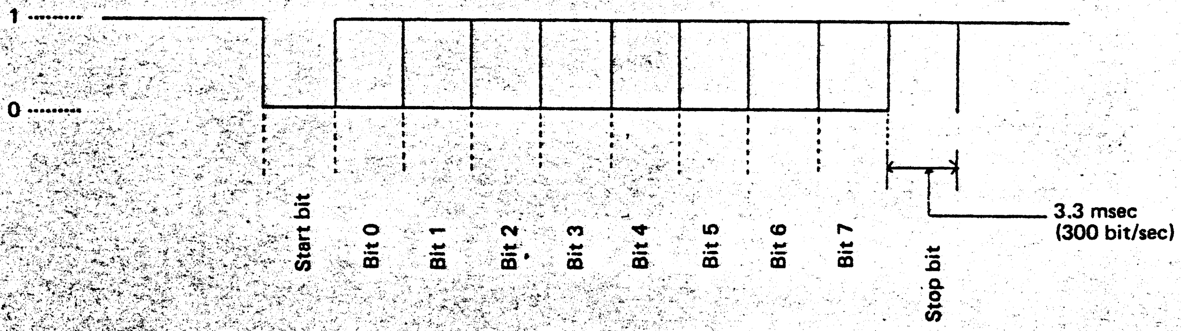


Fig. 4-1 Start-stop Data Transmission Format

The figure below shows an example of signal status when data 3A (ØØ111Ø1Ø$_2$) is transmitted in start-stop format with a single stop bit.

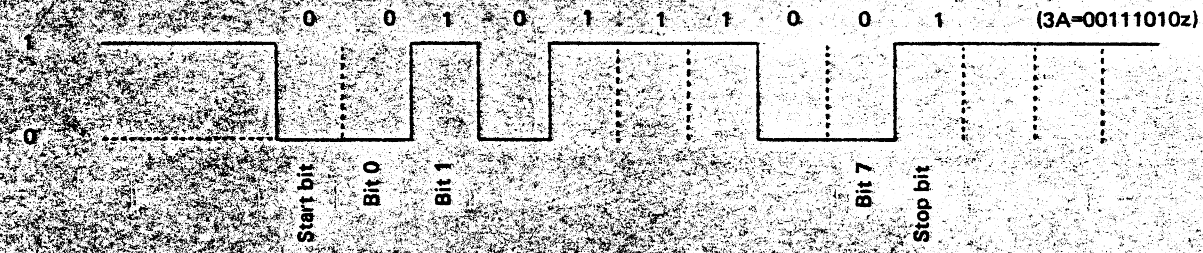


Fig. 4-2 Start-stop DATA 3A

Data 1 is represented by a low signal (-3 to -8V) and data Ø by a high signal (+3 to +8V) as shown in Fig 4-3 below.

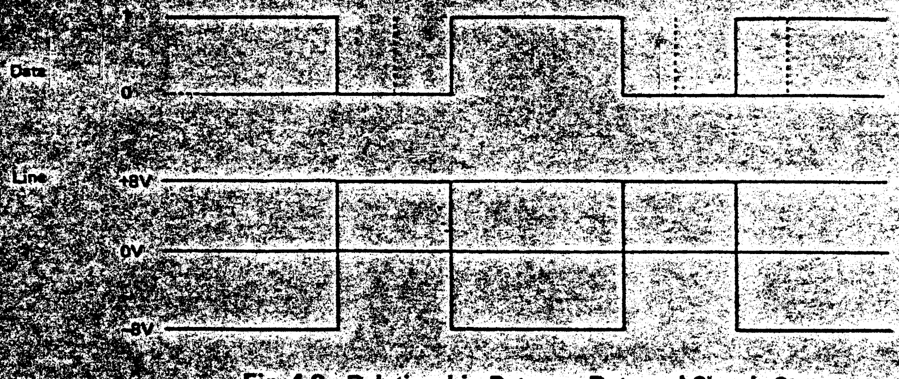


Fig. 4-3 Relationship Between Data and Signals States

The status signal lines are RTS (output), CTS (input), DTR (output), DSR (input), and CD (input). These signals are ON when high and OFF when low (Fig. 4-4).

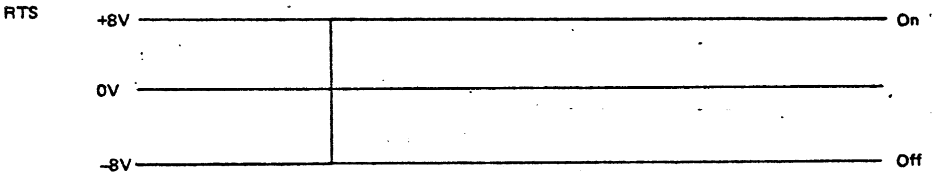


Fig. 4-4 Signal Line Output Status (RTS)

The HX-20 is provided with two types of interfaces. These are serial and RS-232C. The serial interface uses the serial port of the MCU and the bit rates and word length are fixed. The RS232C interface, however, performs handshaking by software. It can therefore support bit rates up to 4800 bps and both the bit rate and word length can be set by the user. Fig. 4-5 shows the respective range of functions for the serial and RS-232C interfaces.

	Transmission speed	Word length (bits)	Stop bit (bits)	Control lines (input)	Control lines (output)
Serial	38.4K BPS 4.8K BPS 600 BPS 150 BPS	8		1	1
RS232C	4.8K BPS max.	5, 6, 7 or 8	1 or 2	3 (CTS, DSR, and CD)	2 (RTS and DTR)

Fig. 4-5 Functions of Serial and RS-232C Interfaces

The serial interface is used for communication between the master and slave MCUs and for the floppy disk units.

4.2 I/O Ports

Table 4-1 I/O Ports for Serial Communication

MCU	Port (address)	Input/ output	Signal name or function	Signal state	Port bit state
Master	P10	Input	DSR (RS-232C)	High	0
				Low	1
	P11	Input	CTS (RS-232C)	High	0
				Low	1
	P16	Input	PIN (serial control line)	High	0
				Low	1
	P21	Output	TXD (RS-232C)	High	0
				Low	1
	P22	Output	Selection of slave or serial for CPU serial communication		0: Slave 1: Serial
	RMCR (0010)	Output	Serial bit rate control		

MCU	Port (address)	Input/ output	Signal name or function	Signal state	Port bit state
	TRCSR (0011)	Input	Serial control and status		
	SRDR (0012)	Input	Serial receive data		
	STDR (0013)	Output	Serial transmit data		
	$26 Bit 5	Output	POUT (serial control line)	High	0
				Low	1
Slave	P20	Input	RXD (RS-232C)	High	0
				Low	1
	P31	Output	RTS (RS-232C)	High	0
				Low	1
	P36	Output	Serial and RS-232C interface driver ON/OFF		0: On 1: OFF
	P45	Output	P20 signal selection		0: RS-232C 1: Micro-cassette
	P47	Input	CD (RS-232C)	High	0
				Low	1

Note: DSR: Data set ready
CTS: Clear to send
TXD: Transmit data
RTS: Request to send
DTR: Data terminal ready
RXD: Receive data
CD: Carrier detect

4.3 Serial Communication Procedure

The SCI (serial communication interface) in the MCU performs serial communication in the following procedure.

(1) Driver ON

The communication driver is turned ON. The port for driver ON is connected to the slave MCU. Subroutine SERONF turns the drivers ON/OFF.

(2) Serial switching

The serial communication lines of the MCU can be used either for external data communication or for communication with the slave MCU. Normally, the slave MCU is selected. To select external communication, port P22 of the main MCU is set to 1.

(3) Bit rate setting

RMCR (address 10) sets the bit rate. The bit rate is normally set to 38.4K BPS. Table 4-2 shows selection of bit rates by RMCR.

Table 4-2 SCI Bit Time and Bit Rates

Lower 4 bits	Hexadecimal	Bit time/bit rate
0100	4	26 µsec/38.4 BPS
0101	5	208 µsec/4.8K BPS
0110	6	1.67msec/600 BPS
0111	7	6.67msec/150 BPS

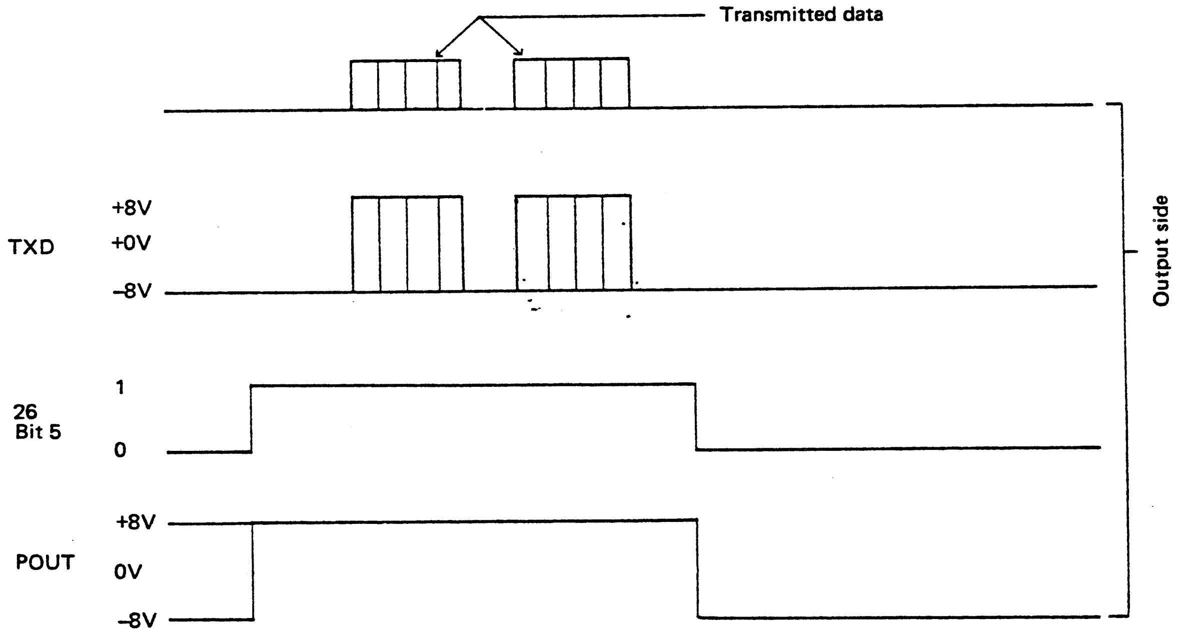
(4) Data transmission (one byte)
TRCSR (address 0011) is input and when it is confirmed that TDRE (bit 5 of TRCSR) is 1, one byte of data is transmitted by writing it to TDR(address 0013).

(5) Data reception (one byte)
TRCSR is input and if RDRF (bit 7 of TRCSR) is 1, serial data can be received by SRDR (address 12). One byte of Serial data is then received. Note that if the received data is not fetched before the next data is received, an overrun error occurs (ORFE is set at 1).

(6) Termination procedure
The bit rate is set to 38.4K bps (procedure 3) and the driver is turned OFF. This procedure is followed because transmission of commands to the slave MCU is always performed at 38.4K PBS.

4.4 Control Lines

Two control lines are available: PIN (input) and POUT (output). PIN is connected to P16 (bit 6 of port 1) and POUT is connected to bit 5 of address 26. Both of these signals are set at 1 when the signal goes high and at 0 when the signal goes low. Subroutine WRTP26 is used to set data in address 26. Fig. 4-6 shows the relationship of the signals and ports.



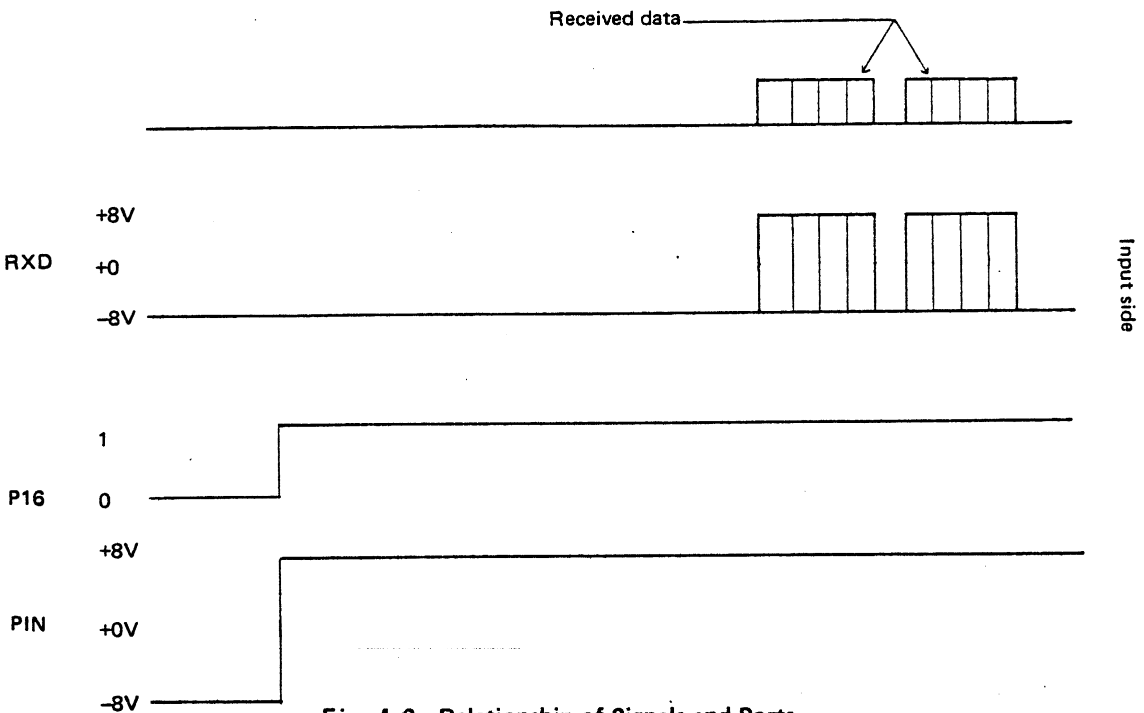


Fig. 4-6 Relationship of Signals and Ports

4.5 High-Speed Serial Communication

EPSP (EPSON Serial Communication Protocol) is provided to enable serial communication between the HX-20 and a floppy disk unit (TF-20) or between two HX-20s.

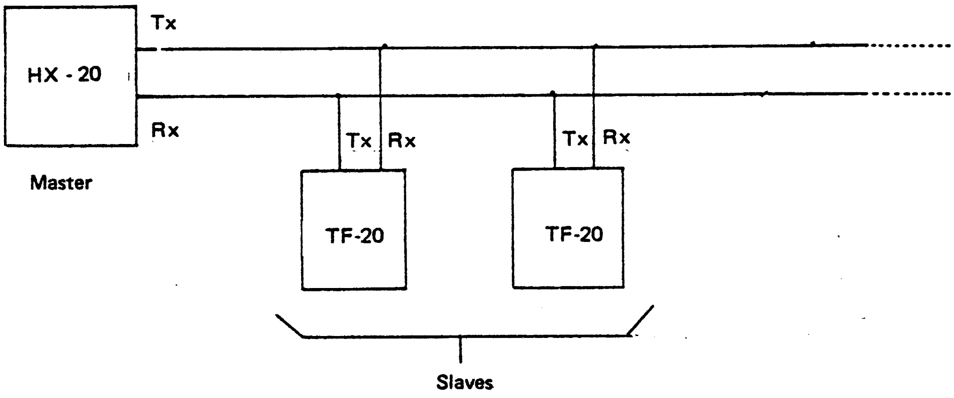


Fig. 4-7 Connection of Slave Devices to HX-20 for Serial Communication

Fig. 4-7 shows how slave devices can be connected by data lines to the HX-20. Up to two slave devices can be connected to a single master device. Each slave is assigned a device number by the master. The master then uses the device number to select which of the slaves to perform communication with. The master can only communicate with one slave at a time. Communication between slave devices cannot be performed.

Fig. 4-8 shows the format for messages sent from the HX-20 to a slave device.

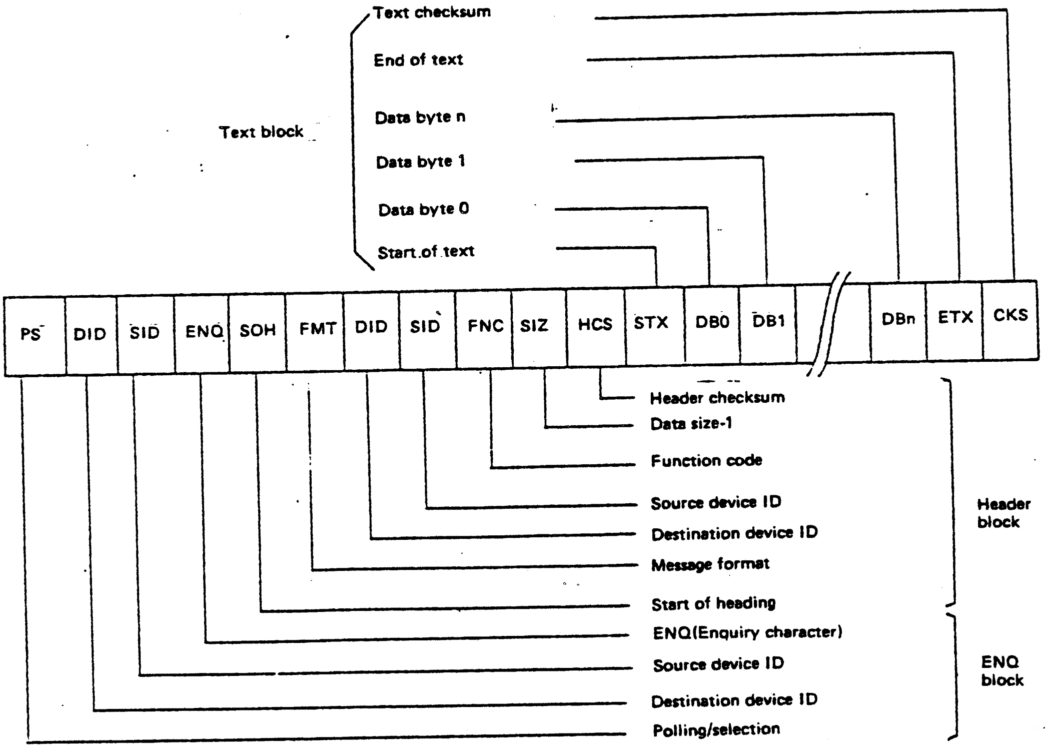


Fig. 4-8 Message Format

Messages sent from the master can be divided into three blocks described below.

(1) ENQ block
PS to ENQ in Fig. 4-8. The master sends this block to request connection with a slave.

(2) Header block
SOH to HCS in the above figure.
This block specifies the data format etc.

(3) Text block
STX to CKS in the above figure. The text block contains the actual data transmitted.

Details of each block are as follows.
ENQ block
The contents of the ENQ block are shown below

PS	DID	SID	ENQ

The master device selects one of the slave devices and issues a connection request to it, using this block. When connection with the slave has been established, the header and text blocks are sent. Once a slave device has been connected, this procedure is not repeated until a new slave device is selected for communication. The selected slave device issues an ACK signal in response to the connection request from the master (Fig. 4-9).

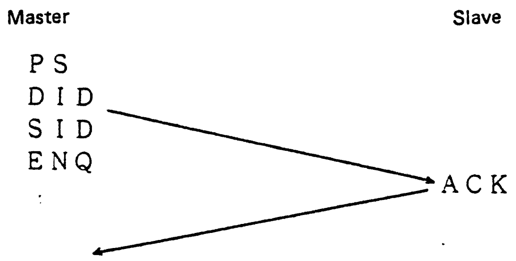


Fig. 4-9 ENQ Block Procedure

PS specifies polling/selection. At present, however, only selection is supported. The code for PS is 31_{16} = 1.
DID indicates the destination device ID. The following destination device IDs are available.

31_{16}	: Floppy disk drive A
32_{16}	: Floppy disk drive B
33_{16}	: Floppy disk drive C
34_{16}	: Floppy disk drive D
20_{16}	: Master HX-20

The code for NEQ is 05_{16}.

Header block
The master transmits the header block to specify the message format and the function codes as well as text size to be sent to the floppy disk unit in the text block that follows.
The contents of the header block are shown below.

SOH	FMT	DID	SID	FNC	SIZ	HCS

SOH: Indicates the start of the header. The value is 01.
FMT: Indicates the header block format.
00 indicates that the master device is transmitting a message to a slave device.
01 indicates that a slave device is transmitting a message to the master device.
DID: Indicates the destination device ID. The codes for DID are the same as in the ENQ block.
SID: Indicates the source device ID.
FNC: Specifies the function of the disk unit. Must be 00 to FF. For details of each function, refer to the descriptions in the corresponding sections.
SIZ: Indicates the text block size. This value is the number of bytes in the text block (excluding STX and CKS) minus 1. The value of SIZ must be in the range 0 to 255_{10}.
HCS: Indicates the checksum of the header block. The value is such that the lower 8 bits of the sum of the values of the header block (SOH to HCS) will all be 0.

When the slave device receives a correct header block, it responds by sending 'ACK' to the corresponding source device. If the slave device receives an incorrect header block, it responds by sending 'NCK' to the source device.

Text block
The text block contains the actual data to be sent to the selected device. The text block follows the header block.
The contents of the text block are shown at right.

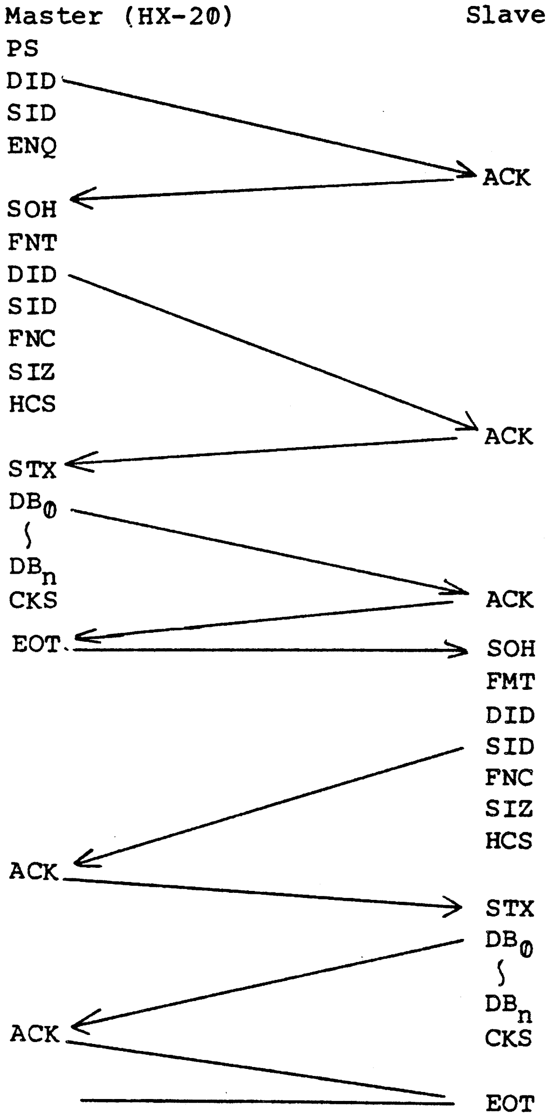
STX	DB_0	DB_1	~	DBn	CKS

STX: Indicates the start of text. The value is 02.
DB_0: Data 0
DB_n: Data n (n < 255)
EXT: Indicates the end of text.
CKS: Indicates the checksum of the text block. The value is such that the lower bits of the sum of the values of the text block (STX to CKS) will be 0.

When the slave device receives a correct text block, it responds by sending ACK to the source device. If the slave device receives an incorrect text block, it responds by sending 'NCK' to the source device.

Switching Transmit State

There are cases when the master (HX-20) will request data transmission from a slave device (e.g., floppy disk unit). In this case, the sending and receiving sides (master and slave) are reversed. Switching over from master-to-slave to slave-to-master data transmission is accomplished by the following procedure.
The master sends EOT (code 04) to the slave after it has received an ACK from the slave indicating the slave has correctly received the text block. The slave device, after receiving EOT, sends the header and text blocks to the master device. It then sends EOT to the master and the transmit state returns to master-to-slave. (Fig. 4-11).



Details of protocol are shown in the EPSP standard at the end of this chapter.

4.6 Subroutines for Serial Communication

The following four subroutines support serial communication using EPSP procedures:

1. SERONF: Turns ON/OFF the serial communication drivers.
2. SEROUT: Transmits the ENQ, header, and text blocks.
3. SERIN: Receives the header and text blocks.
4. SRIWIT: Sets constants and performs initialization.

4.7 High-Speed Serial Subroutines

Subroutine name	Entry point	Description
SERONF	FF73	Turns ON/OFF the high-speed serial driver. This subroutine checks bit 4 of 'SRSTS' and turns ON the driver only when both are off.
		The contents of the SERONF parameters are the same as those of RSONOF.
SEROUT	FF70	High-speed serial data output (EPSP-based data transmission). This subroutine transmits the ENQ, header, and text blocks to the specified device according to the ENQ... SOH... STX procedure.
		Parameters: At Entry (X): Head address of a data packet (A): Indicates whether to proceed to the receive procedure after completion of the transmit procedure. 00: Transmit procedure only. 01: (LSB=1) proceeds to the receive procedure after completion of the transmit procedure. Packets 1. EMT (1 byte) 2. DID (1 byte) 3. SID (1 byte) 4. FNC (1 byte) 5. SIZ (1 byte) 6. (data string) (1 byte) ∫ n At Return (C): Abnormal I/O flag (A): Return codes 00: Normal end B0: Time out B1: Not linked (device error) B2: Communication error B3: Driver OFF. (Z): according to the value of (A) Registers retained None Subroutines referenced CHKRS Variables used R0, R1, R2, R3, R4, and R5H
SERIN	FF6D	Receives the header and text blocks according to the SOH ... STX ... procedure (high-speed serial data block reception.)
		Parameters: At Entry (X): Head address of receive data block.

Subroutine name	Entry point	Description
		Parameters: At Return (C): Abnormal I/O flag (A): Return codes 00: Normal B0: Time out B2: Error during receive procedure (B): Indicates the receive block status when (A) is 00. 00: Data with a header string (SOH...) received. 01: Data without a header string received. (Z): According to the value of (A). Note: The format of a data block received is the same as that of the data block transmitted. 1. FMT 2. DID 3. SID 4. FNC 5. SIZ 6. Data ≀ n Registers retained None Subroutines referenced CHKRS Variables used R0, R1, R2, R3, R4, and R5H

SRINIT	FFIC	Sets constants and performs high-speed serial initialization.
		Values of constants on initialization.
		SRTCN ← 3
		SRTMO ← 10_{10}
		SREMO ← 100_{10}
		SRAMO ← 10_{10}
		SRTDL ← 1
		Others ← 0

		Parameters: At Entry (A): Value of SRMODE 00 or 01 (00: master) At Return None Registers retained None Subroutines referenced None Variables used None

4.8 High-Speed Serial Communication Work Areas

Address	Variable name	Byte count	Description
1C4 1C4	SRFMT	1	FMT (format) data
1C5 1C5	SRDDEV	1	DID (Destination Device ID) data
1C6 1C6	SRSDEV	1	SID (Source Device ID) data
1C7 1C7	SRFNC	1	FNC (function) data
1C8 1C8	SRSIZ	1	SIZ (Size) data
1C9 1C9	SRACKC	1	ACK character (Sent from destination device on completion of block transmission)
1CA 1CA	SRTRCN	1	Number of time same block has been sent.
1CB 1CB	SRTIMO	1	Time out for received characters (unit: msec)
1CC 1CC	SRETMO	1	Time out for received block reception (unit: msec)
1CD 1CD	SRATMO	1	Time out for received ACK characters (unit: msec)
1CE 1CE	SRMODE	1	Relationship between devices (0: Master, any other value: Slave)
1CF 1CF	SRETDL	1	Idle time after EOT transmission (unit: msec)
1DO 1DO	SRBLCN	1	Number of received data (block reception)
1D1 1D1	SRERMD	1	Error (block reception)
1D2 1D2	SRRVFL	1	Not used
1D3 1D4	SREIX	2	Address where received data is stored (block reception)

APPENDIX SERIAL COMMUNICATION PROTOCOL (EPSP)

1. BASIC LINE SPECIFICATION

1.1 Transmission Speed
1.2 Synchronization
1.3 Communication
1.4 Transmission
1.5 Response System
1.6 Error Control
1.7 Transmission Codes
1.8 Bit Transmission Sequence bit 0, bit 1, ... bit 7

2. TRANSMISSION CHARACTERS AND SEQUENCE

PS
DID
SID — Request receiving side to prepare to receive data
ENQ

SOH Indicates start of header block.
STX Indicates start of text block.
ETX Indicates end of text block.
ACK Acknowledge
NAK Negative acknowledge
DLE ; Waits for WAK, acknowledge or transmission.
ENQ Prompt for block response.
EOT Releases data lines.

PS must be '1'=31_{16}. Control characters, DID and SID must be 8 bits (MSB=0).

3. MESSAGE FORMAT

3.1 Header Format

SOH Start of header

FMT Text format
00: Indicates that the master is transmitting a block.
01: Indicates that a slave is transmitting a block.

DID Destination ID

SID Source ID

FNC Text function

SIZ Text size (in bytes)
This value is the length of the text block (excluding STX, ETX and CSK) minus 1.

HCS Checksum of header block
This is a value such that the lower 8 bits of the sum of SOH to ACS are 0.

3.2 Text Format

STX: Start of text
DB∅: Data ∅
DB1: Data 1
∫
DBn: Data n
ETX: End of text
CKS Checksum of a text block

The value of CKS is such that the lower 8 bits of the sum of STX to CKS are ∅.

Text length excluding STX, ETX and CKS must be within 256 bytes.

4. RESPONSE TO SLAVE SELECTION SEQUENCE

(a) ACK (Acknowledgement)

Indicates that that slave can receive a block. The master then initiates data transmission.

(b) NAK (Negative acknowledgement)

Indicates that the corresponding I/O device is not connected or that an error has occurred and the slave cannot receive data. The master then issues EOT and terminates the data link.
The master will also send EOT to terminate the data link by transmitting if no response is received within a fixed period of time or an invalid response other than ACK and NAK is received after a selection sequence has been sent.

5. HEADER BLOCK TRANSMISSION

5.1 Response to a Header Block

(a) ACK (Acknowledge)

Indicates that the slave has received a correct header block. The master proceeds to the next phase.

(b) NAK (Negative acknowledge)

Indicates that the slave has received an incorrect header block. In this case, the master repeats transmission of the same block. If the master still receives NAK after the block has been transmitted a specified number of times, it assumes a line error and terminates the data link (by send EOT).

(c) WAK (Acknowledge and temporary wait)

Indicates that the slave has received a correct block but that it cannot yet receive the next block. The master will wait and then issue ENQ to prompt a response from the slave.

(d) No response or invalid response

If no response is made within a given time or a response other than ACK, NAK or WAK is received, the master will issue ENQ to prompt a response from the slave.

If no correct response is received even after ENQ has been transmitted a specified number of times, the master assumes an error and terminates the data link.

6. TERMINATION

(a) When, after sending EXT to the slave, the master receives ACK, it sends EOT to the slave and terminates the data link.

(b) When a transmission error occurs after the data link has been established, or during data transmission, the master will terminate the data link by transmitting EOT.

7. TIME SUPERVISION

(1) Number of selection sequences transmitted

The master will repeat the selection sequence after receiving a response other than ACK from the slave for the number of times listed in the table below.

	Mode 0	Mode 1
NAK	One time	One time
No response or invalid response	Three times (at 1-sec intervals)	Three times (at 3-sec intervals)

(2) Number of transmitted ENQs (response retransmit request)

	Mode 0	Mode 1
No response or invalid response	Three times (at 1-sec intervals)	Three times (at 3-sec intervals)

(3) Timers

	Mode 0	Mode 1
Response wait timer	1 sec	3 sec
Interblock supervision	32 sec	96 sec
Character supervision	1 sec	3 sec

8. TERMINAL NUMBERS

31_{16} : Floppy disk drive A
32_{16} : Floppy disk drive B
33_{16} : Floppy disk drive C
34_{16} : Floppy disk drive D

Terminal numbers (slave): 30_{16} through $3F_{16}$ for mode 0
40_{16} through $5F_{16}$ for mode 1
Center number (master) : 20_{16}

9. OMISSION OF A HEADER BLOCK

If the terminal previously transmitted to is still selected and the header to be transmitted is the same as the last transmission, the header may be omitted. In this case, the master need only transmit the data block following STX. The slave treats this data block without header as if it included the header of the previously received data block.

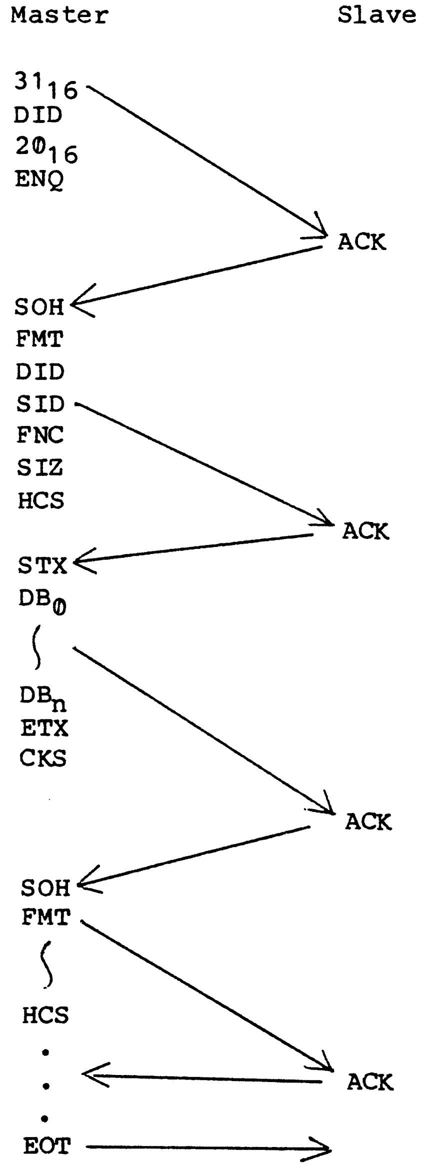
10. TRANSMISSION CONDITIONS SUPPORTED BY VER-1

Transmission speed is 38.4K BPS.
Mode 0 is used for a time supervision.
Header block cannot be omitted.

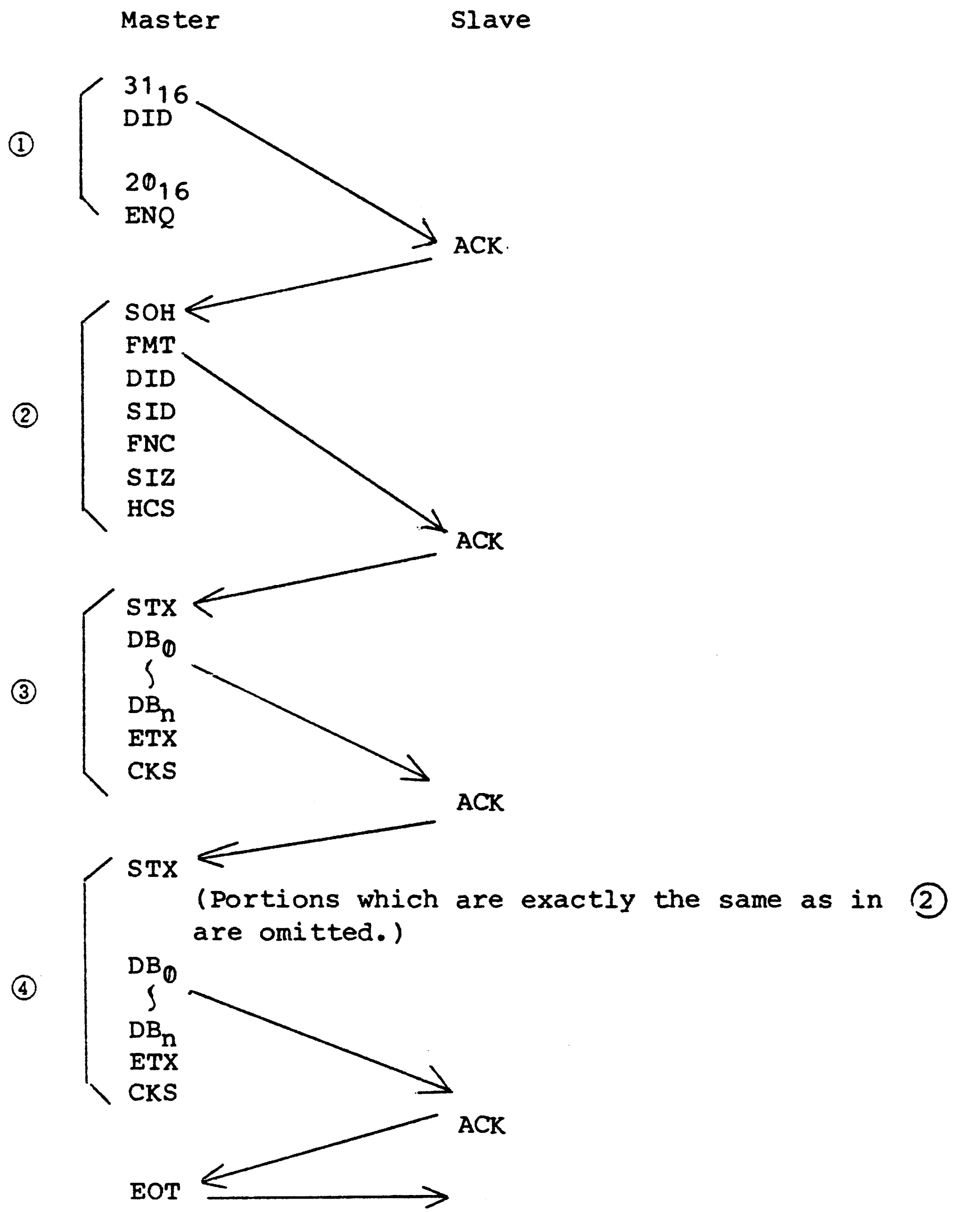
TRANSMISSION PROCEDURE DIAGRAMS

1. WITHOUT ERRORS

(1) When the slave does not send a data block to the master in response to the master's transmission and a header is not omitted.

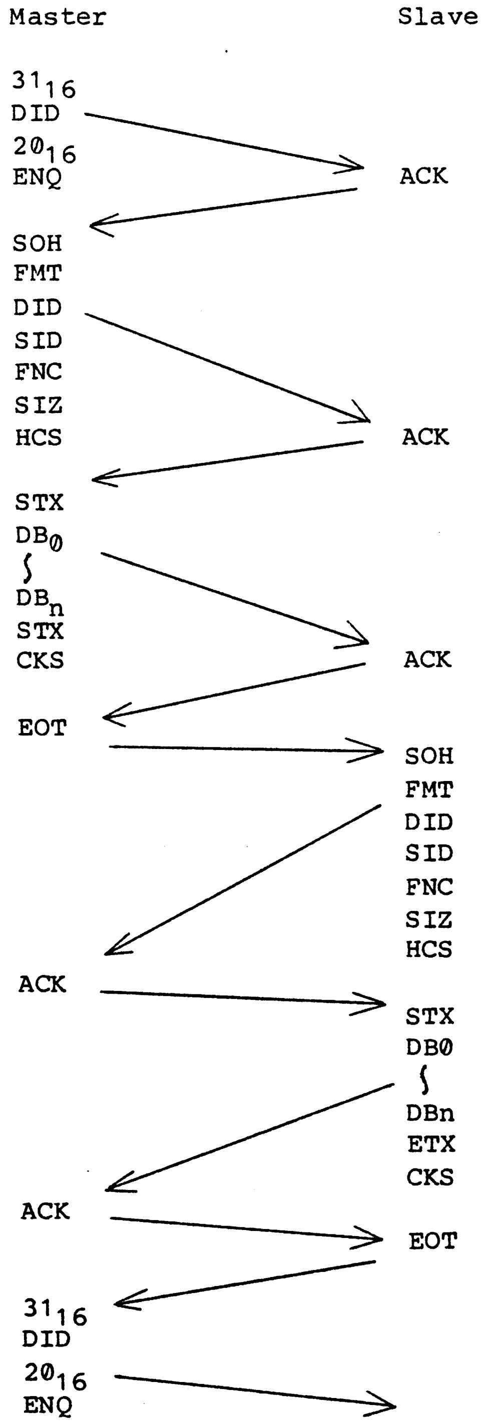


(2) When the terminal does not transmit a data block to the master in response to the master's transmission but the header is omitted.

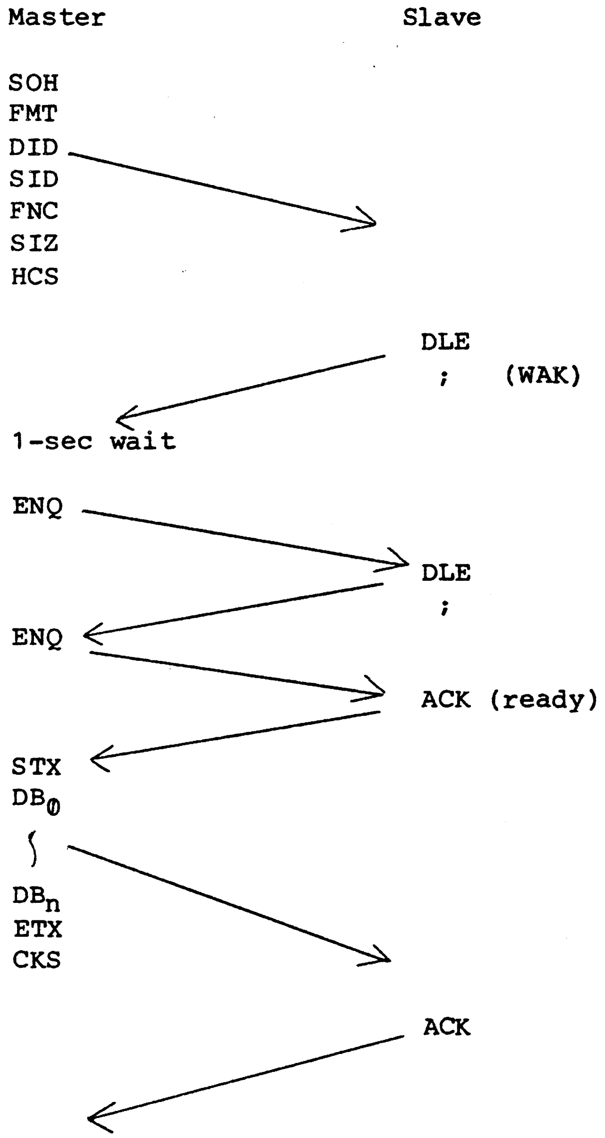


(Portions which are exactly the same as in ② are omitted.)

(3) When the terminal transmits a data block to the master in response to EOT from the master and the header is not omitted.

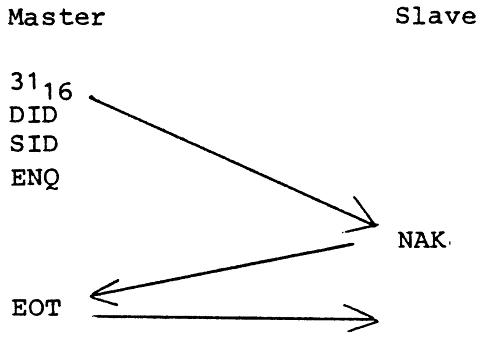


(4) When the slave responds with WAK to a block transmission with header from the master.

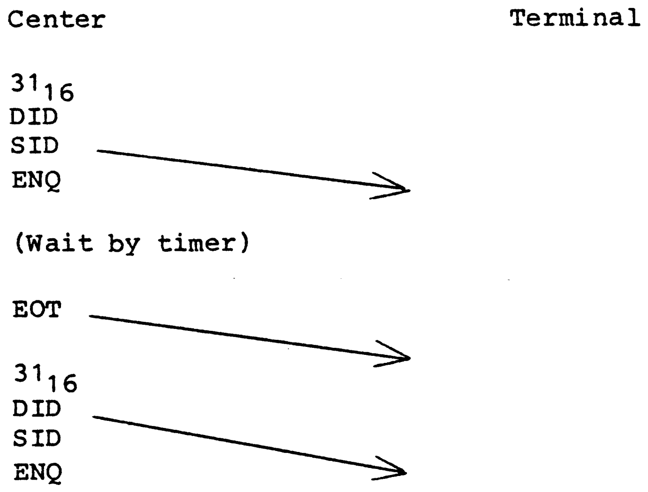


2. WITH ERRORS

(1) When the slave responds by sending NAK in response to ENQ from the master.

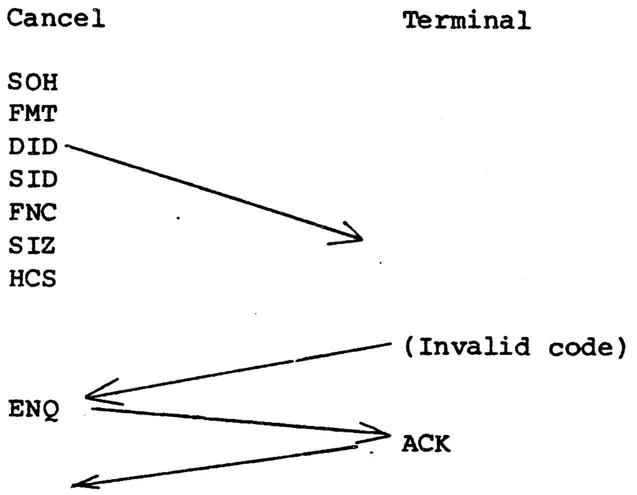


(2) When no response or an invalid response is received from the slave in response to ENQ from the master.

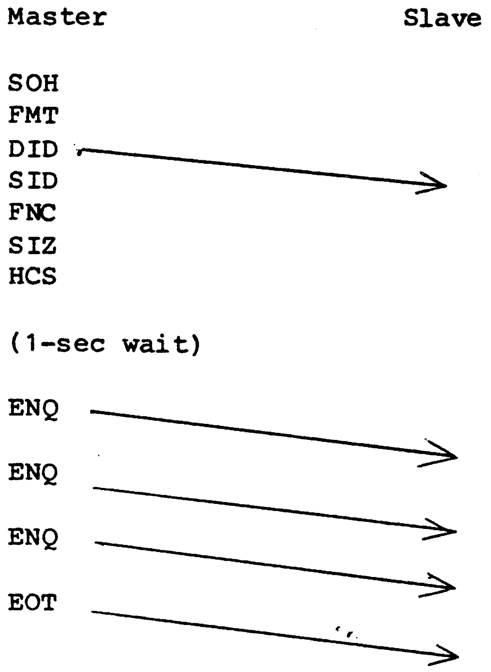


ENQ is transmitted three times.

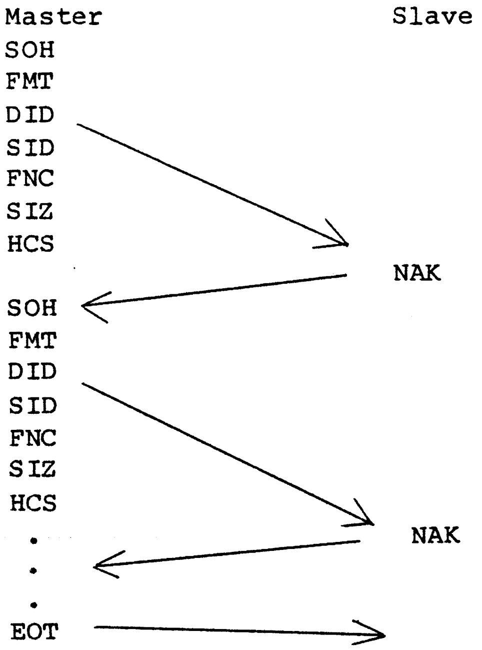
(3) When the master receives an invalid code from the slave in response to transmitting a header to the terminal.



(4) When no response is received from the slave in response transmission of a header from the master.



(5) When NAK is sent from the slave in response to transmission of a header from the master.



If the master receives NAK three or more times, it terminates the data link by transmitting EOT.
If the slave transmits NAK three times in succession, the master will not send the header but will send EOT to terminate the data link.

(6) When the master receives an invalid response code from the slave in response to text transmission, the master handles this as in (3) above.

(7) If no response is received from the slave, the master handles this as in (4) above.

(8) If the slave responds with NAK when the master transmits text, the master handles this as in (5) above (text retransmission).

(9) When the master does not receive a correct response after sending EOT to the slave.
   a. If there is no response, the master waits for 1 second and terminates the data link by transmitting EOT.
   b. If the master receives a code other than EOT from the slave, it terminates the data link by transmitting EOT.

(10) When the slave has not correctly received and responded to EOT sent from the master (when response from the slave is necessary)

a. If the slave has not received EOT and does not respond to the master, the master will wait (3 seconds in mode Ø), terminate the data link and restart the link procedure from the beginning.

b. If the slave receives a code other than EOT, the master assumes that the terminal has made no response. If the slave does not receive EOT after the master has sent EOT the specified number of times (3 times), the center returns to the link start procedure.

(11) If the slave does not transmit a header after the master transmits EOT, the master requests the slave to transmit the header by retransmitting EOT after waiting for a given time. If the slave does not transmit the header even after EOT has been transmitted the specified number of times, the master assumes that an error occurred and terminates the data link.

Function character code table

	Ø	1
Ø		DLE
1	SOH	
2	STX	
3	ETX	
4	EOT	
5	ENQ	NAK
6	ACK	
7		
8		
9		
A		
B		
C		
D		
E		
F		

```
      *** 6301 CROSS MACROASSEMBLER   VER1.0 ***                          10/26/82     13:48:39

ERR   SEQ    LOC  OBJECT       PROGRAM  EPSP        --- SERIAL (EPSP) EXAMPLE ---

    00001                                    NAM    EPSP
    00002                             *
    00003                             *  READ CHARACTER FROM KEYBOARD AND SEND CHARACTERS TO ANOTHER HC-20
    00004                             *   BY EPSP.
    00005                             *  AND AT ANOTHER HC-20.
    00006                             *   RECEIVED CHARACTER FROM EPSP, DISPLAY CHARACTERS ON THE VIRTUAL
    00007                             *   SCREEN.
    00008                             *
    00009                             *  FILE NAME   'EX$1' BY  K.A
    00010                                    TTL    --- SERIAL (EPSP) EXAMPLE ---
    00011                             *
    00012                                    OPT    LOAD
    00013                                    OPT    PAGE=55
    00014                             *
    00015                             *
    00016                             * SERIAL COMMUNICATION
    00017                             *
    00018                             * CONDITION SWITCH
    00019              0000  A        SRSL   EQU    0          * SERIAL SELECT PROCEDURE
    00020                             *
    00021                             * COMMON DEFINITION
    00022                             *
    00023                             * MPU 6301 I/O PORT
    00024              0002  A        PORT1  EQU    $02        * I/O PORT 1
    00025              0003  A        PORT2  EQU    $03        * I/O PORT 2
    00026              0006  A        PORT3  EQU    $06        * I/O PORT 3
    00027                             *
    00028                             * OTHER REGISTERS
    00029              0011  A        TRCSR  EQU    $11        * TRANSMIT/RECEIVE CONTROL & STATUS REGISTER
    00030              0010  A        RMCR   EQU    $10        * RATE & MODE CONTROL REGISTER (RMCR)
    00031              0013  A        STDR   EQU    $13        * SERIAL TRANSMIT DATA REGISTER
    00032              0012  A        SRDR   EQU    $12        * SERIAL DATA RECEIVE DATA REGISTER
    00033                             *
    00034                             * SUBROUTINE ENTRY POINT
    00035              FF4F  A        DSPSCR EQU    $FF4F      * DISPLAY ONE CHARACTER TO VIRTUAL SCREEN
    00036              FF5E  A        SCRFNC EQU    $FF5E      * VIRTUAL SCREEN FUNCTION
    00037              FF6D  A        SERIN  EQU    $FF6D      * SERIAL RECEIVE
    00038              FF70  A        SEROUT EQU    $FF70      * SERIAL TRANSMITTE
    00039              FF73  A        SERONF EQU    $FF73      * SERIAL DRIVER ON/OFF
    00040              FF9A  A        KEYIN  EQU    $FF9A      * GET ONE CHARACTER FROM KEYBOARD BUFFER
    00041              FF9D  A        KEYSTS EQU    $FF9D      * GET NUMBER OF CHARACTERS IN THE KEY BUFFER
    00042                             *
    00043A 0050                              ORG    $50
    00044                             * GENERAL REGISTERS USED BY I/O ROUTINE
    00045              0050  A        R0     EQU    *          * 2 BYTES REGISTER  (R0H,R0L)
    00046A 0050        0001  A        R0H    RMB    1
    00047A 0051        0001  A        R0L    RMB    1
    00048              0052  A        R1     EQU    *          * 2 BYTES REGISTER  (R1H,R1L)
    00049A 0052        0001  A        R1H    RMB    1
    00050A 0053        0001  A        R1L    RMB    1
    00051              0054  A        R2     EQU    *          * 2 BYTES REGISTER  (R2H,R2L)
    00052A 0054        0001  A        R2H    RMB    1
    00053A 0055        0001  A        R2L    RMB    1
    00054              0056  A        R3     EQU    *          * 2 BYTES REGISTER  (R3H,R3L)
    00055A 0056        0001  A        R3H    RMB    1
```

```
      00056A 0057     0001  A        R3L    RMB     1
      00057           0058  A        R4     EQU     *           * 2 BYTES REGISTER  (R4H,R4L)
      00058A 0058     0001  A        R4H    RMB     1
      00059A 0059     0001  A        R4L    RMB     1
      00060           005A  A        R5     EQU     *           * 2 BYTES REGISTER  (R5H,R5L)
      00061A 005A     0001  A        R5H    RMB     1
      00062A 005B     0001  A        R5L    RMB     1
      00063A 007A                           ORG     $7A
      00064A 007A     0001  A        SRSTS  RMB     1           * SERIAL STATUS
      00065                          *                          * BIT 0,1: RS232 MODE(00:STOP 01:INTERRUPT READ
      00066                          *                                                 02:READ ONE CHARACTER)
      00067                          *                          * BIT 2: EXECUTE/PAUSE (0:ON EXECUTE  1:PAUSE)
      00068                          *                          * BIT 3: RS232 DRIVER (0:OFF  1:DRIVER ON
      00069                          *                          * BIT 4: SERIAL DRIVER (0:OFF  1:DRIVER O
      00070                          *                          * BIT 5,6,7: CPU SERIAL RECEIVE INTERRUPT MODE
      00071A 007B     0001  A        RUNMOD RMB     1           * RUN MODE   ($80:BASIC   $00:SYSTEM)
      00072A 007C     0001  A        SIOSTS RMB     1           * SLAVE I/O STATUS   (EACH BIT  0:OFF,   1:ON)
      00073                          *                          * BIT 0: PRINTER
      00074                          *                          * BIT 1: EXTERNAL CASSETTE
      00075                          *                          * BIT 2: INTERNAL CASSETTE
      00076                          *                          * BIT 3: RS232C ON (READ)
      00077                          *                          * BIT 4: SPEAKER ON
      00078                          *                          * BIT 5: PROM CASSETTE
      00079                          *                          * BIT 6: BAR CODE READER
      00080                          *                          * BIT 7: BREAK SLAVE CPU (0:ON EXECUTE
      00081                          *                          *                             1:BROKEN BY INTERRUPT
      00082A 007D     0001  A        MIOSTS RMB     1           * MAIN I/O STATUS  EACH BIT (0:OFF 1:ON)
      00083                          *                          * BIT 0: LCD ON READ/WRITE CHARACTERS
      00084                          *                          * BIT 1: ON CONTINUE SEND COMMAND TO SLAVE CPU
      00085                          *                          * BIT 2: ON CONTINUE TO SEND SERIAL LINE (1:ON)
      00086                          *                          * BIT 3: ON CLOCK INTERRUPT (1:ON)
      00087                          *                          * BIT 4: (POWER FAIL)
      00088                          *                          * BIT 5: (OFF POWER SWITCH)
      00089                          *                          * BIT 6: ON PAUSE KEY
      00090                          *                          * BIT 7: ON BREAK KEY
      00091                          *
      00092                          *
      00093                          * RAM COMMON WORK AREA
      00094A 01C4                           ORG     $1C4
      00095                          *
      00096                          * WORK FOR SERIAL COMMUNICATION
      00097           01C4  A        SRWKTP EQU     *           * SERIAL WORK TOP ADDRESS
      00098A 01C4     0001  A        SRFMT  RMB     1           * FORMAT                                      (0)
      00099A 01C5     0001  A        SRDDEV RMB     1           * DESTINATION DEVICE                          (1)
      00100A 01C6     0001  A        SRSDEV RMB     1           * SOURCE DEVICE                               (2)
      00101A 01C7     0001  A        SRFNC  RMB     1           * FUNCTION                                    (3)
      00102A 01C8     0001  A        SRSIZ  RMB     1           * TEXT SIZE                                   (4)
      00103A 01C9     0001  A        SRACKC RMB     1           * RECEIVED ACK CHARACTER                      (5)
      00104A 01CA     0001  A        SRTRCN RMB     1           * SEND TRY COUNT                              (6)
      00105A 01CB     0001  A        SRTIMO RMB     1           * FOR RECEIVE CHARACTER TIME OVER LIMIT       (7)
      00106A 01CC     0001  A        SRETMO RMB     1           * FOR RECEIVE BLOCK TIME OVER LIMIT           (8)
      00107A 01CD     0001  A        SRATMO RMB     1           * FOR RECEIVE ACK TIME OVER LIMIT             (9)
      00108A 01CE     0001  A        SRMODE RMB     1           * SERIAL MASTER/SLAVE MODE (0:MASTER)         (10)
      00109                          *                         *                           (NOT:SLAVE)
      00110A 01CF     0001  A        SRETDL RMB     1           * AFTER SEND 'EOT', IDLING TIME (1 = 1 MILI SEC
     00111A 01D0     0001  A        SRBLCN RMB    1         * FOR RECEIVE BLOCK, BLOCK COUNTER
     00112A 01D1     0001  A        SRERMD RMB    1         * FOR RECEIVE BLOCK, ERROR MODE
     00113A 01D2     0001  A        SRRVFL RMB    1         * FOR RECEIVE BLOCK, RECEIVED CHARACTER FLAG
     00114A 01D3     0002  A        SREIX  RMB    2         * FOR RECEIVE BLOCK, DATA STORED ADDRESS
     00115           01D5  A        SRWKBT EQU    *         * WORK BOTTOM
     00116                          *
     00117                          *
     00118                          * SERIAL COMMUNICATION ROUTINE
     00119                          *
     00120           0001  A        SOH    EQU    $01       * SOH
     00121           0002  A        STX    EQU    $02       * STX
     00122           0003  A        ETX    EQU    $03       * ETX
     00123           0004  A        EOT    EQU    $04       * EOT
     00124           0005  A        ENQ    EQU    $05       * ENQ
     00125           0006  A        ACK    EQU    $06       * ACK
     00126           0015  A        NAK    EQU    $15       * NAK
     00127           0010  A        DLE    EQU    $10       * DLE
     00128           003B  A        WAK    EQU    $3B       * WAK (;)
     00129           0031  A        DEVCRT EQU    $31       * DEVICE NO. (CRT)
     00130                          *
     00131A 1000                           ORG    $1000
     00132                          *
     00133                          *
     00134                          * OUT TO SERIAL ROUTINE
     00135                          *
     00136                          *  INITIALIZATION OF SERIAL
     00137                          *  1. CLEAR FMT, DID, SID, FNC, SIZ WORK
     00138                          *  2. SET 'NAK' CODE TO ACK CHARACTER AREA
     00139                          *  3. SET RETRY COUNT (INITIAL 5)
     00140                          *  4. SET TIME OVER COUNT (INITIAL 0.5 SEC)
     00141                          *  5. SET START BLOCK TIME OVER COUNT (INITIAL 10 SEC)
     00142                          *
     00143                          *
     00144                          * RECEIVE ONE CHARACTER FROM SERIAL PORT
     00145                          * ON ENTRY
     00146                          *  PARAMETER NONE
     00147                          * ON EXIT
     00148                          *  (C): I/O ERROR FLAG  0:OK  1:ERROR
     00149                          *  (V): TIME OVER FLAG  0:OK  1:TIME OVER   (TIME OVER = 0.1 SEC)
     00150                          *  (A): RECEIVED CHARACTER (IF (C)=0 AND (Z)=1)
     00151                          * REGISTER PRESERVE  B,X
     00152                          *
     00153A 1000 B6 01CB  A         SRVSGL LDA A  SRTIMO    * SET TIME OVER COUNTER
     00154                          * ENTRY POINT (PARAMETER  (A):TIME OVER LIMIT)
     00155A 1003 37                 SRVSXX PSH B
     00156A 1004 16                        TAB
     00157A 1005 8D 0D 1014         SRVS40 BSR    SRVBYT    * RECEIVE ONE CHARACTER
     00158A 1007 25 09 1012                BCS    SRVS50    * I/O ERROR ?
     00159A 1009 28 07 1012                BVC    SRVS50    * OK ?
     00160A 100B 5D                        TST B            * TIME OVER LIMIT CHECK
     00161A 100C 27 F7 1005                BEQ    SRVS40
     00162A 100E 5A                        DEC B
     00163A 100F 26 F4 1005                BNE    SRVS40
     00164A 1011 0B                        SEV              * TIME OUT
     00165A 1012 33                 SRVS50 PUL B
```

```
00166A 1013 39                         RTS
00167                           *
00168                           *
00169                           * RECEIVE ONE BYTE
00170                           * REGISTER PRESERVE  B,X
00171                           *
00172A 1014 3C                  SRVBYT PSHX
00173A 1015 CE 0BB8   A                LDX    #3000       * 21*1.6*3000 = 100,000
00174A 1018 4F                  SRVS10 CLR A              * (C)<---0, (V)<--- 0              (1 C/S)
00175A 1019 09                         DEX                * NOT RECEIVED, CHECK TIME OVER (1 C/S)
00176A 101A 0B                         SEV                * PRESET (V)                       (1 C/S)
00177A 101B 27 12 102F                 BEQ    SRVS30      *                                  (2 C/S)
00178A 101D 0D                         SEC                * PRESET I/O ERROR FLAG            (1 C/S)
00179A 101E 7B 807D   A                TIM    #$80,MIOSTS *                                  (3 C/S)
00180A 1021 26 0C 102F                 BNE    SRVS30      *                                  (2 C/S)
00181A 1023 7B 0403   A                TIM    #$4,PORT2 *   CONNECTED EXTERNAL SERIAL       (3 C/S)
00182A 1026 26 07 102F                 BNE    SRVS30      *                                  (2 C/S)
00183A 1028 7D 0011   A                TST    TRCSR       * RECEIVED ?                       (3 C/S)
00184A 102B 2A EB 1018                 BPL    SRVS10      *                                  (2 C/S)   21 C/S
00185                           * RECEIVED                 * (A) <--- RECEIVED CHARACTER
00186A 102D 96 12     A                LDA A  SRDR        * (C),(V) <--- 0 BY 'TST' INSTRUCTION
00187A 102F 38                  SRVS30 PULX
00188A 1030 39                         RTS
00189                           *
00190                           * WAIT TO BE SELECTED
00191                           *   RECEIVE SEQUENCE
00192                           *    1. WAIT SERIAL IDLING
00193                           *    2. CHECK 'EOT'
00194                           * PARAMETER
00195                           * ON ENTRY
00196                           *  (A):DESTINATION DEVICE  (FOR SENDING SIDE)
00197                           *  (B):SOURCE DEVICE       (FOR SENDING SIDE)
00198                           *  (X):TIME OVER LIMIT (1=0.1 SEC,  0:NO LIMIT)
00199                           * ON EXIT
00200                           *  (C):I/O ERROR FLAG  (0:NORMAL  1:ERROR)
00201                           *  (A):RETURN CODE  (0:NORMAL) ($B3:TIME OUT ERROR)
00202                           *  (Z):DEPEND ON VALUE OF (A)
00203                           *
00204                           * WORK USE AS REGISTER
00205                           *  R0H:31
00206                           *  R0L:DID
00207                           *  R1H:SID
00208                           *  R1L:ENQ
00209                           *
00210                           *  ENTRY POINT: RECEVED 'EOT', CHECK ENQ PATTERN.
00211A 1031 DD 51     A         SRSLET STD    R0L         * SET ENQ PATTERN
00212A 1033 86 05     A                LDA A  #ENQ
00213A 1035 97 53     A                STA A  R1L
00214A 1037 86 31     A                LDA A  #$31
00215A 1039 97 50     A                STA A  R0H         * SET P1
00216A 103B 20 2E 106B                 BRA    SRSL18
00217                           *
00218                           *  ENTRY POINT: WAIT 'EOT' 'P1', ... 'ENQ' PATTERN
00219A 103D DD 51     A         SRSLCT STD    R0L         * SET ENQ PATTERN
00220A 103F 86 05     A                LDA A  #ENQ
```

```
00221A 1041 97 53      A            STA A  R1L
00222A 1043 86 31      A            LDA A  #$31
00223A 1045 97 50      A            STA A  R0H        * SET P1
00224                        *
00225            1047  A     SRSL10 EQU    *
00226A 1047 71 FB03    A            AIM    #$FF-$4,PORT2 * SELECT EXTERNAL SERIAL
00227            104A  A     SRSL11 EQU    *
00228A 104A 72 0111    A            OIM    #$1,TRCSR * SET WAKE UP FLAG
00229            104D  A     SRSL13 EQU    *
00230A 104D 7B 0111    A            TIM    #$1,TRCSR * IDLE ?
00231A 1050 27 0E 1060              BEQ    SRSL15
00232A 1052 0D                      SEC
00233A 1053 7B 0403    A            TIM    #$4,PORT2
00234A 1056 26 30 1088              BNE    SRSL30     * BROKEN SERIAL ?
00235A 1058 96 11      A            LDA A  TRCSR      * IGNORE RECEIVED CHARACTER
00236A 105A 2A F1 104D              BPL    SRSL13
00237A 105C 96 12      A            LDA A  SRDR
00238A 105E 20 EA 104A              BRA    SRSL11
00239A 1060 BD 1014    A     SRSL15 JSR    SRVBYT     * READ CHARACTER
00240A 1063 25 23 1088              BCS    SRSL30     * I/O ERROR ?
00241A 1065 29 22 1089              BVS    SRSL40
00242                        * RECIEVD  'EOT' ?
00243A 1067 81 04      A            CMP A  #EOT
00244A 1069 26 1E 1089              BNE    SRSL40
00245A 106B 5F               SRSL18 CLR B             * (B):RECEIVED COUNTER  DID:0  SID:1
00246A 106C BD 1014    A     SRSL20 JSR    SRVBYT
00247A 106F 29 18 1089              BVS    SRSL40     * TIME OVER ?
00248A 1071 25 15 1088              BCS    SRSL30     * I/O ERROR ?
00249A 1073 3C                      PSHX
00250A 1074 CE 0050    A            LDX    #R0
00251A 1077 3A                      ABX
00252A 1078 A1 00      A            CMP A  0,X
00253A 107A 38                      PULX
00254A 107B 26 0C 1089              BNE    SRSL40
00255A 107D 5C                      INC B
00256A 107E C1 04      A            CMP B  #4         * RECEIVED  00 'DID' 'SID' 'ENQ' ?
00257A 1080 26 EA 106C              BNE    SRSL20
00258                        * RECEIVED 'ACK' SEQUENCE'
00259A 1082 86 06      A            LDA A  #ACK
00260A 1084 BD 1098    A            JSR    SSRSGL
00261A 1087 4F                      CLR A
00262A 1088 39               SRSL30 RTS
00263                        * TIME OUT ?
00264A 1089 8C 0000    A     SRSL40 CPX    #0         * CHECK TIME OUT ?
00265A 108C 27 B9 1047              BEQ    SRSL10
00266A 108E 09                      DEX
00267A 108F 26 B6 1047              BNE    SRSL10
00268A 1091 86 33      A            LDA A  #$33       * TIME OUT ERROR RETURN
00269A 1093 7F 01C5    A            CLR    SRDDEV
00270A 1096 4D                      TST A
00271A 1097 39                      RTS
00272                        *
00273                        * SEND ONE CHARACTER TO SERIAL PORT
00274                        * ON ENTRY
00275                        *        (A): SEND CHARACTER
```

```
    00276                          * ON EXIT
    00277                          *       (C): I/O BREAK FLAG   0:OK   1:ERROR
    00278                          * REGISTER PRESERVE  ALL
    00279                          *
    00280             1098  A      SSRSGL EQU     *
    00281A 1098 0D                 SSRS10 SEC            * PRESET I/O ERROR FLAG
    00282A 1099 7B 807D  A                TIM     #$80,MIOSTS * BREAK ?
    00283A 109C 26 0D 10AB                BNE     SSRS20
    00284A 109E 7B 0403  A                TIM     #$4,PORT2 * CONNECT TO EXTERNAL SERIAL ?
    00285A 10A1 26 08 10AB                BNE     SSRS20
    00286A 10A3 7B 2011  A                TIM     #$20,TRCSR * SEND READY ?
    00287A 10A6 27 F0 1098                BEQ     SSRS10
    00288A 10A8 97 13    A                STA A   STDR
    00289A 10AA 0C                        CLC            * OK RETURN
    00290A 10AB 39                 SSRS20 RTS
    00291                          * SERIAL INITIALIZE
    00292                          *  ON ENTRY
    00293                          *  (A):MODE  (MASTER:0  SLAVE:NONZERO)
    00294                          *
    00295A 10AC 36                 SRINIT PSH A
    00296A 10AD CC 0011  A                LDD     #SRWKBT-SRWKTP * CLEAR WORK (A):PATTERN, (B):COUNT
    00297A 10B0 CE 01C4  A                LDX     #SRWKTP
    00298A 10B3 3C                        PSHX
    00299A 10B4 A7 00    A         CLRB   STA A   0,X
    00300A 10B6 08                        INX
    00301A 10B7 5A                        DEC B
    00302A 10B8 26 FA 10B4                BNE     CLRB
    00303                          *
    00304A 10BA 38                        PULX
    00305A 10BB 32                        PUL A
    00306A 10BC A7 0A    A                STA A   SRMODE-SRWKTP,X * SET MASTER/SLAVE MODE
    00307A 10BE 62 0306  A                OIM     #3,SRTRCN-SRWKTP,X * SEND RETRY COUNT = 3
    00308A 10C1 CC 0A64  A                LDD     #10*256+100
    00309A 10C4 ED 07    A                STD     SRTIMO-SRWKTP,X * TIME OVER LIMIT = 1 SEC
    00310                          *                             * RECEIVE BLOCK TIME OVER LIMIT=10
    00311A 10C6 A7 09    A                STA A   SRATMO-SRWKTP,X * RECEIVE ACK TIME OVER = 1 SEC
    00312A 10C8 6C 08    A                INC     SRETDL-SRWKTP,X * AFTER 'EOT', IDLING TIME
    00313                          *
    00314A 10CA 39                        RTS
    00315                          *
    00316                          *
    00317                          * RECEIVE FROM SERIAL (FOR SLAVE DEVICE)
    00318                          *  ON ENTRY
    00319                          *   (X):RECEIVED DATA STORED ADDRESS
    00320                          *  ON EXIT
    00321                          *   (A):RETURN CODE   0:OK  $80:TIME OVER   $82:RECEIVE ERROR
    00322                          *                    $88: RECEIVED 'EOT'
    00323                          *   (B): 0:RECEIVED WITH HEADER   1:RECEIVED WITHOUT HEADER
    00324                          *        (EFFECTIVE (A)=0)
    00325                          *   (Z):DEPEND ON VALUE OF (A)
    00326                          *
    00327                          *  WORK USE AS REGISTER
    00328                          *   R0H:TOP CHARACTER OF BLOCK ($1 OR $2)
    00329                          *   R1:ADDRESS OF STORED DATA
    00330                          *   R2L:BLOCK LENGTH
```

```
     00331                              *    R3H:RETRY COUNT
     00332                              *    R3L:TIME OVER COUNTER
     00333                              *    R4H:TOP CHARACTER OF BLOCK
     00334                              *    R4L:OMITTED HEADER FLAG (0:NOT  1:OMITTED)
     00335                              *
     00336                              *
     00337                              *
     00338                              *
     00339                              * ERROR RETURN ROUTINE
     00340                              *    SET ERROR CODE TO (A), CLEAR (C)
     00341                              *
     00342A 10CB 86 B0     A            SRERB0 LDA A  #$B0       * ERROR $B0 (TIME OVER)
     00343A 10CD 20 0A 10D9                    BRA    SRER10
     00344                              *
     00345A 10CF 86 B1     A            SRERB1 LDA A  #$B1       * ERROR $B1  (
     00346A 10D1 20 06 10D9                    BRA    SRER10
     00347A 10D3 86 B8     A            SRERB8 LDA A  #$B8       * ERROR $B1  (RECEIVED 'EOT')
     00348A 10D5 20 02 10D9                    BRA    SRER10
     00349                              *
     00350A 10D7 86 B3     A            SRERB3 LDA A  #$B3       * ERROR B3 (DRIVER OFF)
     00351A 10D9 7F 01C5   A            SRER10 CLR    SRDDEV     * CLEAR DID (FOR START FROM ENQ PROCESS)  (C)
     00352          10DC   A            SRER20 EQU    *
     00353A 10DC 7E 117F   A                   JMP    SRRB90
     00354                              *
     00355                              *
     00356                              * EPSP  RECEIVE (SLAVE DEVICE) SUBROUTINE
     00357                              * RECEIVE FROM SERIAL
     00358                              *   ON ENTRY
     00359                              *     (X):RECEIVED DATA STORED ADDRESS
     00360                              *
     00361A 10DF DF 52     A            SERRCV STX    R1
     00362A 10E1 7B 107A   A                   TIM    #$10,SRSTS * DRIVER ON ?
     00363A 10E4 27 F1 10D7                    BEQ    SRERB3
     00364                              *
     00365                              *   SELECT SERIAL (DETATCH SLAVE)
     00366A 10E6 96 11     A                   LDA A  TRCSR      * SAVE TRCSR FOR FECOVER RS232
     00367A 10E8 97 5B     A                   STA A  R5L
     00368                              *
     00369          10EA   A            SERINS EQU    *
     00370A 10EA 0F                             SEI               *
     00371A 10EB 71 EF11   A                   AIM    #$FF-$10,TRCSR * SERIAL INTERRUPT DISABLE
     00372A 10EE 71 FB03   A                   AIM    #$FF-$4,PORT2
     00373          10F1   A            INSR05 EQU    *
     00374A 10F1 4F                            CLR A
     00375A 10F2 4C                            INC A
     00376A 10F3 97 59     A            SRRB10 STA A  R4L        * OMITTED HEADER BLOCK (INITIAL)
     00377A 10F5 B6 01CA   A            SRRB20 LDA A  SRTRCN     * SET RETRY COUNT
     00378A 10F8 97 56     A                   STA A  R3H
     00379                              * RECEIVE FIRST CHARACTER
     00380A 10FA DE 52     A            SRRB30 LDX    R1         * (X): STORED DATA ADDRESS
     00381A 10FC B6 01CC   A                   LDA A  SRETMO     * SET TIME OVER FOR WAITING BLOCK
     00382A 10FF BD 1003   A                   JSR    SRVSXX
     00383A 1102 25 D8 10DC                    BCS    SRER20
     00384A 1104 29 C5 10CB                    BVS    SRERB0     * TIME OVER ERROR ?
     00385A 1106 C6 04     A                   LDA B  #4         * (B): BLOCK SIZE (PRESET FOR HEADER BLOCK)
```

```
00386A 1108 81 01     A              CMP A  #SOH
00387A 110A 27 2E 113A               BEQ    SRRB50
00388A 110C 5C                       INC B              * (B): 5
00389A 110D 3A                       ABX                * (X): DATA STORED ADDRESS
00390A 110E F6 01C8   A              LDA B  SRSIZ       * (B): BLOCK SIZE (FOR DATA BLOCK)
00391A 1111 81 02     A              CMP A  #STX
00392A 1113 27 25 113A               BEQ    SRRB50
00393A 1115 81 05     A              CMP A  #ENQ
00394A 1117 27 19 1132               BEQ    SRCE10
00395A 1119 81 04     A              CMP A  #EOT        * EOT ?
00396A 111B 27 B6 10D3               BEQ    SRERB8
00397                         *
00398                         * OTHER CODES (SKIP CURRENT BLOCK AND SEND 'NAK')
00399A 111D BD 1014   A       SRRB40 JSR    SRVBYT      * RECEIVED ONE CHARACTER ?
00400A 1120 25 5D 117F               BCS    SRRB90
00401A 1122 28 F9 111D               BVC    SRRB40
00402                         * TIME OVER (NOT RECEIVED DATA 0.1 SEC)
00403                         *
00404                         * ERROR 'NAK' SEND
00405A 1124 86 15     A       SRCSER LDA A  #NAK
00406A 1126 B7 01C9   A              STA A  SRACKC      * SET NAK CHARACTER FOR 'ENQ'
00407A 1129 D6 56     A              LDA B  R3H         * RETRY COUNT CHECK
00408A 112B 27 A2 10CF               BEQ    SRERB1
00409A 112D 7A 0056   A              DEC    R3H
00410A 1130 27 9D 10CF               BEQ    SRERB1
00411                         * ENTRY FROM 'ENQ'
00412A 1132 B6 01C9   A       SRCE10 LDA A  SRACKC
00413A 1135 BD 1098   A              JSR    SSRSGL      * SEND NAK
00414A 1138 20 C0 10FA               BRA    SRRB30
00415                         *
00416                         *
00417                         * RECEIVE DATA BLOCK (SOH....  OR  STX....)
00418                         *
00419A 113A 5C                SRRB50 INC B
00420A 113B D7 55     A              STA B  R2L
00421A 113D 97 58     A              STA A  R4H         * R4H:RECIEVD FIRST CHARACTER
00422A 113F 97 50     A              STA A  R0H         * R0H:$1 (SOH) OR $2 (STX)
00423A 1141 16                       TAB                * (B):CHECKSUM
00424                         * RECEIVE DATA STRING LOOP
00425A 1142 BD 1000   A       SRRB70 JSR    SRVSGL
00426A 1145 25 38 117F               BCS    SRRB90
00427A 1147 29 DB 1124               BVS    SRCSER      * TIME OVER ?
00428A 1149 A7 00     A              STA A  0,X
00429A 114B 08                       INX
00430A 114C 1B                       ABA
00431A 114D 16                       TAB
00432A 114E 7A 0055   A              DEC    R2L
00433A 1151 26 EF 1142               BNE    SRRB70
00434                         *
00435A 1153 BD 1000   A       SRRB75 JSR    SRVSGL      * RECEIVE CHECKSUM
00436A 1156 25 27 117F               BCS    SRRB90
00437A 1158 29 CA 1124               BVS    SRCSER
00438A 115A 1B                       ABA
00439A 115B 16                       TAB
00440A 115C 7A 0050   A              DEC    R0H         * IF STX..., RECEIVE 'ETX>
```

```
      00441A 115F 26 F2 1153               BNE    SRRB75
      00442A 1161 5D                       TST B              * CHECKSUM OK ?
      00443A 1162 26 C0 1124               BNE    SRCSER
      00444                         *
      00445A 1164 86 06     A              LDA A  #ACK
      00446A 1166 B7 01C9   A              STA A  SRACKC      * SAVE SEND ACK CODE FOR 'ENQ'
      00447A 1169 BD 1098   A              JSR    SSRSGL
      00448A 116C 25 11 117F               BCS    SRRB90
      00449A 116E DC 58     A              LDD    R4          * R4H<--- FIRST CHARACTER OF BLOCK, (B):MODE
      00450A 1170 88 01     A              EOR A  #SOH
      00451A 1172 26 0A 117E               BNE    SRRB80      * IF 'SOH', RECEIVED HEADER BLOCK  (A=0)
      00452                         *
      00453A 1174 DE 52     A              LDX    R1          * SET COUNTER
      00454A 1176 E6 04     A              LDA B  SRSIZ-SRFMT,X
      00455A 1178 F7 01C8   A              STA B  SRSIZ
      00456A 117B 7E 10F3   A              JMP    SRRB10
      00457                         * COMPLETED TO RECEIVE DATA BLOCK
      00458                         *
      00459A 117E 4F                SRRB80 CLR A
      00460A 117F 0E                SRRB90 CLI
      00461A 1180 DE 52     A              LDX    R1
      00462A 1182 71 FB7D   A              AIM    #$FF-$4,MIOSTS * STATUS, STOP SERIAL COMMUNICATION
      00463                         * RECOVER RS232  (NOT CHANGE C)
      00464A 1185 7B B07D   A              TIM    #$B0,MIOSTS * BROKEN ?
      00465A 1188 27 01 118B               BEQ    SRRB9A      * NOTE. AFTER 'CLI' INSTRUCTION, 'BREAK'
      00466A 118A 0D                       SEC                *  MAY BE CAUSED.
      00467                         *
      00468             118B  A     SRRB9A EQU    *
      00469A 118B 7B 037A   A              TIM    #$3,SRSTS * ON RS232 READ RUNNING ?
      00470A 118E 27 0E 119E               BEQ    SRRB92
      00471                         * WAIT 250 MICRO SEC (FOR SERIAL TERMINAL TO RECEIVE CHARACTER)
      00472A 1190 36                       PSH A
      00473A 1191 86 32     A              LDA A  #50
      00474A 1193 4A                SRRB91 DEC A
      00475A 1194 26 FD 1193               BNE    SRRB91
      00476A 1196 72 0403   A              OIM    #$4,PORT2 * SELECT SERIAL SLAVE CPU
      00477A 1199 96 5B     A              LDA A  R5L         * RECOVER TRCSR
      00478A 119B 97 11     A              STA A  TRCSR
      00479A 119D 32                       PUL A
      00480                         *
      00481A 119E 8A 00     A       SRRB92 ORA A  #$0         * FOR RECOVER (Z) (UNCHANGE C)
      00482A 11A0 39                       RTS
      00483                         *
      00484                         *


      00486                         *
      00487                         * PROGRAM OF SENDING SIDE (MAIN DEVICE)
      00488                         *  GET CHARACTERS FROM KEYBOARD AND SEND BY EPSP.
      00489                         *
      00490                         *
      00491                         *
      00492A 11A1 86 01     A       OPNBIS LDA A  #1          * DRIVER ON
      00493A 11A3 BD FF73   A              JSR    SERONF
      00494A 11A6 CE 12B8  A                 LDX     #SCRPSD    * SET SCREEN PACKET  X:DATA ADDRESS
      00495A 11A9 C6 02    A                 LDA B   #SCRPSE-SCRPSD * (B):NUBER OF DATA
      00496A 11AB A6 00    A          INIT10 LDA A   0,X
      00497A 11AD A7 08    A                 STA A   SCRPK1-SCRPSD,X
      00498A 11AF 08                         INX
      00499A 11B0 5A                         DEC B
      00500A 11B1 26 F8 11AB                 BNE     INIT10
      00501                           *
      00502A 11B3 CE 12C0  A                 LDX     #SCRPK1    * INITIALIZE SCREEN
      00503A 11B6 BD FF5E  A                 JSR     SCRFNC     * SELECT SCREEN DEVICE (DISPLAY CONTROLER)
      00504                           *
      00505A 11B9 BD FF9A  A          REPEAT JSR     KEYIN
      00506A 11BC 25 0C 11CA                 BCS     BRKRTN
      00507A 11BE B7 1231  A                 STA A   BUF
      00508A 11C1 4F                         CLR A
      00509A 11C2 CE 122C  A                 LDX     #SNDPKT    * SERIAL TRANSMITTE
      00510A 11C5 BD FF70  A                 JSR     SEROUT
      00511                           *
      00512A 11C8 20 EF 11B9                 BRA     REPEAT
      00513A 11CA 39                  BRKRTN RTS
      00514                           *
      00515                           *
      00516                           *


      00518                           *
      00519                           *
      00520                           * PROGRAM OF RECEIVING SIDE (SLAVE DEVICE)
      00521                           *  GET CHARACTERS FROM EPSP AND DISPLAY ON THE VIRTUAL SCREEN.
      00522                           *
      00523                           *
      00524                           *
      00525A 11CB 86 01    A          RECSID LDA A   #1         * DRIVER ON
      00526A 11CD BD FF73  A                 JSR     SERONF
      00527A 11D0 86 01    A                 LDA A   #1         * SERIAL MASTER/SLAVE MODE = SLAVE
      00528A 11D2 B7 01CE  A                 STA A   SRMODE
      00529A 11D5 CE 132E  A                 LDX     #SCRPRD    * SET SCREEN PACKET  X:DATA ADDRESS
      00530A 11D8 C6 0E    A                 LDA B   #SCRPRE-SCRPRD * (B):NUBER OF DATA
      00531A 11DA A6 00    A          RECS10 LDA A   0,X
      00532A 11DC A7 92    A                 STA A   SCRPK1-SCRPRD,X
      00533A 11DE 08                         INX
      00534A 11DF 5A                         DEC B
      00535A 11E0 26 F8 11DA                 BNE     RECS10
      00536                           *
      00537A 11E2 CE 12C0  A                 LDX     #SCRPK1    * INITIALIZE SCREEN
      00538A 11E5 BD FF5E  A                 JSR     SCRFNC     * SELECT SCREEN DEVICE
      00539A 11E8 CE 12C8  A                 LDX     #SCRPK2
      00540A 11EB BD FF5E  A                 JSR     SCRFNC     * SET SCREEN SIZE AND BUFFER ADDRESS
      00541A 11EE CE 12CD  A                 LDX     #SCRPK3    *
      00542A 11F1 BD FF5E  A                 JSR     SCRFNC     * SET CURSOR MARGIN
      00543A 11F4 CE 12CF  A                 LDX     #SCRPK4    *
      00544A 11F7 BD FF5E  A                 JSR     SCRFNC     * SET SCROLL STEP
      00545A 11FA CE 12D2  A                 LDX     #SCRPK5    *
      00546A 11FD BD FF5E  A                 JSR     SCRFNC     * SET SCROLL SPEED
```

```
    00547                               * DEVICE : TREAT AS DISPLAY CONTROLLER.
    00548A 1200 CC 3020  A                     LDD     #$3020     * WAIT TO BE EPSP SELECTED
    00549A 1203 CE 0000  A                     LDX     #0
    00550A 1206 BD 103D  A                     JSR     SRSLCT
    00551A 1209 25 BF 11CA              RCVR10 BCS     BRKRTN
    00552                               *
    00553A 120B CE 1233  A              RCVRPT LDX     #RCVPKT    * RECEIVE DATA
    00554A 120E BD 10DF  A                     JSR     SERRCV
    00555A 1211 25 B7 11CA                     BCS     BRKRTN
    00556A 1213 27 0F 1224                     BEQ     RCVR20     * ERROR ?
    00557A 1215 81 B8    A                     CMP A   #$B8       * RECEIVED 'EOT' ?
    00558A 1217 26 F2 120B                     BNE     RCVRPT
    00559A 1219 CC 3020  A                     LDD     #$3020     * WAIT TO BE EPSP SELECTED
    00560A 121C CE 0000  A                     LDX     #0
    00561A 121F BD 1031  A                     JSR     SRSLET
    00562A 1222 20 E5 1209                     BRA     RCVR10
    00563                               *
    00564A 1224 B6 1238  A              RCVR20 LDA A   RCVPKT+5   * DISPLAY RECEIVED CHARACTERS ON THE VIRTUAL
    00565A 1227 BD FF4F  A                     JSR     DSPSCR     *  SCREEN (LCD)
    00566A 122A 20 DF 120B                     BRA     RCVRPT
    00567                               *
    00568                               * PACKET OF SEND DATA STRING
    00569                               *
    00570A 122C      00     A           SNDPKT FCB     $0         * FORMAT
    00571A 122D      30     A                  FCB     $30        * SID (DISPLAY CONTROLLER)
    00572A 122E      20     A                  FCB     $20        * DID (HC-20)
    00573A 122F      92     A                  FCB     $92        * FUNCTION
    00574A 1230      00     A                  FCB     0          * DATA LENGTH
    00575A 1231      00     A           BUF    FCB     0          * DATA
    00576A 1232      00     A                  FCB     0
    00577                               *
    00578                               * PACKET OF RECEIVE DATA STRING
    00579                               *
    00580A 1233      00     A           RCVPKT FCB     $0         * FORMAT
    00581A 1234      30     A                  FCB     $30        * SID (DISPLAY CONTROLER)
    00582A 1235      20     A                  FCB     $20        * DID (HC-20)
    00583A 1236      92     A                  FCB     $92        * FUNCTION
    00584A 1237      00     A                  FCB     0          * DATA LENGTH
    00585A 1238      0080   A                  RMB     128        * DATA
    00586                               *
    00587                               * SCREEN PACKET FOR SENDING SIDE
    00588A 12B8      84     A           SCRPSD FCB     $84        * SCREEN DEVICE SELECT (DISPLAY CONTROLER)
    00589A 12B9      30     A                  FCB     $30
    00590                               *
    00591            12BA   A           SCRPSE EQU     *
    00592                               *
    00593                               *
    00594                               * WORK AREA
    00595A 12BA      0006   A                  RMB     6
    00596A 12C0      84     A           SCRPK1 FCB     $84        * SELECT SCREEN DEVICE
    00597A 12C1      22     A                  FCB     $22
    00598A 12C2      0006   A                  RMB     6
    00599A 12C8      87     A           SCRPK2 FCB     $87        * SET SCREEN SIZE AND BUFFER ADDRESS
    00600A 12C9      13     A                  FCB     19,3
         A 12CA      03     A
```

```
     00601A 12CB    12D4  A              FDB    SCRBUF
     00602                        *
     00603A 12CD    C3    A       SCRPK3 FCB    $C3        * SET CURSOR MARGIN
     00604A 12CE    04    A              FCB    4
     00605                        *
     00606A 12CF    C4    A       SCRPK4 FCB    $C4        * SET SCROLL STEP
     00607A 12D0    0A    A              FCB    10         * X
     00608A 12D1    03    A              FCB    3          * Y
     00609                        *
     00610A 12D2    CB    A       SCRPK5 FCB    $CB        * SET SCROLL SPEED
     00611A 12D3    09    A              FCB    9
     00612                        *
     00613A 12D4    005A  A       SCRBUF RMB    90
     00614                        * SCREEN PACKET FOR RECEIVING SIDE
     00615A 132E    84    A       SCRPRD FCB    $84        * SCREEN DEVICE SELECT (LCD)
     00616A 132F    22    A              FCB    $22
     00617                        *
     00618A 1330    87    A              FCB    $87        * SET SCREEN SIZE AND BUFFER ADDRESS
     00619A 1331    13    A              FCB    19,3
          A 1332    03    A
     00620A 1333    12D4  A              FDB    SCRBUF
     00621                        *
     00622A 1335    C3    A              FCB    $C3        * SET CURSOR MARGIN
     00623A 1336    04    A              FCB    4
     00624                        *
     00625A 1337    C4    A              FCB    $C4        * SET SCROLL STEP
     00626A 1338    0A    A              FCB    10         * X
     00627A 1339    03    A              FCB    3          * Y
     00628                        *
     00629A 133A    CB    A              FCB    $CB        * SET SCROLL SPEED
     00630A 133B    09    A              FCB    9
     00631                        *
     00632          133C  A       SCRPRE EQU    *
     00633                        *
     00634                        *
     00635          0000  A              END
***** TOTAL ERRORS      0
```

CHAPTER 5

Page

5.1 GENERAL 5-1
5.2 DATA TRANSMISSION METHOD 5-1
5.3 DATA RECEPTION METHOD 5-1
5.4 DATA COMMUNICATION 5-2
5.5 NOTES ON I/O OPEN CONDITION 5-3
5.6 BIT RATE SETTING 5-3
5.7 RTS OPERATION AND CARRIER DETECTION 5-4
5.8 COMMUNICATIONS USING A MODEM 5-5
5.9 CAUTIONS FOR SERIAL DRIVER ON/OFF 5-7
5.10 ANOTHER METHOD OF MANAGING CONTROL LINES 5-9
5.11 RS-232C SUBROUTINES 5-11
5.12 RS-232C WORK AREAS 5-15

APPENDIX Sample Lists...................................... 5-17

5.1 General

The RS-232C port performs communication by the start-stop synchronization method (refer to the description of serial communication in Chapter 4). Generation of the TXD binary signal and read of the RXD binary signal are performed by software. The master MCU transmits data (TXD) and the slave MCU receives data (RXD). The slave MCU receives 1 character of data which it sends to the master MCU via the SCI. The master MCU then uses an SCI interrupt to store this data in the receive buffer (Fig. 5-1).

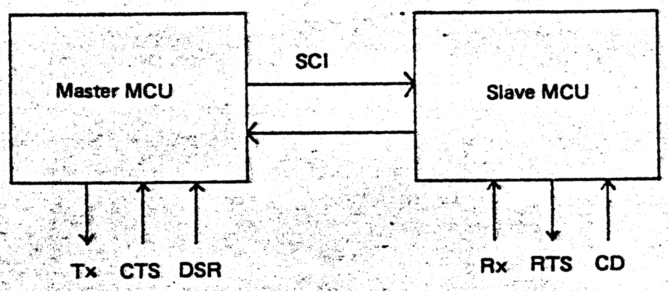


Fig. 5-1 Assignment of RS-232C Functions

5.2 Data Transmission Method

TXD is controlled by port P21 of the master MCU. When a value is set in the OCR and the OCF is set to 1, the value of the OLVL (bit 0 of TCSR) is output from P21 (Fig. 5-2).

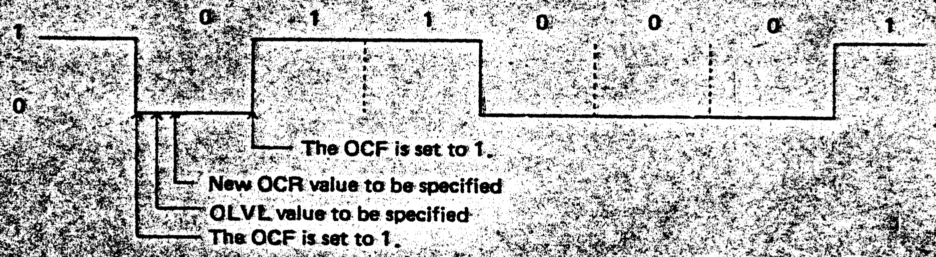


Fig. 5-2 Timing of TXD Transmission

5.3 Data Reception Method

Receive data is input to port P20 of the slave MCU. Input of a start bit in P20 is monitored.

The value of FRC when it takes the value specified by IEDG (bit 1 of TCSR) is set in ICR and this is used to measure the timing of the start bit. Based on this, the calculated center of each pulse is sampled to obtain one character of data (Fig. 5-3).

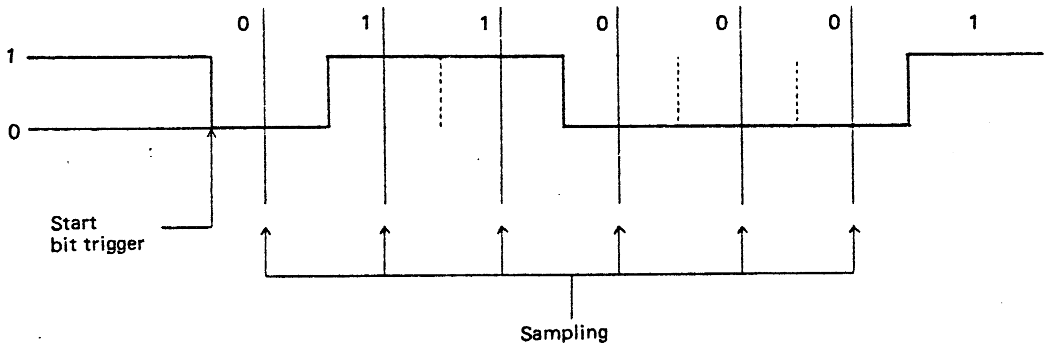


Fig. 5-3 Sampling of Receive Data

One character of data is then transmitted to the master MCU via the SCI (Fig. 5-4).
The master MCU enables receive interrupt by the SCI. The SCI receive interrupt routine stores the receive data in the receive buffer. When the buffer becomes full, an error flag is set and data received subsequent to this will be discarded. The slave MCU cancels input of data through the RS-232C port when a command is sent to it from the master MCU.

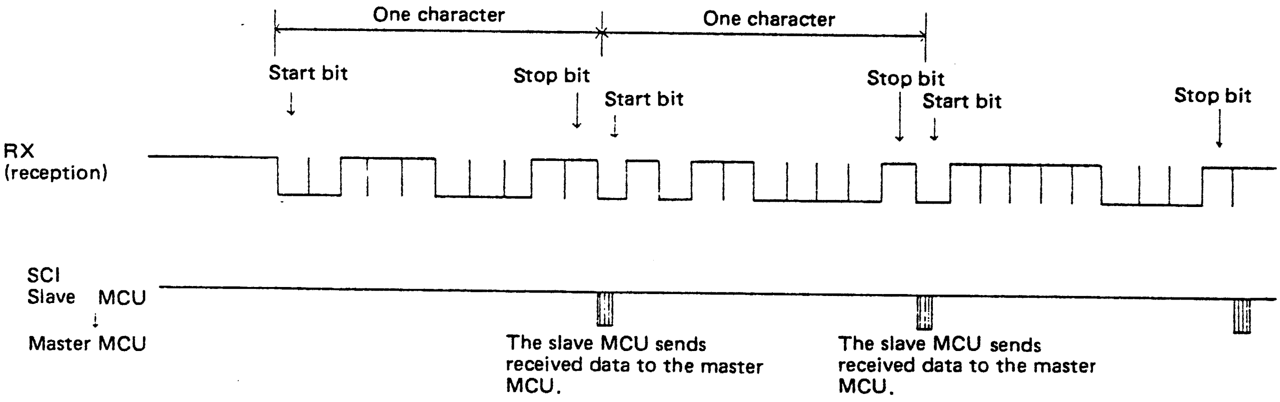


Fig. 5-4 Timing of Data Reception

5.4 Data Communication
Data communication via the RS-232C port is performed by the following procedures.

(1) Setting Parameters
Values for bit rate, word length, parity bit, stop bit length, CD, RTS, DSR detection, are specified by subroutine RSMST. This subroutine specifies the values for constants used in data communication in the I/O work area.

(2) Driver ON

Subroutine RSONOF turns the RS-232C driver ON. When the driver is turned ON, both RTS and TXD go low (RTS is turned OFF and TXD becomes logic 1). A 10-bit preamble (logic 1) is then output. DTR is directly connected to the driver power and therefore goes high (ON) when the driver is turned ON.

(3) Receive buffer open

The receive buffer in the master MCU is opened by subroutine RSOPEN. Once the receive buffer has been opened, the slave MCU begins sending data. The RTS value is set to the value specified procedure (1) above.

(4) Input of one character

Data is fetched from the receive buffer using subroutine RSGET. The data received by the slave MCU is stored in the receive buffer during SCI interrupt processing.

(5) Output of one character

Subroutine RSPUT outputs one character of data. Note that no buffer is used when outputting data.

(6) Termination of data reception

Subroutine RSCLOS terminates RS-232C data reception.

(7) Driver OFF

RSONOF is used to turn the RS-232C driver OFF.

5.5 Notes on I/O Open Condition

The main MCU enable SCI interrupt during RS-232C reception. When the SCI port is accessed directly, the SCI interrupt must be disabled. When the slave MCU receives new data from the SCI port, it cancels data reception from the RS-232C port. The master MCU uses subroutine SNSCOM to send a command to the slave MCU during RS-232C reception and calls subroutine CHKRS (resumption of the interrupted RS-232C data reception) upon completion of transmission of the command.

5.6 Bit Rate Setting

Subroutine RSMST is used to set bit rates for RS-232C transmission (110, 150, 300, 600, 1200, 2400, 4800 and 9600 BPS). To set a transmission speed other than one of those listed above, RSMST must be called and the desired bit rate set directly in variable RSBAUD (01AF, 01B0). This 2-byte variable indicates the number of MCU clock pulses and is set at 1000_{16} for a bit rate of 150 BPS. A bit rate of 75 BPS is therefore obtained by setting 2000_{16} in variable RSBAUD. Note that this value is used directly by the transmission subroutine so the bit rate will change as soon as the value of RSBAUD is altered.

5.7 RTS Operation and Carrier Detection

When using a half-duplex MODEM, the RTS output must be changed and the carrier ON/OFF must be detected. Both RTS and the carrier ports are connected to the slave MCU. RTS control and CD detection are performed by the procedures described below.

(1) RTS Control

Method 1: Subroutine RSOPEN

RTS is set when reception is opened by subroutine RSOPEN. Reception is temporarily closed (subroutine RSCLOS) and the appropriate parameters are set by subroutine RSMST (the previously set parameters remain effective if this is not performed). Reception is then reopened by subroutine RSOPEN. (Fig. 3-5).

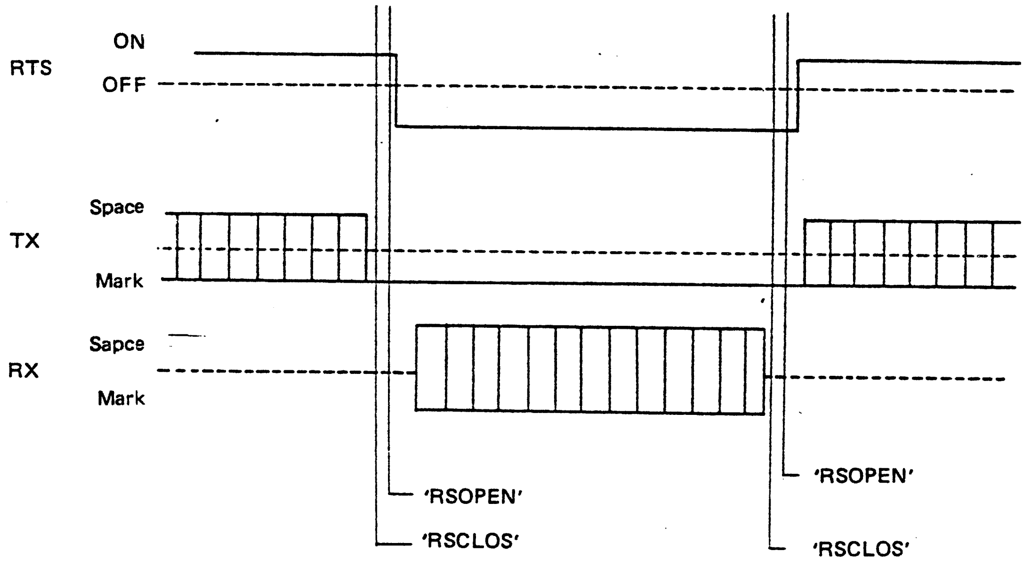


Fig. 5-5 RTS Control (1)

Method 2: Slave MCU command

When performing half-duplex communication, RTS is normally turned ON while data is being transmitted and turned OFF when data is being received. Command 4D, sent to the slave MCU, controls the RTS. This command should be used to turn RTS ON before the start of data transmission. RTS should be turned OFF to open reception (Fig. 5-5).

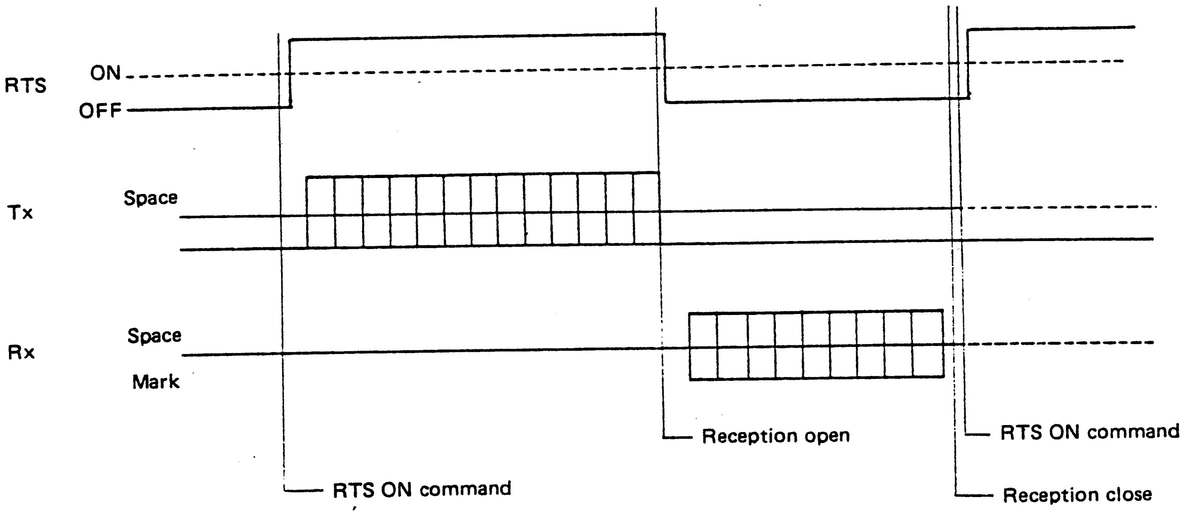


Fig. 5-6 RTS Control (2)

(2) Carrier detection

When the reception is opened, the carrier status is set in port P12 of the master MCU (port P47 of the slave MCU actually detects the carrier status but this data is set in port P12 of the master MCU by software). When the carrier is OFF, P12 of the master MCU is set to 1. When the carrier is ON, P12 is set to 0. Note that after reception has been opened, if carrier OFF status has been detected, carrier ON will cause data reception to start but P12 will not become 0.

The system waits for carrier ON by the following two methods.

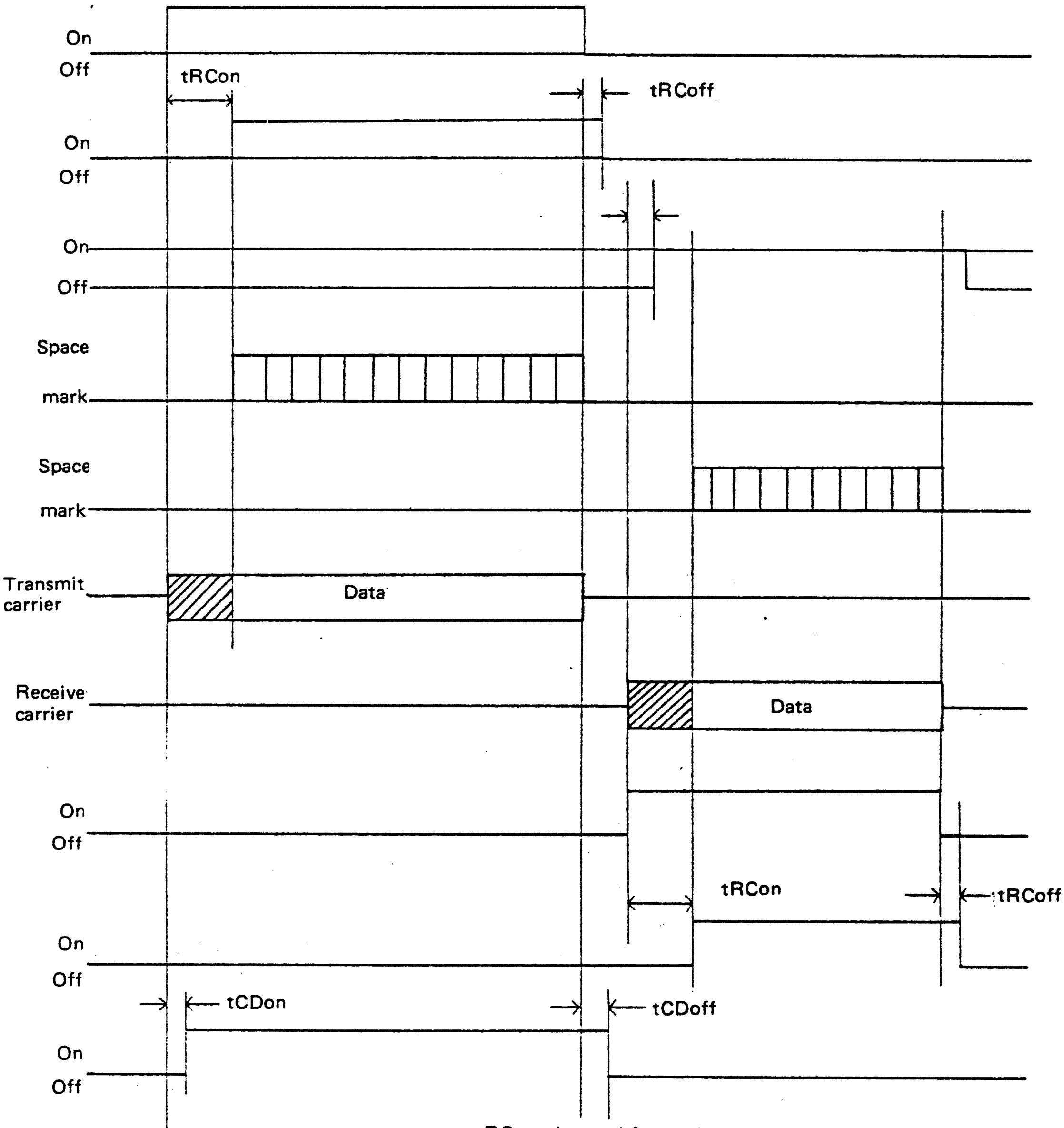
Method 1: If P12 is 1 when the reception is opened, reception is closed and then reopened. This is repeated until carrier ON is detected.

Method 2: Command 80, which sets the value of the slave MCU port in port P12 of the master MCU, is executed for the slave MCU until the carrier is set ON (P12 is set to 0). Reception is then opened.

5.8 Communications Using a MODEM

When using a MODEM, in addition to the data lines for transmission and reception, the control lines must be operated. Fig. 7 shows the timing for a 1200-BPS, half-duplex MODEM.

When data communication is performed as shown in this figure, RTS control as well as CTS and CD detection must be confirmed.



rRCon: Interval from when RTS is set ON to when CTS is set ON
tRCoff: Interval from when RTS is set OFF to when CTS is set OFF
COon: Interval from when the carrier is received to when the CD is set ON
COoff: Interval from when the carrier is turned OFF to when the CD is set OFF

Fig. 5-7 Timing of 1200-BPS, Half-duplex MODEM

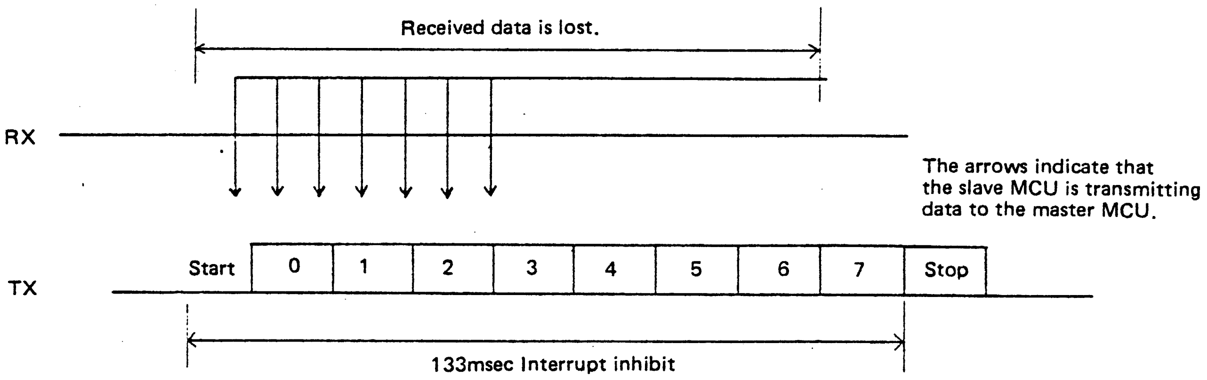
The reception routine provides a mode in which data can be received even if no carrier has been detected. If the carrier OFF state is not of great importance, the reception can be opened in this mode and the carrier ignored.

1200-BPS reverse channels
A 1200-BPS MODEM may use a 75-BPS reverse channel. This is performed by the following two procedures.

(i) 1200-BPS transmission and 75-BPS reception. This is enabled by opening reception (RSOPEN) at 75 BPS and then setting the mode (RSMOD) at 1200 BPS.

(ii) 1200-BPS reception and 75-BPS transmission
Reception is opened at 1200 BPS and the bit rate is set to 75 BPS (2000_{16} in variable RSBAUD).

Since master MCU interrupt is disabled during data transmission, data received at this time will be lost as shown in Fig. 5-8.



The arrows indicate that the slave MCU is transmitting data to the master MCU.

Fig. 5-8 Full-duplex Communication at 1200 and 75 BPS

To protect receive data, the data transmission routine in which interrupt inhibit instruction SEI is omitted must be used. (See end of this chapter.)

5.9 Cautions For Serial Driver ON/OFF

(1) When the Driver is Turned ON

Signal rise may be unstable when the driver is turned ON as shown in Fig. 5-9.

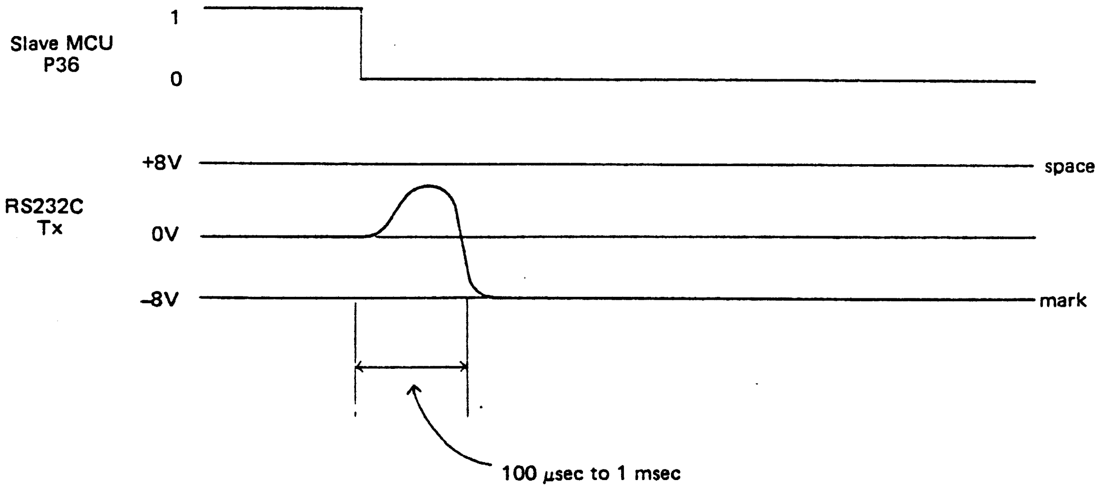


Fig. 5-9 Voltage Change when Driver is Turned ON

In this case, the receiving side may receive incorrect data because it interprets the space state when the driver is turned ON as the start bit.

(2) When the Driver is Turned OFF

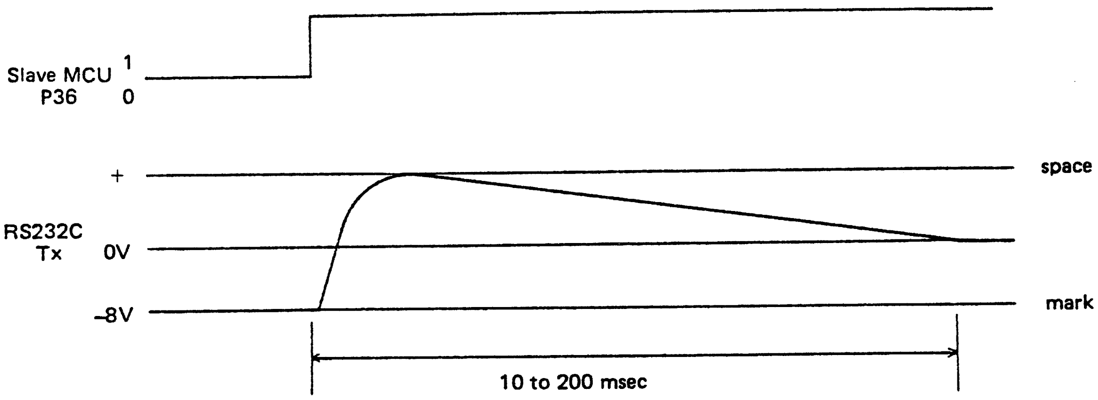


Fig. 5-10 Voltage Change when Driver is Turned OFF

The voltage may change as shown in Fig. 5-10 when the driver is turned OFF. Again, the receiving side may interpret the resulting several tens or hundreds of bits of space states as data, resulting in erroneous data reception.
The driver is turned OFF when the input through the RS-232C port is closed in BASIC. Turn the serial driver ON if you wish to leave the driver on after the RS-232C output is closed. (In terms of software, the serial and RS-232C driver are treated as separate elements. Therefore the driver will only be turned OFF when both drivers are set to OFF from software.)
Press the BREAK key and check the contents of bit 7 of address 7A. When bit 7 is 0, the driver is ON and when it is 1, the driver is OFF. The default value for bit 7 is 0.

5.10 Another Method of Managing Control Lines

Since the RTS and CD control lines are connected to the slave MCU, during RTS control and CD detection there is an idle time (time required for exchanging the master MCU commands) which may cause the user inconvenience.

To avoid this, serial POUT and PIN can be used instead of RTS and CTS as control lines (Fig. 5-11).

POUT corresponds to bit 5 of address 26, and is active low.

Subroutine WRTP26 is used to set data in address 26. PIN corresponds to bit 6 of port 1 and is also active low.

(Example)

Note:
As the floppy disk unit does not use PIN and POUT for serial communication, the RS-232C port can use them as control lines.

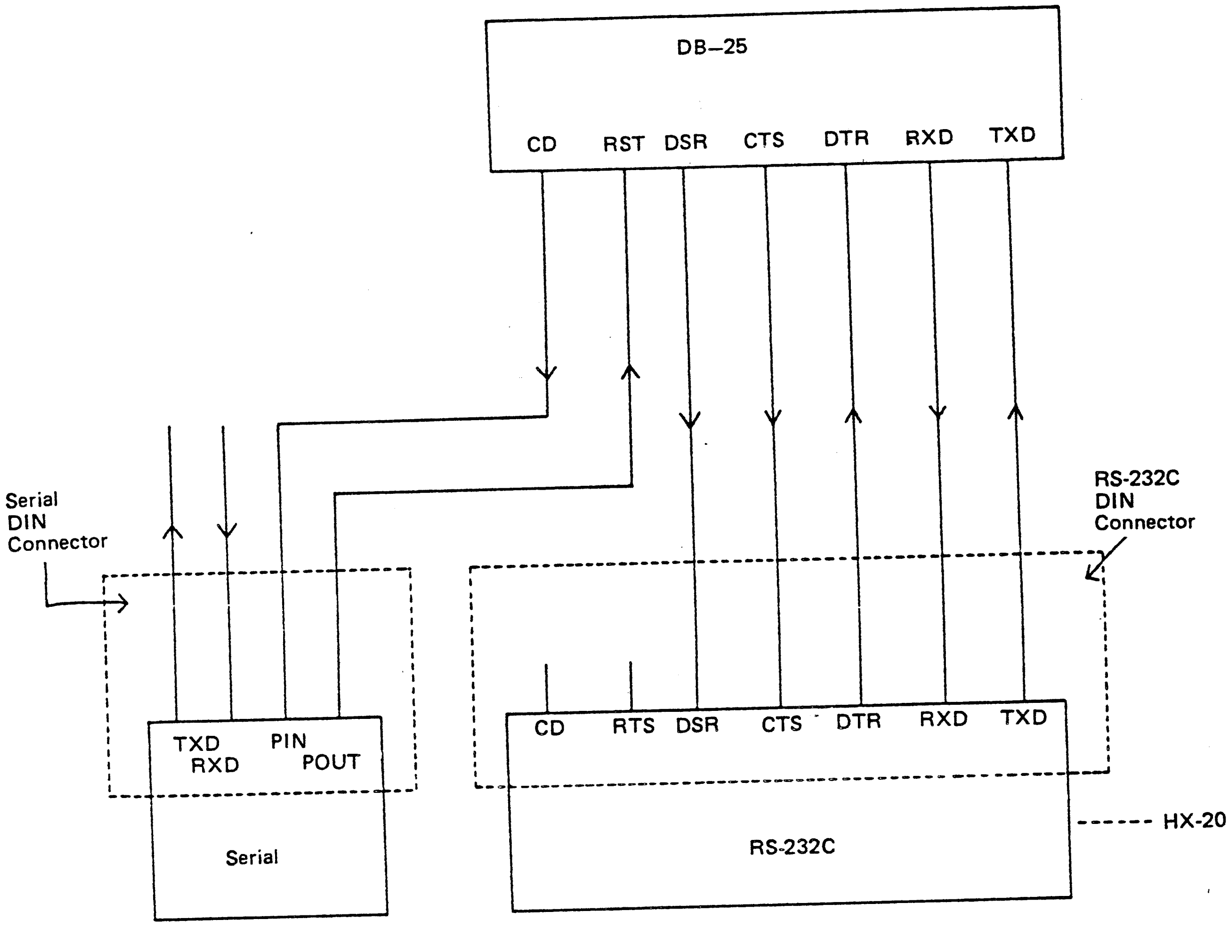


Fig. 5-11 Modification of RS-232C Control Lines

5.11 RS-232C Subroutines

Subroutine name	Entry point	Description
RSMST	FF8A	Specifies the RS-232C mode. Sets values in variables RSBITL, RSMODS, and RSBAUD. Communications with the slave MCU are not performed.
7 0 A 11110Ø01 B 0000Ø000		Parameters: At Entry (A): Mode Bit 0 and 1: Stop bit length (1, 2 or 3) Bit 2: Specifies whether or not carrier detection will be performed. 0: Carrier detection 1: No carrier detection Bit 3: RTS (0: OFF 1: ON) Bit 4: DRS 0: Checks DSR 1: Does not check DSR Bit 5: CTS 0: Checks CTS 1: Does not check CTS Bits 6 and 7: Parity 0: Even 1: Odd 2 or 3: None (B): Bit rate and word length Bits 0 through 3: Word length (5, 6, 7, and 8) Bits 4 through 7: Bit rate 0: 110 BPS 1: 150 BPS 2: 300 BPS 3: 600 BPS 4: 1200 BPS 5: 2400 BPS 6: 4800 BPS 7: 9600 BPS (transmission only) At Return None Registers retained (A), (B), and (X) Subroutines referenced None Variables used None
RSONOF	FF85	Turns ON/OFF the RS-232C driver. When bits 3 and 4 of SRSTS are off, this subroutine turns the driver ON and transmits a 10-bit preamble (data logic 1). If the driver is already ON, the ON procedure will be ignored but no error will occur.

Subroutine name	Entry point	Description
		Parameters: At Entry (A) 0: Turns OFF the driver power. 1: Turns ON the driver power. At Return (A): Error code (C): Abnormal I/O flag (Z): According to the value of (A). Registers retained (B) and (X) Subroutines referenced SNSCOM Variables used None
RSOPEN	FF82	Opens the RS-232C input, initiates fetching data into a buffer, and exchanges commands between the master and slave MCUs. Receive data is stored in the receive buffer via the SCI (interrupt processing). When the RS-232C input is opened, RTS is set at the value specified in subroutine RSMST.
		Parameters: At Entry (A, B): Receive buffer size (X): Starting address of the receive buffer At Return (C): Abnormal I/O flag (A): Return codes 00: RS-232C input has been correctly opened. 01: The driver is OFF. Registers retained None Subroutines referenced SNSCOM, SNSCOW and SNSDAT Variables used None (Example) In this example, a 260-byte monitor buffer is opened as the receive buffer. LDA A #$OD Even parity, CTS/DSR check, RTS high CD check, 1 stop bit LDA B #$27 300 BPS 7-bit word length JSR RSMST LDA A #1 Driver ON JSR RSONOF LDD #260 Buffer size = 260 bytes LDX #CASBUF JSR RSOPEN
RSCLOS	FF7F	Closes input to the RS-232C port and sends a command to the slave MCU to terminate reception. This subroutine does not turn the driver OFF.

Subroutine name	Entry point	Description
		Parameters: At Entry None At Return (C): Abnormal I/O flag (A): Return codes 00: RS-232C has been correctly closed. (Only this code is currently available.) (Z): According to the value of (A) Registers retained (B) and (X) Variables used None Subroutines referenced None
RSGSTS	FF7C	Inputs the value of the status register. When a receive error occurs, this subroutine fetches the error status from the slave MCU and inputs this value to the master MCU. Then, the error status of the slave MCU is cleared. Logic 1 in any bit indicates an error.
		Parameters: At Entry None At Return (C): Abnormal I/O flag (A): Status managed by master MCU (RS-232 transmitting side) Bit 7: 1: Receive buffer overflow (B): Status managed by the slave MCU (RS-232C receiving side) Bit 0: Carrier disconnection (OFF) Bit 1: Parity error Bit 2: Overrun error Bit 5: Receive error Registers retained (X) Subroutines referenced SNSCOM and CHKRS Variables used None
RSGET	FF79	Fetches one character from the receive buffer. The data in the receive buffer is stored in word length + parity bit format. Once a character is fetched, the parity bit is set to 0. This parity bit is not stored in the receive buffer if the format is 8 bits + 1 parity bit.
		Parameters: At Entry None At Return (C): Abnormal I/O flag (A): Received character (B): Return codes

Subroutine name	Entry point	Description
		00: Normal 01: Receive buffer full C0: Parity error C1: Carrier disconnection (OFF) Note: Carrier disconnection (OFF) error occurs not when the carrier falls but when the buffer becomes empty. (Z): According to the value of (B). Registers retained (X) Subroutines referenced None Variables used R0H
RSPUT	FF76	Transmits one character through the the RS-232C port. Note that no transmit buffer is provided.
		Parameters: At Entry (A): Output characters If the number of bits to be transmitted is less than 8 bits, data is right-justified. The remaining bits (including the parity bit) can be any value. At Return (C): Abnormal I/O flag (B): Return codes 00: Normal 01: No data transmitted when DSR is OFF. 02: No data transmitted when CTS is OFF. 03: No data transmitted when both DSR and CTS are OFF. (Z): According to the value of (B) Registers retained (A) and (X) Subroutines referenced None Variables used R0, R1, and R2H
CHKRS	FF16	Sends a command to the slave MCU to resume the interrupted RS-232C input.
		Parameters: At Entry None At Return None Registers retained (A), (B), (X), and condition code (CC) Subroutines referenced RSRSRT Variables used None

5.12 RS-232C Work Areas

Address (from)	 (to)	Variable name	Byte count	Description
1AF	1B0	RSBAUD	2	RS-232C bit rates (clock cycles) 150 BPS : 1000_{16} 300 BPS : 800_{16}
1B1	1B2	RSCRC	2	Polynominal expressions generated for CRC Polynominal expression CRC-CCITT $(1+x^5+x^{12}+x^{16})$ equals 8408_{16} (default value) CRC-16 $(1+x^2+x^{15}+x^{16})$ equals $A001_{16}$ x^{16} is always 1, x^{15} is bit 0, and x^0 is bit 15.
1B3	1B4	RSBCC	2	BCC register for CRC check
1B5	1B5	RSBITL	1	RS-232C word length (stop bit excluded) Word length must be 5, 6, 7, or 8.
1B6	1B6	RSMODS	1	RS-232C mode Bits 0 and 1: Stop bit length (bit 1, bit 0): (0,1) = 1 (1,0) = 2 Bit 2: Carrier (CD) detection 0: Carrier detection 1: No carrier detection Bit 3: RTS 0: RTS OFF (low level) 1: RTS (high level) Bit 4: DSR check 0: Checks if DSR is OFF. 1: Does not check if DSR is OFF. Bit 5: CTS check 0: Checks if CTS is OFF. 1: Does not check if CTS is OFF. Bits 6 and 7: Parity (bit 7, bit 6) = (0,0) : Even parity (0,1) : Odd parity (1,x) : No parity x: don't care
1B7	1B7	RSSTSR	1	RS-232C error status register For all bits of this variable, logic 0 indicates normal operation and logic 1 indicates error. Bit 0: Carrier disconnection (OFF) Bit 1: Parity Bit 2: Overrun Bit 3: Undefined Bit 4: Undefined Bit 5: Receive error Bit 6: Transmit error Bit 7: Receive buffer overflow

Address		Variable name	Byte count	Description
(from)	(to)			
1B8	1B9	RSBFAD	2	Starting address of RS-232C receive buffer
1BA	1BB	RSBFBT	2	Last address of RS-232C receive buffer plus 1
1BC	1BD	RSBFSZ	2	Size of RS-232C receive buffer (in bytes)
1BE	1BE	RSINP	2	Pointer indicating the last data stored in the RS-232C receive buffer (Indicates the next address the buffer in which received data will be stored.)
1C0	1C1	RSOUP	2	Pointer indicating the last data fetched from the RS-232C receive buffer (Indicates the next address to be fetched when data is fetched from the receive buffer.)
1C2	1C3	RSDCNT	2	Number of data in the RS-232C receive buffer (in bytes)

```
ERR   SEQ    LOC  OBJECT       PROGRAM  RS232C      -- RS232C SEND/RECEIVE DATA ROUTINE ---

     00001                                   NAM     RS232C
     00002                                   TTL     -- RS232C SEND/RECEIVE DATA ROUTINE ---
     00003                            *
     00004                            * RS232C SUBROUTINE.
     00005                            *  2 SUBROUTINES
     00006                            *  1. GET RECEIVED CHARACTER FROM RS232C RECEIVED DATA BUFFER (RSGET).
     00007                            *  2. TRANSMITTE ONE CHARACTER TO TXD LINE (RSPUT).
     00008                            *
     00009                            * FILE NAME  'EX$A'     BY K AKAHANE
     00010                                   OPT     PAGE=55
     00011                                   OPT     LOAD
     00012                            *
     00013                            * MCU 6301 I/O PORT
     00014             0002   A       PORT1  EQU     $02          * I/O PORT 1
     00015             0003   A       PORT2  EQU     $03          * I/O PORT 2
     00016             0006   A       PORT3  EQU     $06          * I/O PORT 3
     00017                            * OTHER REGISTERS
     00018             0009   A       FRC    EQU     $09          * FREE RUNNING COUNTER
     00019             000B   A       OCR    EQU     $0B          * OUTPUT COMPARE REGISTER
     00020             0008   A       TCSR   EQU     $08          * TIME CONTROL AND STATUS REGISTER
     00021                            * GENERAL REGISTERS USED BY I/O ROUTINE
     00022A 0050                             ORG     $50
     00023             0050   A       R0     EQU     *            * 2 BYTES REGISTER  (R0H,R0L)
     00024A 0050       0001   A       R0H    RMB     1
     00025A 0051       0001   A       R0L    RMB     1
     00026             0052   A       R1     EQU     *            * 2 BYTES REGISTER  (R1H,R1L)
     00027A 0052       0001   A       R1H    RMB     1
     00028A 0053       0001   A       R1L    RMB     1
     00029             0054   A       R2     EQU     *            * 2 BYTES REGISTER  (R2H,R2L)
     00030A 0054       0001   A       R2H    RMB     1
     00031A 0055       0001   A       R2L    RMB     1
     00032             0056   A       R3     EQU     *            * 2 BYTES REGISTER  (R3H,R3L)
     00033A 0056       0001   A       R3H    RMB     1
     00034A 0057       0001   A       R3L    RMB     1
     00035A 007A                             ORG     $7A
     00036A 007A       0001   A       SRSTS  RMB     1            * SERIAL STATUS
     00037                            *                           * BIT 0,1: RS232 MODE(00:STOP 01:INTERRUPT   AD
     00038                            *                                                02:READ ONE CHARACTER)
     00039                            *                           * BIT 2: EXECUTE/PAUSE (0:ON EXECUTE  1:PAUSE)
     00040                            *                           * BIT 3: RS232 DRIVER (0:OFF  1:DRIVER ON)
     00041                            *                           * BIT 4: SERIAL DRIVER (0:OFF  1:DRIVER ON)
     00042                            *                           * BIT 5,6,7: CPU SERIAL RECEIVE INTERRUPT MODE
     00043                            *                           *        0:EXTERNAL CASSETTE READ
     00044                            *                           *        1:MICRO CASSETTE READ
     00045                            *                           *        2:RS232C READ
     00046                            *                           *        3:READ FROM SERIAL COMMUNICATION
     00047                            *                           *        4:EXTERNAL CASSETTE WRITE
     00048                            *                           *        5:MICRO CASSETTE WRITE
     00049                            *                           *        6,7:UNDEFINED FOR WRITE
     00050A 007B       0001   A       RUNMOD RMB     1            * RUN MODE   ($80:BASIC   $00:SYSTEM)
     00051A 007C       0001   A       SIOSTS RMB     1            * SLAVE I/O STATUS   (EACH BIT  0:OFF,   1:ON)
     00052                            *                           * BIT 0: PRINTER
     00053                            *                           * BIT 1: EXTERNAL CASSETTE
     00054                            *                           * BIT 2: INTERNAL MICRO CASSETTE
     00055                            *                           * BIT 3: RS232C ON (READ)
```

```
    00056                        *                        * BIT 4: SPEAKER ON
    00057                        *                        * BIT 5: ROM CASSETTE
    00058                        *                        * BIT 6: BAR CODE READER
    00059                        *                        * BIT 7: BREAK SLAVE CPU (0:ON EXECUTE
    00060                        *                        *                           1:BROKEN BY INTERRUPT
    00061A 007D    0001   A      MIOSTS RMB    1          * MAIN I/O STATUS  EACH BIT (0:OFF 1:ON)
    00062                        *                        * BIT 0: LCD ON READ/WRITE CHARACTERS
    00063                        *                        * BIT 1: NOW SENDING COMMAND TO SLAVE CPU
    00064                        *                        * BIT 2: NOW SENDING DATA TO SERIAL LINE (1:ON)
    00065                        *                        * BIT 3: ON CLOCK INTERRUPT  (1:ON)
    00066                        *                        * BIT 4: (POWER FAIL)
    00067                        *                        * BIT 5: (OFF POWER SWITCH)
    00068                        *                        * BIT 6: ON PAUSE KEY
    00069                        *                        * BIT 7: ON BREAK KEY
    00070                        * WORK AREA
    00071A 01AF                        ORG     $1AF
    00072                        *
    00073                        *
    00074                        * RS232C WORK AREA
    00075          01AF   A      RSWKTP EQU    *          * RS232C WORK TOP ADDRESS
    00076A 01AF    0002   A      RSBAUD RMB    2          * RS232C BIT RATE (NUMBER OF CLOCK CYCLE)
    00077A 01B1    0002   A      RSCRC  RMB    2          * RS232C GENELATING POLYNOMIAL
    00078A 01B3    0002   A      RSBCC  RMB    2          * RS232C BCC REGISTER
    00079A 01B5    0001   A      RSBITL RMB    1          * RS232C BIT LENGTH (5 6 7 8)
    00080A 01B6    0001   A      RSMODS RMB    1          * RS232C MODE
    00081                        *                        * (0,1:NUMBER OF STOP BITS)
    00082                        *                        * (2: CARRIER DETECT MASK   0:CHECK 1:MASK)
    00083                        *                        * (3: CLEAR TO SEND  0:LOW   1:HIGH)
    00084                        *                        * (4: DSR 0:CHECK 1:NO CHECK)
    00085                        *                        * (5: CTS 0:CHECK 1:NO CHECK)
    00086                        *                        * (6,7:PARITY 00:EVEN 01:ODD 10,11:NONE PARITY)
    00087                        * RS232C BUFFER POINTER
    00088A 01B7    0001   A      RSSTSR RMB    1          * RS232C STATUS REGISTER
    00089                        *                        * (0: CARRIER DETECT  0:NORMAL 1:ERROR)
    00090                        *                        * (1: PARITY          0:NORMAL 1:ERROR)
    00091                        *                        * (2: OVERRUN         0:NORAML 1:ERROR)
    00092                        *                        * (5: READ ERROR      0:NORMAL 1:ERROR)
    00093                        *                        * (6: WRITE ERROR     0:NORMAL 1:ERROR)
    00094                        *                        * (7: BUFFER OVER     0:NORMAL 1:OVERFLOW)
    00095                        *
    00096A 01B8    0002   A      RSBFAD RMB    2          * RS232C READ BUFFER ADDRESS
    00097A 01BA    0002   A      RSBFBT RMB    2          * RS232C READ BUFFER BOTTOM ADDRESS + 1
    00098A 01BC    0002   A      RSBFSZ RMB    2          * RS232 READ BUFFER SIZE (0001 - FFFF)
    00099A 01BE    0002   A      RSINP  RMB    2          * POINTER WHERE NEXT RECEIVED CHARACTER IS STOR
    00100A 01C0    0002   A      RSOUP  RMB    2          * POINTER WHERE NEXT CHARACTER IS LOADED
    00101A 01C2    0002   A      RSDCNT RMB    2          * NUMBER OF DATA IN THE BUFFER
    00102                        *
    00103                        * RS232C: GET ONE CHARACTER FROM RECEIVE BUFFER
    00104                        *   1: GET ONE CHARACTER FROM RS232 RECEIVE BUFFER
    00105                        *   2: IF BIT LENGTH < 8, AND 'PARITY CHECK' MODE, DO PARITY CHECK AND
    00106                        *      SET RETURN CODE
    00107                        * PARAMETER
    00108                        * ON ENTRY
    00109                        *   NONE
    00110                        * ON EXIT
```

```
 00111                              *   (A): CHARACTER (WITHOUT PARITY BIT)
 00112                              *   (B): STATUS   $01:RECEIVED BUFFER IS EMPTY
 00113                              *                 $00:NORMAL
 00114                              *                 MSB= 1:ERROR  0:NORMAL
 00115                              *                 $C0:PARITY ERROR    $C1:CD ERROR (CARRIER DOWN)
 00116                              *   (C): SLAVE STATUS   0:NORMAL  1:ERROR
 00117                              *  SET Z N FLAG DEPEND ON VALUE OF (B) REGISTER
 00118                              * REGISTER  PRESERVE  X
 00119                              *  WORK USE AS REGISTER
 00120                              *    ROH:EFFECTIVE BITS AS DATA (BIT LENGTH=7 THEN $7F,
 00121                              *                                BIT LENGTH=8 THEN $FF)
 00122             01C4  A          RSGET  EQU    *
 00123A 01C4 C6 01     A                   LDA B  #$01       * PRESET 'BUFFER EMPTY' CODE
 00124A 01C6 0D                            SEC               * PRESET ERROR I/O FLAG
 00125A 01C7 7B B07D   A                   TIM    #$80,MIOSTS * ERROR I/O ?
 00126A 01CA 26 42 020E                    BNE    RSIN23
 00127A 01CC 3C                            PSHX
 00128                              *
 00129A 01CD FE 01C2   A                   LDX    RSDCNT     * ARE THERE DATA IN THE BUFFER ?
 00130A 01D0 27 3D 020F                    BEQ    RSIN25     *  (B):1
 00131                              * SET EFFECTIVE BITS TO ROH
 00132A 01D2 FC 01B5   A                   LDD    RSBITL     * (B):RSMODS
 00133A 01D5 7F 0050   A                   CLR    ROH        * (A):BIT LENGTH
 00134A 01D8 0D                     RSIN1A SEC
 00135A 01D9 79 0050   A                   ROL    ROH        * ROH <-- ($7F IF B=7), <--- ($FF IF B=8)
 00136A 01DC 4A                            DEC A
 00137A 01DD 26 F9 01D8                    BNE    RSIN1A
 00138                              *
 00139A 01DF 0F                            SEI               * IF RS232C RECEIVED INTERRUPT IS CAUSED, THE
 00140A 01E0 FE 01C0   A                   LDX    RSOUP      *  POINTER MAY BE DESTROYED.
 00141A 01E3 A6 00     A                   LDA A  0,X        * (A): DATA
 00142A 01E5 08                            INX
 00143A 01E6 BC 01BA   A                   CPX    RSBFBT     * IF THE POINTER SHOWS BOTTOM ADDRESS + 1 OF THE
 00144A 01E9 26 03 01EE                    BNE    RSIN10     * BUFFER, POINTER MUST BE SET TO TOP ADDRESS.
 00145A 01EB FE 01B8   A                   LDX    RSBFAD
 00146A 01EE FF 01C0   A            RSIN10 STX    RSOUP
 00147A 01F1 FE 01C2   A                   LDX    RSDCNT
 00148A 01F4 09                            DEX
 00149A 01F5 FF 01C2   A                   STX    RSDCNT     * DATA COUNTER <---  CURRENT VALUE - 1
 00150A 01F8 0E                            CLI
 00151                              * PARITY ERROR CHECK
 00152A 01F9 58                            ASL B             * PARITY CHECK MODE ?
 00153A 01FA 25 0F 020B                    BCS    RSIN15     * MODE = 'CHECK PARITY' ?
 00154A 01FC 58                            ASL B
 00155A 01FD 16                            TAB               * (B) <--- DATA,  (C)<--- PARITY MODE (0:EVEN)
 00156A 01FE 94 50     A                   AND A  ROH        *  TAKE DATA BITS(IGNORE PARITY RIT)
 00157A 0200 24 02 0204             RSIN11 BCC    RSIN12
 00158A 0202 C8 80     A                   EOR B  #$80
 00159A 0204 58                     RSIN12 ASL B
 00160A 0205 26 F9 0200                    BNE    RSIN11
 00161A 0207 56                            ROR B             * BIT7,BIT6 <--- (C)
 00162A 0208 57                            ASR B
 00163A 0209 20 01 020C                    BRA    RSIN20     * PARITY ERROR = $C0
 00164                              *
 00165A 020B 5F                     RSIN15 CLR B             * NORMAL RETURN
```

```
   00166                          * BUFFER IS EMPTY
   00167            020C  A       RSIN20 EQU    *
   00168A 020C 5D                        TST B            * CLEAR (C), SET (Z)
   00169A 020D 38                        PULX
   00170A 020E 39                 RSIN23 RTS
   00171                          * BUFFER IS EMPTY, IS CARRIER DOWN ?
   00172            020F  A       RSIN25 EQU    *
   00173A 020F 7B 047A  A               TIM    #$4,SRSTS * ON PAUSE ?
   00174A 0212 26 F8 020C               BNE    RSIN20
   00175A 0214 7B 0402  A               TIM    #$4,PORT1 * SFLAG = ON ?
   00176A 0217 27 F3 020C               BEQ    RSIN20
   00177A 0219 C6 C1    A               LDA B  #$C1      * CD ERROR
   00178A 021B 20 EF 020C               BRA    RSIN20
   00179                          *
   00180                          *
   00181                          *
   00182                          *
   00183                          * SEND ONE TRANSMITTED CHARACTER SUBROUTINE
   00184                          * PARAMETER
   00185                          * ON ENTRY
   00186                          *  TRANSMITTED CHARACTER
   00187                          * ON EXIT
   00188                          *  (B): BIT 0 (1:DSR LOW) CHARACTER IS NOT SENT
   00189                          *       BIT 1 (1:CTS LOW) CHARACTER IS NOT SENT
   00190                          *       BIT 2 - 7 (ALWAYS 0)
   00191                          *  (Z)  DEPEND ON VALUE OF (B)
   00192                          *  (C)  0:NORMAL  1:I/O ERROR
   00193                          * REGISTER PRSERVE  A,X
   00194                          *
   00195                          *  WORK USE AS REGISTER
   00196                          *  R0H:PARITY BIT (LSB)
   00197                          *  R0L:FLAG OF 'WITH PARITY BIT'  (0:YES   1:NO)
   00198                          *  R1H:SAVE DATA
   00199                          *  R1L:BIT LENGTH
   00200                          *
   00201                          * NOTE. OCR IS USED. AND OCR IS USED BY KEY POUTINE EITHER.
   00202                          *
   00203            021D  A       RSPUT  EQU    *
   00204A 021D 0D                        SEC              * PRESET I/O ERROR FLAG
   00205A 021E 7B B07D  A               TIM    #$B0,MIOSTS * I/O ERROR ?
   00206A 0221 26 0F 0232               BNE    SNDR04
   00207                          *
   00208                          * CHECK DSR, CTS
   00209A 0223 F6 01B6  A               LDA B  RSMODS    * TAKE MODE (DSR CTS BITS)
   00210A 0226 57                        ASR B            * RSMODS (DSR:BIT 4,CTS:BIT 5) MASK=1
   00211A 0227 57                        ASR B
   00212A 0228 57                        ASR B
   00213A 0229 57                        ASR B            * PORT1 (DSR:BIT 0, CTS:BIT 1) NORMAL='LOW'
   00214A 022A 53                        COM B
   00215A 022B D4 02    A               AND B  PORT1     * CHECK DSR, CTS
   00216A 022D C4 03    A               AND B  #$3
   00217A 022F 27 02 0233               BEQ    SNDR05
   00218A 0231 0C                        CLC              * CTS, DSR LOW (ERROR)
   00219A 0232 39                 SNDR04 RTS
   00220                          *
```

```
00221A 0233 36                    SNDR05 PSH A
00222A 0234 97 52      A                 STA A   R1H
00223A 0236 3C                           PSHX
00224A 0237 CE 01AF    A                 LDX     #RSWKTP    * (X): TOP RAM ADDRESS OF WORK AREA FOR RS232C
00225A 023A 0F                           SEI                * DISABLE INTERRUPT
00226A 023B A6 06      A                 LDA A   RSBITL-RSWKTP,X
00227A 023D 97 53      A                 STA A   R1L
00228A 023F 4F                           CLR A
00229A 0240 E6 07      A                 LDA B   RSMODS-RSWKTP,X * RSMODS  (BIT7:WITH PARITY FLAG
00230A 0242 05                           ASLD                 BIT6:EVEN OR ODD)
00231A 0243 97 51      A                 STA A   ROL        * ROL:NUMBER OF PARITY BITS (ROL: 0 OR 1)
00232A 0245 4F                           CLR A
00233A 0246 05                           ASLD
00234A 0247 97 50      A                 STA A   ROH        * LSB <--- PARITY
00235                             *
00236             0249 A          SNDR20 EQU     *
00237A 0249 7B 4008    A                 TIM     #$40,TCSR * OCR OVERFLOW ?
00238A 024C 26 0B 0259                   BNE     SNDR30
00239                             * NOT OVERFLOW
00240A 024E DC 0B      A                 LDD     OCR        * 'TIME TILL NEXT EDGE' < 1.6*$20 MICRO SEC ?
00241A 0250 93 09      A                 SUBD    FRC        * YES, THEN WAIT OCR OVERFLOW, NOW START 'START
00242A 0252 83 0020    A                 SUBD    #$20       *  BIT'.
00243A 0255 2B F2 0249                   BMI     SNDR20     * NO, THEN WAIT TIME OF START BIT
00244A 0257 20 07 0260                   BRA     SNDR40
00245                             * OCR OVER, SET NEXT TIME
00246A 0259 DC 09      A          SNDR30 LDD     FRC        * SET TIME OF START BIT
00247A 025B C3 0020    A                 ADDD    #$20
00248A 025E DD 0B      A                 STD     OCR
00249             0260 A          SNDR40 EQU     *
00250A 0260 71 FE08    A                 AIM     #$FF-$01,TCSR * SET 'LOW'
00251                             *
00252             0263 A          SNDR45 EQU     *
00253A 0263 7B 4008    A                 TIM     #$40,TCSR * WAIT UNTIL OVERFLOW
00254A 0266 27 FB 0263                   BEQ     SNDR45
00255                             * SET NEXT DATA BIT
00256A 0268 5F                           CLR B
00257A 0269 77 0052    A                 ASR     R1H
00258A 026C 59                           ROL B              * (B)  0 OR 1
00259A 026D 26 05 0274                   BNE     SNDR50
00260                             * SET  0
00261A 026F 71 FE08    A                 AIM     #$FF-$1,TCSR
00262A 0272 20 06 027A                   BRA     SNDR53     * (PARITY IS NOT CHENGED)
00263                             * SET 1
00264             0274 A          SNDR50 EQU     *
00265A 0274 72 0108    A                 OIM     #$1,TCSR
00266A 0277 75 0150    A                 EIM     #$1,ROH    * COMPULE PARITY
00267                             *
00268A 027A E8 05      A          SNDR53 EOR B   RSBCC+1-RSWKTP,X * COMPUTE CRC
00269A 027C A6 04      A                 LDA A   RSBCC-RSWKTP,X
00270A 027E 04                           LSRD
00271A 027F 24 04 0285                   BCC     SNDR54
00272A 0281 A8 02      A                 EOR A   RSCRC-RSWKTP,X
00273A 0283 E8 03      A                 EOR B   RSCRC+1-RSWKTP,X
00274A 0285 ED 04      A          SNDR54 STD     RSBCC-RSWKTP,X
00275                             * SET NEXT TIME
```

```
00276A 0287 DC 0B     A             LDD    OCR
00277A 0289 E3 00     A             ADDD   RSBAUD-RSWKTP,X
00278A 028B DD 0B     A             STD    OCR
00279A 028D 7A 0053   A             DEC    R1L       * FINISHED ?
00280A 0290 26 D1 0263              BNE    SNDR45
00281                       * ADD PARITY ?
00282A 0292 96 51     A             LDA A  R0L
00283A 0294 26 0A 02A0              BNE    SNDR60
00284A 0296 D6 50     A             LDA B  R0H       * SET PARITY  (R1H <--- R0H)
00285A 0298 4C                      INC A            * 'ADD PARITY' FLAG <--- 'NONE' (R0L <--- 0)
00286A 0299 DD 51     A             STD    R0L
00287A 029B 7C 0053   A             INC    R1L       * BIT COUNT <--- 1
00288A 029E 20 C3 0263              BRA    SNDR45
00289                       *
00290                       * ADD STOP BITS
00291          02A0   A      SNDR60 EQU    *         * WAIT UNTIL START OF LAST BIT
00292A 02A0 7B 4008   A             TIM    #$40,TCSR
00293A 02A3 27 FB 02A0              BEQ    SNDR60
00294                       *
00295A 02A5 DC 0B     A             LDD    OCR
00296A 02A7 E3 00     A             ADDD   RSBAUD-RSWKTP,X
00297A 02A9 DD 0B     A             STD    OCR
00298                       *
00299A 02AB 72 0108   A             OIM    #$1,TCSR  * STOP BIT
00300          02AE   A      SNDR70 EQU    *
00301A 02AE 7B 4008   A             TIM    #$40,TCSR * WAIT UNTIL START TIME OF STOP BIT
00302A 02B1 27 FB 02AE              BEQ    SNDR70
00303                       *
00304A 02B3 EE 06     A             LDX    RSMODS-RSWKTP-1,X
00305A 02B5 18                      XGDX             * (X):OCR LAST TIME,
00306A 02B6 C4 03     A             AND B  #$3       * (B):MSMODS (LS 3BITS:NUMBER OF STOP BITS)
00307A 02B8 26 01 02BB              BNE    SNDR80
00308A 02BA 5C                      INC B            * IF 0, 1 STOP BIT
00309A 02BB 4F               SNDR80 CLR A            * (X): NUMBER OF STOP BITS
00310A 02BC 18                      XGDX
00311A 02BD F3 01AF   A      SNDR90 ADDD   RSBAUD    * (A,B):HIGH BIT TIME
00312A 02C0 09                      DEX
00313A 02C1 26 FA 02BD              BNE    SNDR90
00314                       *
00315A 02C3 DD 0B     A             STD    OCR
00316                       *
00317A 02C5 38                      PULX
00318A 02C6 32                      PUL A
00319A 02C7 0E                      CLI              * IF RECEIVED KEY INTERRUPT, KEY SAMPLING TIME
00320                       *                       * NOT PUNCTUAL.
00321A 02C8 5F                      CLR B
00322A 02C9 39                      RTS
00323                       *
00324          0000   A             END
***** TOTAL ERRORS     0
```

```
       *** 6301 CROSS MACROASSEMBLER   VER1.0 ***                                          10/21/82    09:42:16

ERR    SEQ    LOC  OBJECT         PROGRAM  TERM         --- TERMINAL MODE WITHOUT HARD COPY ---

     00001                               *
     00002                                      NAM    TERM
     00003                               * TSS TERMINAL MODE
     00004                               * 300 BPS, FULL DUPLEX, WITHOUT HARD COPY
     00005                               * FILE NAME    'EX$5'  BY  K.A
     00006                                      TTL    --- TERMINAL MODE WITHOUT HARD COPY ---
     00007                                      OPT    LOAD
     00008                                      OPT    PAGE=55
     00009                               *
     00010A 1000                                ORG    $1000
     00011                               *
     00012                               * EXEMPLE OF TERMINAL MODE
     00013                               *
     00014            FF4F  A            DSPSCR EQU    $FF4F
     00015            FF5E  A            SCRFNC EQU    $FF5E
     00016            FF85  A            RSONOF EQU    $FF85
     00017            FF88  A            RSMST  EQU    $FF88
     00018            FF82  A            RSOPEN EQU    $FF82
     00019            FF7F  A            RSCLOS EQU    $FF7F
     00020            FF79  A            RSGET  EQU    $FF79
     00021            FF76  A            RSPUT  EQU    $FF76
     00022            FF9A  A            KEYIN  EQU    $FF9A
     00023            FF9D  A            KEYSTS EQU    $FF9D
     00024                               *
     00025                               * INITIALIZE
     00026A 1000 CC 8422  A                     LDD    #$8422    * CONSTRUCT SCREEN PACKET
     00027A 1003 FD 105B  A                     STD    SCRPK1
     00028A 1006 86 37    A                     LDA A  #$37
     00029A 1008 B7 105D  A                     STA A  SCRPK2.
     00030A 100B CC 1303  A                     LDD    #$1303
     00031A 100E FD 105E  A                     STD    SCRPK2+1
     00032A 1011 CC 1400  A                     LDD    #$1400
     00033A 1014 FD 1060  A                     STD    SCRPK2+3
     00034A 1017 CE 105B  A                     LDX    #SCRPK1   * INITIALIZE SCREEN
     00035A 101A BD FF5E  A                     JSR    SCRFNC
     00036A 101D CE 105D  A                     LDX    #SCRPK2
     00037A 1020 BD FF5E  A                     JSR    SCRFNC
     00038A 1023 CC 3D27  A                     LDD    #$3D27    * SET MODE(STOP:1 CD:NO-CHECK, RTS:ON, PARITY:EV
     00039                               *                      * 7 BITS LENGTH, 300 BPS
     00040A 1026 BD FF88  A                     JSR    RSMST
     00041A 1029 86 01    A                     LDA A  #1        * RS232C DRIVER ON
     00042A 102B BD FF85  A                     JSR    RSONOF
     00043A 102E FE FFDC  A                     LDX    $FFDC     * (X):BUFFER ADDRESS (SYSTEM BUFFER)
     00044A 1031 CC 0104  A                     LDD    #260      * (A,B): BUFFER SIZE
     00045A 1034 BD FF82  A                     JSR    RSOPEN    * RECEIVE OPEN
     00046                               *
     00047A 1037 BD FF9D  A            REDKEY JSR    KEYSTS    * ACCEPT FROM KEY BOARD ?
     00048A 103A 25 1E 105A                     BCS    BRKRTN    * IF BREAK KEY IS PRESSED, RETURN (IN BASIC MODE
     00049A 103C 27 09 1047                     BEQ    RCVRS
     00050                               * ACCEPTED CHARACTER FROM KB.
     00051A 103E BD FF9A  A                     JSR    KEYIN
     00052A 1041 BD FF76  A                     JSR    RSPUT     * TRANSMIT ACCEPTED CHARACTER.
     00053A 1044 BD FF4F  A                     JSR    DSPSCR    * DISPLAY ACCEPTES CHRACTER TO VIRTUAL SCREEN.
     00054A 1047 FE FFD8  A            RCVRS  LDX    $FFD8     * ARE THERE RECEIVED CHARACTER IN THE BUFFER ?
     00055A 104A EC 00    A                     LDD    0,X
```

```
ERR   SEQ    LOC  OBJECT       PROGRAM  TERM        --- TERMINAL MODE WITHOUT HARD COPY ---

     00056A 104C 27 E9 1037             BEQ     REDKEY
     00057A 104E BD FF79  A             JSR     RSGET
     00058A 1051 81 7F    A             CMP A   #$7F
     00059A 1053 24 E2 1037             BCC     REDKEY      * IGNORE 7F - FF CHARACTERS
     00060A 1055 BD FF4F  A             JSR     DSPSCR      * DISPLAY RECEIVED CHRACTER TO VIRTUAL SCREEN.
     00061A 1058 20 DD 1037             BRA     REDKEY
     00062                       *
     00063A 105A 39              BRKRTN RTS
     00064                       * VIRTUAL SCREEN PACKET
     00065A 105B    84    A      SCRPK1 FCB     $84         * SELECT SCREEN DEVICE (LCD)
     00066A 105C    22    A             FCB     $22
     00067A 105D    87    A      SCRPK2 FCB     $87         * SET SCREEN SIZE AND BUFFER ADDRESS
     00068A 105E    13    A             FCB     19,3
          A 105F    03    A
     00069A 1060    1400  A             FDB     $1400
     00070                       *      :
     00071                       *
     00072          0000  A             END
***** TOTAL ERRORS     0
```

```
ERR   SEQ    LOC  OBJECT      PROGRAM  TERM        --- TSS TERMINAL MODE WITH HARDCOPY ---

    00001                                 NAM    TERM
    00002                                 TTL    --- TSS TERMINAL MODE WITH HARDCOPY ---
    00003                           *
    00004                           *  FILE NAME  'EXS2'        BY K.A
    00005                                 OPT    LOAD
    00006                                 OPT    PAGE=55
    00007                           *
    00008                           * EXEMPLE OF TERMINAL MODE
    00009                           *  300 BPS FULL DUPLEX TERMINAL MODE  (1200 BPS)
    00010                           *  VIRTUAL SCREEN SIZE = 20*4
    00011                           *  RECEIVED AND TRNSMITTED CHARACTERS ARE ABLE TO PRINT TO SERIAL
    00012                           *   PRINTER (MP-80, ...). THE CONNECTOR FOR HARD COPY IS 'SERIAL'.
    00013                           *  HARD COPY ROUTINE IS INCLUDED IN INTERRUPT PROCEDURE.
    00014                           *
    00015                           * CABLE
    00016                           *  1. FOR CONNECT TO MODEM (CP-20)
    00017                           *    OPTINAL CABLE
    00018                           *  2. FOR HARD COPY
    00019                           *    HC-20 SERIAL (DIN 5 PINS)     MP-80 SERIAL (DB-25)
    00020                           *     1 (GROUND)     ------------   7 (GROUND)
    00021                           *     2 (PTX)        ------------   3 (RXD)
    00022                           *     3 (PRX)        ------------   2 (TXD)
    00023                           *     4 (POUT)       ------------   6 (DSR)
    00024                           *     5 (PIN)        ------------   20 (DTR)
    00025                           *     FG             ------------   1 (PROTECTIVE GROUND)
    00026                           *
    00027                           *
    00028                           * OPERATION
    00029                           *  PF1 KEY: START HARD COPY
    00030                           *  PF2 KEY: STOP HARD COPY
    00031                           *  PF3 KEY: 1200 BPS (DISPLAY MONITOR (RECEIVED CHARACTER) = OFF)
    00032                           *  PF4 KEY: 300 BPS
    00033                           *  PF5 KEY: QUIT
    00034                           *  PF6 KEY: MONITOR DISPLAY ON
    00035                           *  PF7 KEY: MONITOR DISPLAY OFF
    00036                           *  PF8 KEY: ESC 'I'+$20 '0'
    00037                           *
    00038                           * 1200 BPS FULL DUPLEX TERMINAL PROCEDURE
    00039                           *   1: PF3 (1200 BPS)
    00040                           *   2: PF6 (MONITOR DISPLAY OFF, HARD COPY ON)
    00041                           *   3: (PF8 ?????)
    00042                           *
    00043                           * SUBROUTINE ENTRY POINT
    00044           FF4F   A        DSPSCR EQU    $FF4F      * DISPLAY ONE CHARACTER TO VIRTUAL SCREEN
    00045           FF5E   A        SCRFNC EQU    $FF5E      * VIRTUAL SCREEN FUNCTION
    00046           FF85   A        RSONOF EQU    $FF85      * RS232C DRIVER ON/OFF
    00047           FF88   A        RSMST  EQU    $FF88      * SET RS232C PARAMETERS
    00048           FF73   A        SERONF EQU    $FF73      * SERIAL DRIVER ON/OFF
    00049           FF82   A        RSOPEN EQU    $FF82      * OPEN RS232C RECEIVE
    00050           FF7F   A        RSCLOS EQU    $FF7F      * CLOSE RS232C RECEIVE
    00051           FF79   A        RSGET  EQU    $FF79      * GET RS232C ONE CHARACTER
    00052           FF76   A        RSPUT  EQU    $FF76      * SEND RS232C ONE CHARACTER
    00053           FF9A   A        KEYIN  EQU    $FF9A      * GET ONE CHARACTER FROM KEYBOARD BUFFER
    00054           FF9D   A        KEYSTS EQU    $FF9D      * GET NUMBER OF CHARACTERS IN THE KEY BUFFER
    00055           FF25   A        MENU   EQU    $FF25      * MENU
```

```
00056                         * CONSTANTS OR REGISTERS
00057          0011      A    TRCSR   EQU    $11         * TRANSMIT/RECEIVE CONTROL REGISTER
00058          0013      A    STDR    EQU    $13         * SERIAL TRANSMIT DATA REGISTER
00059          0012      A    SRDR    EQU    $12         * SERIAL RECEIVE DATA REGISTER
00060          0008      A    TCSR    EQU    $08         * TIMER CONTROL AND STATUS REGISTER
00061          000B      A    OCR     EQU    $0B         * OUTPUT COMPARE REGISTER
00062          0009      A    FRC     EQU    $09         * FREE RUNNING COUNTER
00063          0010      A    RMCR    EQU    $10         * RATE AND MODE CONTROL REGISTER
00064                         *                          * 04:38.4 KBPS,  05:4.4 KBPS
00065          0002      A    PORT1   EQU    $02         * I/O PORT1
00066          0003      A    PORT2   EQU    $03         * I/O PORT2
00067          1000      A    BUFSIZ  EQU    4096        * BUFFER SIZE FOR PRINTER
00068          00C8      A    SCBSIZ  EQU    200         * BUFFER SIZE FOR SCREEN
00069          1000      A    RSBSIZ  EQU    4096        * BUFFER SIZE FOR RS232C
00070          0001      A    ECHODT  EQU    1           * TERMINAL MODE = 'ECHO CHARACTER' ?
00071                         *                          * 0:YES,  1:NO
00072          0109      A    SERVCT  EQU    $109        * SCI RECEIVE INTERRUPT ADDRESS
00073                         *
00074A 1000                           ORG    $1000
00075                         *
00076                         * INITIALIZE
00077A 1000 86 01        A            LDA A  #ECHODT
00078A 1002 B7 11E3      A            STA A  ECHO
00079A 1005 CE 11C7      A            LDX    #SCRPKD     * SET SCREEN PACKET  X:DATA ADDRESS
00080A 1008 C6 0E        A            LDA B  #SCRPKE-SCRPKD * (B):NUBER OF DATA
00081A 100A A6 00        A    INIT10  LDA A  0,X
00082A 100C A7 0E        A            STA A  SCRPK1-SCRPKD,X
00083A 100E 08                        INX
00084A 100F 5A                        DEC B
00085A 1010 26 F8 100A                BNE    INIT10
00086                         *
00087A 1012 CE 11D5      A            LDX    #SCRPK1     * INITIALIZE SCREEN
00088A 1015 BD FF5E      A            JSR    SCRFNC      * SELECT SCREEN DEVICE
00089A 1018 CE 11D7      A            LDX    #SCRPK2
00090A 101B BD FF5E      A            JSR    SCRFNC      * SET SCREEN SIZE AND BUFFER ADDRESS
00091A 101E CE 11DC      A            LDX    #SCRPK3     *
00092A 1021 BD FF5E      A            JSR    SCRFNC      * SET CURSOR MARGIN
00093A 1024 CE 11DE      A            LDX    #SCRPK4     *
00094A 1027 BD FF5E      A            JSR    SCRFNC      * SET SCROLL STEP
00095A 102A CE 11E1      A            LDX    #SCRPK5     *
00096A 102D BD FF5E      A            JSR    SCRFNC      * SET SCROLL SPEED
00097                         *
00098A 1030 86 01        A            LDA A  #1          * MONITOR ON
00099A 1032 B7 11EA      A            STA A  MONFLG
00100                         *
00101A 1035 CC 11F1      A            LDD    #BUF        * SET BUFFER POINTER FOR HARD COPY
00102A 1038 FD 11EB      A            STD    BPIN
00103A 103B FD 11ED      A            STD    BPOUT
00104A 103E CC 0000      A            LDD    #0          * CHARACTER COUNTER = 0
00105A 1041 FD 11EF      A            STD    BUFCNT
00106A 1044 B7 11E4      A            STA A  PRTFLG      * HARD COPY = 'NO'
00107                         * REWRITE SERIAL RECEIVE INTERRUPT VECTOR
00108                         * NOTE. IF WE WANT TO SEND A CHARACTER TO THE PRINTER, WE MAY DETATCH
00109                         *       SLAVE MPU WHILE 20 MILI SECOND AFTER WE GOT THE CHARACTER FROM
00110                         *       SLAVE MCU.
```

```
    00111A 1047 FC 010A  A               LDD     SERVCT+1  * SAVE VECTOR ADDRESS
    00112A 104A FD 11E5  A               STD     SERADR
    00113A 104D CC 1146  A               LDD     #SERINT   * WRITE NEW INTERRUPT ADDRESS
    00114A 1050 FD 010A  A               STD     SERVCT+1
    00115                        *
    00116A 1053 CC 3D27  A               LDD     #$3D27    * SET MODE(STOP:1 CD:NO-CHECK, RTS:ON, PARITY:EV
    00117                        *                        * 7 BITS LENGTH, 300 BPS)
    00118A 1056 FD 11E8  A               STD     RSPARM    * SAVE PARAMETERS
    00119A 1059 BD FF88  A               JSR     RSMST
    00120A 105C 86 01    A               LDA A   #1        * RS232C DRIVER ON
    00121A 105E BD FF85  A               JSR     RSONOF
    00122A 1061 86 01    A               LDA A   #1        * SERIAL DRIVER ON
    00123A 1063 BD FF73  A               JSR     SERONF
    00124                        *
    00125A 1066 CE 22B9  A       INIT30  LDX     #RSBUFF   * (X):BUFFER ADDRESS (SYSTEM BUFFER)
    00126A 1069 CC 1000  A               LDD     #RSBSIZ   * (A,B): BUFFER SIZE
    00127A 106C BD FF82  A               JSR     RSOPEN    * OPEN TO RECEIVE RS232C
    00128A 106F BD FF9D  A       REDKEY  JSR     KEYSTS    * ACCEPT FROM KEY BOARD ?
    00129A 1072 25 7E 10F2*              BCS     BRKRTN    * IF BREAK KEY IS PRESSED, RETURN (IN BASIC MODE
    00130A 1074 27 27 109D               BEQ     RCVRS
    00131                        * ACCEPTED CHARACTER FROM KB.
    00132A 1076 BD FF9A  A               JSR     KEYIN
    00133A 1079 81 FE    A               CMP A   #$FE      * FUNCTION CODES ?
    00134A 107B 26 13 1090               BNE     GETKEY
    00135                        *
    00136                        * FUNCTION KEYS
    00137A 107D C0 F1    A               SUB B   #$F1      * F1 - F10 ?
    00138A 107F 25 1C 109D               BCS     RCVRS     *  NO, IGNORE
    00139A 1081 C1 0A    A               CMP B   #$A
    00140A 1083 24 18 109D               BCC     RCVRS
    00141A 1085 58                       ASL B
    00142A 1086 CE 10D6  A               LDX     #FNCTBL   * GET FUCTION ADDRESS
    00143A 1089 3A                       ABX
    00144A 108A EE 00    A               LDX     0,X       * (X) <-- ENTRY POINT OF EACH SUBROUTINE
    00145A 108C AD 00    A               JSR     0,X
    00146A 108E 20 0D 109D               BRA     RCVRS
    00147A 1090 BD FF76  A       GETKEY  JSR     RSPUT     * TRANSMITTE CHARCTER TO RS232C.
    00148A 1093 F6 11E3  A               LDA B   ECHO      * ECHO ?
    00149A 1096 27 02 109A               BEQ     GETK10
    00150A 1098 8D 1C 10B6               BSR     PSHCHR    * PUSH RECEIVED CHARACTER TO STACK
    00151A 109A BD FF4F  A       GETK10  JSR     DSPSCR    * DISPLAY CHRACTER TO VIRTUAL SCREEN.
    00152                        *
    00153A 109D FE FFD8  A       RCVRS   LDX     $FFD8     * ARE THERE CHARACTERS IN THE RS232C BUFFER ?
    00154A 10A0 EC 00    A               LDD     0,X
    00155A 10A2 27 0F 10B3               BEQ     RCVR80
    00156A 10A4 BD FF79  A               JSR     RSGET
    00157A 10A7 81 7F    A               CMP A   #$7F
    00158A 10A9 24 08 10B3               BCC     RCVR30    * IGNORE 7F - FF CHARACTERS
    00159A 10AB F6 11EA  A               LDA B   MONFLG    * DISPLAY ON ?
    00160A 10AE 27 03 10B3               BEQ     RCVR80
    00161                        *
    00162           10B0  A       RCVR10  EQU     *
    00163A 10B0 BD FF4F  A               JSR     DSPSCR    * DISPLAY CHRACTER TO VIRTUAL SCREEN.
    00164                        *
    00165A 10B3 7E 106F  A       RCVR30  JMP     REDKEY
```

```
ERR  SEQ   LOC  OBJECT        PROGRAM  TERM        --- TSS TERMINAL MODE WITH HARDCOPY ---

   00166                        *
   00167                        *
   00168                        *
   00169                        * PSH RECEIVED CHARACTER TO PRINT STACK
   00170                        * ON ENTRY
   00171                        *  (A): CHARACTER
   00172                        * ON EXIT
   00173                        * REGISTER PRESERVE
   00174                        *  (A), (B)
   00175                        *
   00176A 10B6 7D 11E4  A       PSHCHR TST     PRTFLG     * HARD COPY = YES ?
   00177A 10B9 27 1A 10D5              BEQ     PSHC80
   00178A 10BB 0F                      SEI
   00179A 10BC FE 11EB  A              LDX     BPIN       * PSH A CHARACTER TO THE STACK
   00180A 10BF A7 00    A              STA A   0,X
   00181A 10C1 08                      INX
   00182A 10C2 8C 21F1  A              CPX     #BUF+BUFSIZ
   00183A 10C5 26 03 10CA              BNE     PSHC10
   00184A 10C7 CE 11F1  A              LDX     #BUF
   00185A 10CA FF 11EB  A       PSHC10 STX     BPIN
   00186A 10CD FE 11EF  A              LDX     BUFCNT
   00187A 10D0 08                      INX
   00188A 10D1 FF 11EF  A              STX     BUFCNT
   00189A 10D4 0E                      CLI
   00190A 10D5 39               PSHC80 RTS
   00191                        *
   00192                        *
   00193                        * FUNCTION KEY PROCEDURE TABLE
   00194                        *
   00195A 10D6    10EA  A       FNCTBL FDB     PFKY10     * PF1   (HARD COPY ON)
   00196A 10D8    10EE  A              FDB     PFKY20     * PF2   (HARD COPY OFF)
   00197A 10DA    10F3  A              FDB     PFKY30     * PF3   (1200 BPS)
   00198A 10DC    1108  A              FDB     PFKY40     * PF4   (300 BPS)
   00199A 10DE    1130  A              FDB     PFKY50     * PF5   (QUIT)
   00200A 10E0    1115  A              FDB     PFKY60     * PF6   (MONITOR ON)
   00201A 10E2    1119  A              FDB     PFKY70     * PF7   (MONITOR OFF)
   00202A 10E4    1123  A              FDB     PFKY80     * PF8   (ESC 'I'+$20 '1')
   00203A 10E6    1145  A              FDB     INVLKY     * PF9   (UNDEFINED)
   00204A 10E8    1145  A              FDB     INVLKY     * PF10  (UNDEFINED)
   00205                        *
   00206                        * PF1  PRINT(HARD COPY) ON
   00207A 10EA 86 01    A       PFKY10 LDA A   #$1        * ON PRINT FLAG
   00208A 10EC 20 01 10EF              BRA     PFKY25
   00209                        * PF2  PRINT (HARD COPY) OFF
   00210A 10EE 4F               PFKY20 CLR A              * OFF PRINT FLAG
   00211A 10EF B7 11E4  A       PFKY25 STA A   PRTFLG
   00212          10F2  A       BRKRTN EQU     *
   00213A 10F2 39                      RTS
   00214                        *
   00215                        * PF3 1200 BPS
   00216A 10F3 CC 3D47  A       PFKY30 LDD     #$3D47     * SET MODE(STOP:1 CD:NO-CHECK, RTS:ON, PARITY:E
   00217                        *                        * 7 BITS LENGTH, 1200 BPS)
   00218A 10F6 FD 11E8  A              STD     RSPARM     * SAVE PARAMETERS
   00219A 10F9 BD FF7F  A       PFKY35 JSR     RSCLOS     * CLOSE RS232 FOR OPEN AGAIN.
   00220A 10FC FC 11E8  A              LDD     RSPARM     * CHANGE BIT RATE
```

```
    00221A 10FF BD FF88  A              JSR    RSMST
    00222A 1102 38                      PULX
    00223A 1103 CE 1066  A              LDX    #INIT30    * REWRITE RETURN ADDRESS
    00224A 1106 3C                      PSHX
    00225A 1107 39                      RTS
    00226                        * PF4 300 BPS
    00227A 1108 CC 3D27  A       PFKY40 LDD    #$3D27     * SET MODE(STOP:1 CD:NO-CHECK, RTS:ON, PARITY:EV
    00228                        *                       * 7 BITS LENGTH, 300 BPS)
    00229A 110B FD 11E8  A              STD    RSPARM     * SAVE PARAMETERS
    00230A 110E 86 01    A              LDA A  #1         * DISPLAY MONITOR = ON
    00231A 1110 B7 11EA  A              STA A  MONFLG
    00232A 1113 20 E4 10F9              BRA    PFKY35
    00233                        * PF6  MONITOR ON
    00234A 1115 86 01    A       PFKY60 LDA A  #1
    00235A 1117 20 06 111F              BRA    PFKY75
    00236                        * PF7  MONITOR OFF
    00237A 1119 86 01    A       PFKY70 LDA A  #1         * HARD COPY = ON
    00238A 111B B7 11E4  A              STA A  PRTFLG
    00239A 111E 4F                      CLR A
    00240A 111F B7 11EA  A       PFKY75 STA A  MONFLG
    00241A 1122 39                      RTS
    00242                        * PF8  ESC  'I'+$20 '1'
    00243A 1123 86 1B    A       PFKY80 LDA A  #$1B       * ESC
    00244A 1125 8D 5F 1086*             BSR    PSHCHR
    00245A 1127 86 69    A              LDA A  #'I+$20    * 'I'+$20
    00246A 1129 8D 5B 1086*             BSR    PSHCHR
    00247A 112B 86 31    A              LDA A  #'1        * '1'
    00248A 112D 8D 57 1086*             BSR    PSHCHR
    00249A 112F 39                      RTS
    00250                        * PF5   QUIT
    00251A 1130 BD FF7F  A       PFKY50 JSR    RSCLOS     * CLOSE RS232
    00252A 1133 4F                      CLR A             * DRIVER OFF
    00253A 1134 BD FF85  A              JSR    RSONOF
    00254A 1137 4F                      CLR A
    00255A 1138 BD FF73  A              JSR    SERONF
    00256A 113B FC 11E5  A              LDD    SERADR     * RECOVER INTERRUPT VECTOR
    00257A 113E FD 010A  A              STD    SERVCT+1
    00258A 1141 38                      PULX
    00259A 1142 7E FF25  A              JMP    MENU
    00260                        *
    00261            1145  A     INVLKY EQU    *
    00262A 1145 39                      RTS
    00263                        *
    00264                        *
    00265                        * SERIAL RECEIVE INTERRUPT (RECEIVE RS232C) ROUTINE
    00266                        *  PUSH RECEIVED DATA TO PRINTER STACK AND SEND THE CHARACTER WHICH IS
    00267                        *   IN THE PRINTER STACK
    00268                        *
    00269            1146  A     SERINT EQU    *
    00270A 1146 B6 11E4  A              LDA A  PRTFLG     * HARD COPY = 'YES' ?
    00271A 1149 27 0F 115A              BEQ    SERI80     * NO, JUMP TO INTERRUPT ROUTINE
    00272A 114B 96 11    A              LDA A  TRCSR      * GET DATA
    00273A 114D 96 12    A              LDA A  SRDR
    00274A 114F 84 7F    A              AND A  #$7F       * SUPRESS BIT 7
    00275A 1151 81 7F    A              CMP A  #$7F
```

```
00276A 1153 24 03 1158              BCC     SERI30     * IGNORE 7F - FF
00277A 1155 BD 10B6   A             JSR     PSHCHR
00278                        *. HARD COPY ON
00279A 1158 8D 05 115F       SERI30 BSR     HRDCPY     * SEND 3 CHARACTERS (9 MILI SEC)
00280                        *
00281A 115A FE 11E5   A      SERI80 LDX     SERADR
00282A 115D 6E 00     A             JMP     0,X
00283                        *
00284                        * PRINT TO SERIAL PRINTER
00285                        *  THIS ROUTINE CALLED ONLY IN INTERRUPT
00286                        *  REGISTER PRESERVE
00287                        *     (A)
00288                        *
00289A 115F 36               HRDCPY PSH A
00290A 1160 B6 11E4   A             LDA A   PRTFLG     * HARD COPY = 'YES' ?
00291A 1163 27 60 11C5              BEQ     HARD80
00292                        * YES, PRINTING
00293A 1165 86 03     A             LDA A   #3         * COPY COUNT = 3 (PRINT 3 CHARACTERS)
00294A 1167 B7 11E7   A             STA A   CPYCNT
00295A 116A 7B 4002   A             TIM     #$40,PORT1 * PRINTER READY ?
00296A 116D 26 56 11C5              BNE     HARD80
00297                        * ARE THERE DATA IN THE BUFFER ?
00298A 116F FC 11EF   A             LDD     BUFCNT
00299A 1172 27 51 11C5              BEQ     HARD80
00300                        *
00301A 1174 71 FB03   A             AIM     #$FF-4,PORT2 * DETACH SLAVE MCU, (SELECT SERIAL)
00302A 1177 96 11     A             LDA A   TRCSR
00303A 1179 36                      PSH A              * SAVE TRCSR
00304A 117A 86 05     A             LDA A   #$05       * 4800 BPS
00305A 117C 97 10     A             STA A   RMCR
00306A 117E 86 0A     A             LDA A   #$0A
00307A 1180 97 11     A             STA A   TRCSR
00308                        *
00309A 1182 FE 11ED   A      HARD10 LDX     BPOUT      * LOAD DATA FROM THE STACK
00310A 1185 A6 00     A             LDA A   0,X
00311A 1187 08                      INX
00312A 1188 8C 21F1   A             CPX     #BUF+BUFSIZ
00313A 118B 26 03 1190              BNE     HARD20
00314A 118D CE 11F1   A             LDX     #BUF
00315A 1190 FF 11ED   A      HARD20 STX     BPOUT      * INCREMENT DATA POINTER AT THE BUFFER
00316A 1193 FE 11EF   A             LDX     BUFCNT
00317A 1196 09                      DEX
00318A 1197 FF 11EF   A             STX     BUFCNT
00319                        *
00320A 119A 7B 2011   A      HARD30 TIM     #$20,TRCSR * WAIT READY.
00321A 119D 27 FB 119A              BEQ     HARD30
00322A 119F 97 13     A             STA A   STDR       * STORE DATA TO THE TRANSMIT REGISTER.
00323                        *
00324A 11A1 7A 11E7   A             DEC     CPYCNT     * WERE 3 CHARACTERS SENDED ?
00325A 11A4 27 0A 11B0              BEQ     HARD40
00326A 11A6 7B 4002   A             TIM     #$40,PORT1 * PRINTER READY ?
00327A 11A9 26 05 11B0              BNE     HARD40
00328A 11AB FC 11EF   A             LDD     BUFCNT     * IS BUFFER EMPTY ?
00329A 11AE 26 D2 1182              BNE     HARD10
00330                        *
```

```
00331                              *  WAIT 2 MILI SEC (TIME OF SENDING ONE CHARACTER)
00332A 11B0 7B 2011  A             HARD40 TIM    #$20,TRCSR
00333A 11B3 27 FB 11B0                    BEQ    HARD40
00334A 11B5 CE 0190  A                    LDX    #400
00335A 11B8 09                     HARD50 DEX
00336A 11B9 26 FD 11B8                    BNE    HARD50
00337                              * RECOVER SERIAL COMMUNICATION
00338A 11BB 86 04    A                    LDA A  #$04       * SELECT SLAVE MCU
00339A 11BD 97 10    A                    STA A  RMCR
00340A 11BF 32                            PUL A             * RECOVER TRCSR
00341A 11C0 97 11    A                    STA A  TRCSR
00342A 11C2 72 0403  A                    OIM    #$4,PORT2
00343A 11C5 32                     HARD80 PUL A
00344A 11C6 39                            RTS
00345                              *
00346                              *
00347A 11C7     84   A             SCRPKD FCB    $84        * SCREEN DEVICE SELECT (LCD)
00348A 11C8     22   A                    FCB    $22
00349                              *
00350A 11C9     87   A                    FCB    $87        * SET SCREEN SIZE AND BUFFER ADDRESS
00351A 11CA     13   A                    FCB    19,3
     A 11CB     03   A
00352A 11CC     21F1 A                    FDB    SCRBUF
00353                              *
00354A 11CE     C3   A                    FCB    $C3        * SET CURSOR MARGIN
00355A 11CF     04   A                    FCB    4
00356                              *
00357A 11D0     C4   A                    FCB    $C4        * SET SCROLL STEP
00358A 11D1     0A   A                    FCB    10         * X
00359A 11D2     03   A                    FCB    3          * Y
00360                              *
00361A 11D3     CB   A                    FCB    $CB        * SET SCROLL SPEED
00362A 11D4     09   A                    FCB    9
00363                              *
00364           11D5 A             SCRPKE EQU    *
00365                              *
00366                              *
00367                              * WORK AREA
00368A 11D5     84   A             SCRPK1 FCB    $84        * SCREEN DEVICE SELECT (LCD)
00369A 11D6     22   A                    FCB    $22
00370A 11D7     87   A             SCRPK2 FCB    $87        * SET SCREEN SIZE AND BUFFER ADDRESS
00371A 11D8     13   A                    FCB    19,3
     A 11D9     03   A
00372A 11DA     21F1 A                    FDB    SCRBUF
00373                              *
00374A 11DC     C3   A             SCRPK3 FCB    $C3        * SET CURSOR MARGIN
00375A 11DD     04   A                    FCB    4
00376                              *
00377A 11DE     C4   A             SCRPK4 FCB    $C4        * SET SCROLL STEP
00378A 11DF     0A   A                    FCB    10         * X
00379A 11E0     03   A                    FCB    3          * Y
00380                              *
00381A 11E1     CB   A             SCRPK5 FCB    $CB        * SET SCROLL SPEED
00382A 11E2     09   A                    FCB    9
00383                              *
```

```
     00384A 11E3     01     A       ECHO   FCB    1            * TERMINAL MODE ECHO.
     00385                          *
     00386A 11E4     0001   A       PRTFLG RMB    1            * HARD COPY (MP-80 PRINTER) ON/OFF FLAG
     00387                          *                        * 0:OFF  1:ON
     00388                          *
     00389A 11E5     0002   A       SERADR RMB    2            * SAVE SERIAL RECEIVE INTERRUPT VECTOR.
     00390A 11E7     0001   A       CPYCNT RMB    1            * WORK FOR HARD COPY
     00391A 11E8     0002   A       RSPARM RMB    2            * RS232C OPEN PARAMETER
     00392A 11EA     0001   A       MONFLG RMB    1            * DISPLAY RECEIVED CHARACTER = YES ?
     00393                          *                        *  1:DISPLAY  0:DISPLAY OFF
     00394                          *
     00395                          * SERIAL SEND BUFFER
     00396A 11EB     0002   A       BPIN   RMB    2            * POINTER WHERE NEXT CHARACTER IS STORED
     00397A 11ED     0002   A       BPOUT  RMB    2            * POINTER WHERE NEXT CHARACTER IS LOADED
     00398A 11EF     0002   A       BUFCNT RMB    2            * NUMBER OF CHARACTERS IN THE BUFFER
     00399A 11F1     1000   A       BUF    RMB    BUFSIZ       * BUFFER
     00400                          *
     00401A 21F1     00C8   A       SCRBUF RMB    SCBSIZ       * SCREEN BUFFER
     00402A 22B9     1000   A       RSBUFF RMB    RSBSIZ       * RS232C RECEIVE BUFFER
     00403           32B9   A       WRKEND EQU    *
     00404                          *
     00405                          *
     00406           0000   A              END
***** TOTAL ERRORS     0
```

```
    00001                           *
    00002                                  NAM    MODEM
    00003                                  TTL    --- CONTROL HALF DUPLEX MODEM ---
    00004                           *
    00005                           * TSS TERMINAL OF HALF DUPLEX MODEM
    00006                           *   WITHOUIT HARD COPY
    00007                           * FILE NAME     'EX$8'   BY  K.A
    00008                                  OPT    LOAD
    00009                                  OPT    PAGE=55
    00010                           *
    00011                           *  CONTROL HALF DUPLEX MODEM
    00012                           *
    00013             FF19  A       SNSCOM EQU    $FF19
    00014             FF16  A       CHKRS  EQU    $FF16
    00015                           *
    00016                           *
    00017                           * CONSTANT VALUE
    00018             00FD  A       RSPRM1 EQU    $FD          * STOP BITS = 1, CARRIER DETECT:CHECK
    00019                           *                          * RTS:LOW,  CTS:CHECK  DSR:CHECK
    00020                           *                          * PARITY:NONE
    00021             0048  A       RSPRM2 EQU    $48          * BIT LENGTH = 8, BIT RATE = 1200 BIT/SEC
    00022                           *                          *
    00023                           *
    00024A 1000                            ORG    $1000
    00025                           * ENTRY POINT OF 'START RS232C COMMUNICATION'
    00026                           * PROCEDURE
    00027                           *  1:RTS LOW, SET BIT RATE, DRIVER ON
    00028                           *  2:START TO RECEIVE
    00029                           *
    00030                           *  PARAMETER
    00031                           *   ON ENTRY.  NONE
    00032                           *   ON EXIT.   NONE
    00033                           *
    00034                           *
    00035                           * SUBROUTINE ENTRY POINT
    00036             FF4F  A       DSPSCR EQU    $FF4F        * DISPLAY ONE CHARACTER TO VIRTUAL SCREEN
    00037             FF5E  A       SCRFNC EQU    $FF5E        * VIRTUAL SCREEN FUNCTION
    00038             FF85  A       RSONOF EQU    $FF85        * RS232C DRIVER ON/OFF
    00039             FF88  A       RSMST  EQU    $FF88        * SET RS232C PARAMETERS
    00040             FF82  A       RSOPEN EQU    $FF82        * OPEN RS232C RECEIVE
    00041             FF7F  A       RSCLOS EQU    $FF7F        * CLOSE RS232C RECEIVE
    00042             FF79  A       RSGET  EQU    $FF79        * GET RS232C ONE CHARACTER
    00043             FF76  A       RSPUT  EQU    $FF76        * SEND RS232C ONE CHARACTER
    00044             FF9A  A       KEYIN  EQU    $FF9A        * GET ONE CHARACTER FROM KEYBOARD BUFFER
    00045             FF9D  A       KEYSTS EQU    $FF9D        * GET NUMBER OF CHARACTERS IN THE KEY BUFFER
    00046                           * CONSTANTS OR REGISTERS
    00047             0002  A       PORT1  EQU    $02          * I/O PORT1
    00048             0003  A       PORT2  EQU    $03          * I/O PORT2
    00049             1000  A       RSBSIZ EQU    4096         * BUFFER SIZE FOR RS232C RECEIVE
    00050             0055  A       SCBSIZ EQU    85           * BUFFER SIZE FOR SCREEN
    00051             0001  A       ECHODT EQU    1            * TERMINAL MODE = 'ECHO CHARACTER' ?
    00052                           *                         * 0:YES,  1:NO
    00053                           *
    00054A 1000                            ORG    $1000
    00055                           *
```

```
ERR   SEQ    LOC  OBJECT       PROGRAM  MODEM       --- CONTROL HALF DUPLEX MODEM ---

     00056                           * INITIALIZE
     00057A 1000 CE 10D2  A                 LDX     #SCRPKD   * SET SCREEN PACKET  X:DATA ADDRESS
     00058A 1003 C6 0E    A                 LDA B   #SCRPKE-SCRPKD * (B):NUBER OF DATA
     00059A 1005 A6 00    A          INIT10 LDA A   0,X
     00060A 1007 A7 0E    A                 STA A   SCRPK1-SCRPKD,X
     00061A 1009 08                         INX
     00062A 100A 5A                         DEC B
     00063A 100B 26 F8 1005                 BNE     INIT10
     00064                           *
     00065A 100D CE 10E0  A                 LDX     #SCRPK1    * INITIALIZE SCREEN
     00066A 1010 BD FF5E  A                 JSR     SCRFNC     * SELECT SCREEN DEVICE
     00067A 1013 CE 10E2  A                 LDX     #SCRPK2
     00068A 1016 BD FF5E  A                 JSR     SCRFNC     * SET SCREEN SIZE AND BUFFER ADDRESS
     00069A 1019 CE 10E7  A                 LDX     #SCRPK3    *
     00070A 101C BD FF5E  A                 JSR     SCRFNC     * SET CURSOR MARGIN
     00071A 101F CE 10E9  A                 LDX     #SCRPK4    *
     00072A 1022 BD FF5E  A                 JSR     SCRFNC     * SET SCROLL STEP
     00073A 1025 CE 10EC  A                 LDX     #SCRPK5    *
     00074A 1028 BD FF5E  A                 JSR     SCRFNC     * SET SCROLL SPEED
     00075                           *
     00076A 102B CC 3547  A                 LDD     #$3547     * SET MODE(STOP:1 CD:NO-CHECK, RTS:OFF, PARITY:
     00077                           *                         * 7 BITS LENGTH, 1200 BPS)
     00078A 102E BD FF88  A                 JSR     RSMST
     00079A 1031 86 01    A                 LDA A   #1         * RS232C DRIVER ON
     00080A 1033 BD FF85  A                 JSR     RSONOF
     00081A 1036 CE 1151  A                 LDX     #RSBUF     * (X):BUFFER ADDRESS
     00082A 1039 CC 1000  A                 LDD     #RSBSIZ    * (A,B): BUFFER SIZE
     00083A 103C BD FF82  A                 JSR     RSOPEN     * OPEN RS232C RECEIVE
     00084                           *
     00085A 103F BD FF9D  A          REDKEY JSR     KEYSTS     * ACCEPT FROM KEY BOARD ?
     00086A 1042 25 29 106D                 BCS     BRKRTN     * IF BREAK KEY IS PRESSED, RETURN (IN BASIC MOD
     00087A 1044 27 14 105A                 BEQ     RCVRS
     00088                           * ACCEPTED CHARACTER FROM KB.
     00089A 1046 BD FF9A  A                 JSR     KEYIN
     00090            1049  A        GETKEY EQU     *
     00091A 1049 36                         PSH A
     00092A 104A BD FF4F  A                 JSR     DSPSCR     * DISPLAY CHRACTER TO VIRTUAL SCREEN.
     00093A 104D 32                         PUL A
     00094A 104E 81 0D    A                 CMP A   #$0D       * CR (SEND DATA) CODE ?
     00095A 1050 26 08 105A                 BNE     RCVRS
     00096A 1052 BD 106E  A                 JSR     TXD        * TRANSMITTE DATA STRING TO RS232C
     00097A 1055 86 0A    A                 LDA A   #$0A       * DISPLAY 'LF'
     00098A 1057 BD FF4F  A                 JSR     DSPSCR
     00099                           *
     00100A 105A FE FFD8  A          RCVRS  LDX     $FFD8      * RECEIVED CHARACTER FROM RS232C ?
     00101A 105D EC 00    A                 LDD     0,X
     00102A 105F 27 DE 103F                 BEQ     REDKEY
     00103A 1061 BD FF79  A                 JSR     RSGET
     00104A 1064 81 7F    A                 CMP A   #$7F
     00105A 1066 24 D7 103F                 BCC     REDKEY     * IGNORE 7F - FF CHARACTERS
     00106A 1068 BD FF4F  A                 JSR     DSPSCR
     00107A 106B 20 D2 103F                 BRA     REDKEY
     00108                           *
     00109A 106D 39                  BRKRTN RTS
     00110                           *
```

```
      00111                             *
      00112                             * SEND DATA ON THE SCREEN
      00113                             * PROCEDURE
      00114                             *  1: RTS ON.  WAIT UNTIL CTS ON.
      00115                             *  2: SEND DATA WHICH ARE DISPLAYED ON THE SCREEN.
      00116                             *  3: RTS LOW.
      00117                             * PARAMETERS
      00118                             *  ON ENTRY   NONE
      00119                             *  ON EXIT    NONE
      00120                             *  REGISTER PRESERVE   NONE
      00121                             *
      00122                             *
      00123            106E  A          TXD    EQU    *
      00124A 106E CE 10EE  A                   LDX    #SCRPK7
      00125A 1071 86 91    A                   LDA A  #$91       * GET EXTENT OF VIRTUAL SCREEN
      00126A 1073 A7 00    A                   STA A  0,X
      00127A 1075 BD FF5E  A                   JSR    SCRFNC
      00128                             *
      00129A 1078 86 4D    A            RETRY  LDA A  #$4D       * RTS HIGH
      00130A 107A BD FF19  A                   JSR    SNSCOM
      00131A 107D 86 01    A                   LDA A  #$1
      00132A 107F BD FF19  A                   JSR    SNSCOM
      00133                             *
      00134                             * WAIT  'CTS:ON'
      00135A 1082 7B 0202  A            CTSON  TIM    #$2,PORT1
      00136A 1085 26 FB 1082                   BNE    CTSON
      00137                             *
      00138                             * SEND DATA BLOCK
      00139A 1087 B6 10F2  A                   LDA A  SCRPK7+4   * CALUCLATE 'NUMBER OF CHARACTERS IN THE
      00140A 108A B0 10F0  A                   SUB A  SCRPK7+2   *  CURRENT LINE.
      00141A 108D 4C                           INC A             * (A):LINE DEPTH
      00142A 108E F6 10F1  A                   LDA B  SCRPK7+3   * (B):LINE WIDTH
      00143A 1091 3D                           MUL
      00144A 1092 C1 50    A                   CMP B  #80        * IF OVER 79, COUNT = 79
      00145A 1094 25 01 1097                   BCS    SND05
      00146A 1096 5A                           DEC B
      00147A 1097 F7 10F6  A            SND05  STA B  SCRPK8+3   * SET NUMBER OF CHARACTERS
      00148A 109A F7 1150  A                   STA B  TXCNT
      00149A 109D B6 10F0  A                   LDA A  SCRPK7+2   *
      00150A 10A0 B7 10F5  A                   STA A  SCRPK8+2   * COORDINATE OF (Y) OF START POINT
      00151A 10A3 7F 10F4  A                   CLR    SCRPK8+1   * COORDINATE OF (X) OF START POINT
      00152A 10A6 CE 10F3  A                   LDX    #SCRPK8    * GET CHARACTERS ON THE SCREEN.
      00153A 10A9 BD FF5E  A                   JSR    SCRFNC
      00154                             *
      00155A 10AC CE 10F4  A                   LDX    #SCRPK8+1
      00156A 10AF A6 00    A            SND10  LDA A  0,X
      00157A 10B1 08                           INX
      00158A 10B2 BD FF76  A                   JSR    RSPUT
      00159A 10B5 7A 1150  A                   DEC    TXCNT
      00160A 10B8 26 F5 10AF                   BNE    SND10
      00161                             *
      00162A 10BA 86 0D    A            SND30  LDA A  #$0D       * SEND 'CR' 'LF'
      00163A 10BC BD FF76  A                   JSR    RSPUT
      00164A 10BF 86 0A    A                   LDA A  #$0A
      00165A 10C1 BD FF76  A                   JSR    RSPUT
```

```
   00166                           *
   00167A 10C4 86 4D    A                 LDA A   #$4D      * RTS:LOW
   00168A 10C6 BD FF19  A                 JSR     SNSCOM
   00169A 10C9 86 00    A                 LDA A   #$00
   00170A 10CB BD FF19  A                 JSR     SNSCOM
   00171                           *
   00172A 10CE BD FF16  A                 JSR     CHKRS     * RESTART RECEIVING
   00173                           *
   00174A 10D1 39                         RTS
   00175                           *
   00176                           *
   00177                           *
   00178A 10D2      84  A          SCRPKD FCB     $84       * SCREEN DEVICE SELECT (LCD)
   00179A 10D3      22  A                 FCB     $22
   00180                           *
   00181A 10D4      87  A                 FCB     $87       * SET SCREEN SIZE AND BUFFER ADDRESS
   00182A 10D5      13  A                 FCB     19,3
        A 10D6      03  A
   00183A 10D7      2151 A                FDB     SCRBUF
   00184                           *
   00185A 10D9      C3  A                 FCB     $C3       * SET CURSOR MARGIN
   00186A 10DA      04  A                 FCB     4
   00187                           *
   00188A 10DB      C4  A                 FCB     $C4       * SET SCROLL STEP
   00189A 10DC      0A  A                 FCB     10        * X
   00190A 10DD      03  A                 FCB     3         * Y
   00191                           *
   00192A 10DE      CB  A                 FCB     $CB       * SET SCROLL SPEED
   00193A 10DF      09  A                 FCB     9
   00194                           *
   00195            10E0 A         SCRPKE EQU     *
   00196                           *
   00197                           *
   00198                           * WORK AREA
   00199A 10E0      84  A          SCRPK1 FCB     $84       * SCREEN DEVICE SELECT (LCD)
   00200A 10E1      22  A                 FCB     $22
   00201A 10E2      87  A          SCRPK2 FCB     $87       * SET SCREEN SIZE AND BUFFER ADDRESS
   00202A 10E3      13  A                 FCB     19,3
        A 10E4      03  A
   00203A 10E5      2151 A                FDB     SCRBUF
   00204                           *
   00205A 10E7      C3  A          SCRPK3 FCB     $C3       * SET CURSOR MARGIN
   00206A 10E8      04  A                 FCB     4
   00207                           *
   00208A 10E9      C4  A          SCRPK4 FCB     $C4       * SET SCROLL STEP
   00209A 10EA      0A  A                 FCB     10        * X
   00210A 10EB      03  A                 FCB     3         * Y
   00211                           *
   00212A 10EC      CB  A          SCRPK5 FCB     $CB       * SET SCROLL SPEED
   00213A 10ED      09  A                 FCB     9
   00214                           *
   00215A 10EE      91  A          SCRPK7 FCB     $91       * GET EXTENT OF VIRTUAL SCREEN.
   00216A 10EF      0004 A                RMB     4
   00217                           *
   00218A 10F3      97  A          SCRPK8 FCB     $97
```

```
     00219A 10F4     005A   A                RMB     90
     00220                          *
     00221A 114E     01     A       ECHO     FCB     1          * TERMINAL MODE ECHO.
     00222                          *
     00223A 114F     0001   A       PRTFLG   RMB     1          * HARD COPY (MP-80 PRINTER) ON/OFF FLAG
     00224                          *                         * 0:OFF  1:ON
     00225A 1150     0001   A       TXCNT    RMB     1          * NUMBER OF CHARACTERS WHICH ARE SENT TO HOST
     00226                          *                         * COMPUTER
     00227                          *
     00228A 1151     1000   A       RSBUF    RMB     RSBSIZ     * RS232C RECEIVE BUFFER
     00229                          *
     00230A 2151     0055   A       SCRBUF   RMB     SCBSIZ     * SCREEN BUFFER
     00231                          *
     00232                          *
     00233           0000   A                END
***** TOTAL ERRORS     0
```

CHAPTER 6

Page

6.1 General 6-1
6.2 Data storage (SAVE) 6-1
6.3 I/O Ports 6-3
6.4 Block format 6-4
6.5 File structure 6-4
6.6 Format of header and EOF blocks 6-5
6.7 Interblock gaps 6-6
6.8 Writing blocks 6-6
6.9 Reading blocks 6-7
6.10 File output 6-9
6.11 File open 6-9
6.12 Functions unique to the built-in microcassette 6-10
6.13 Notes on I/O 6-10
6.14 External cassette subroutines 6-12
6.15 Built-in microcassette subroutines 6-16
6.16 Work areas for external cassette 6-19
6.17 Work areas for built-in microcassette 6-20
6.18 Work areas for external cassette headers 6-21
6.19 Work areas for built-in microcassette headers 6-21

6.1 General

Two types of cassettes may be used as external data storages: an external audio cassette and the built-in microcassette (plug-in option). Data sent to cassettes is recorded sequentially. The average speed of data communication with cassettes is 1300 BPS. The format of data stored in the external audio cassette and that of the built-in microcassette are the same so the two types of cassettes are compatible. The only control line used for the external audio cassette is the remote ON/OFF line (REM). The built-in microcassette, however, is controlled by software and performs fast forward, rewind, write, and playback operations in response to commands in BASIC. The tape counter value is also recorded and displayed by software.

6.2 Data storage (SAVE)

(1) Format of one bit

In the recording format of the cassette, one bit is represented by one pulse (Fig. 6-1).

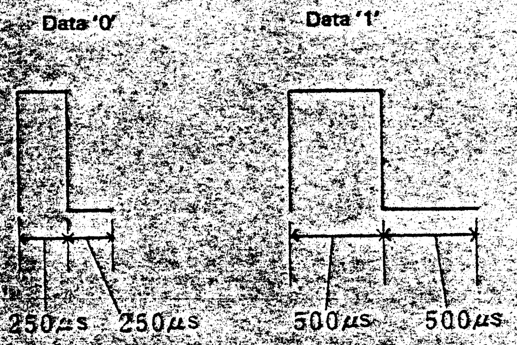


Fig. 6-1 Recording Format for One Bit

Each byte, consisting of 8 data bits and one stop bit, is sent from bit 0. The last bit of a byte is the stop bit (data '1'). (Fig. 6-2.)

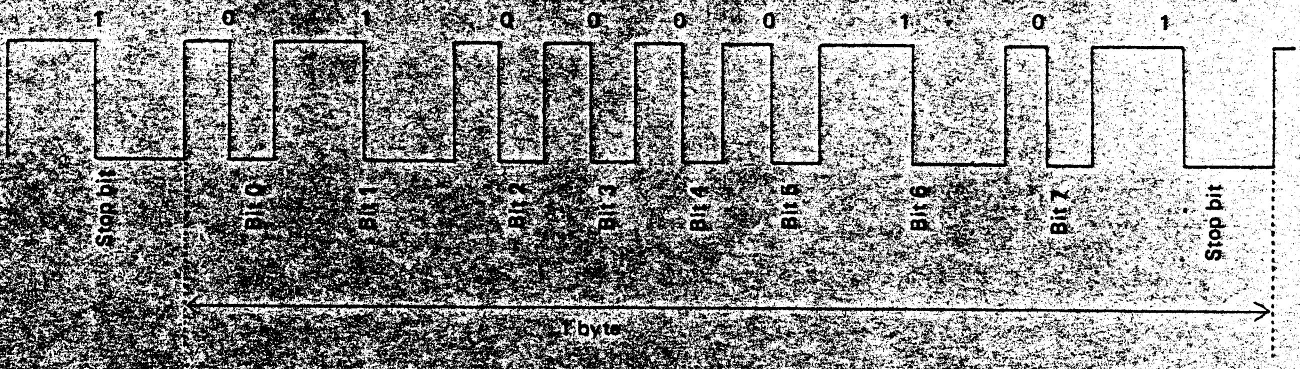


Fig. 6-2 Format of One Byte

(2) Synchronization

The first bit with data '1' which appears after 40 or more bits of data '0' is taken as the first bit (bit 0) of the synchronization character. Synchronization is performed when the data of this first byte is FF and that of the next byte sent is AA (Fig. 6-3).

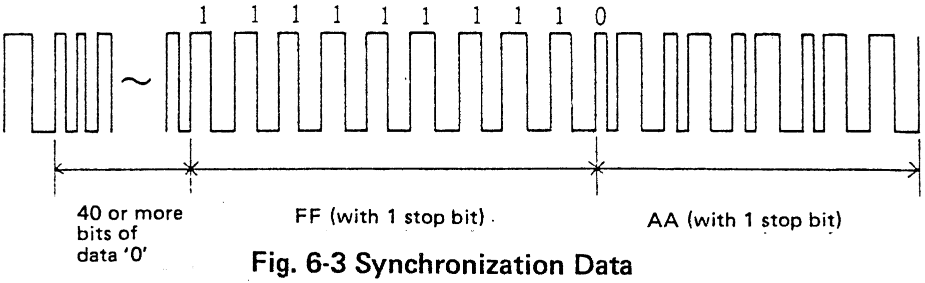


Fig. 6-3 Synchronization Data

The next data sent following the synchronization data will be used as actual data.

(3) Reverse waveform
The normal recording format for data bits is as shown in Fig. 6-1. However, depending on the cassette used, when the signal passes through the playback circuit of the HX-20, the high/low levels of the waveform may be reversed. (Fig. 6-4).

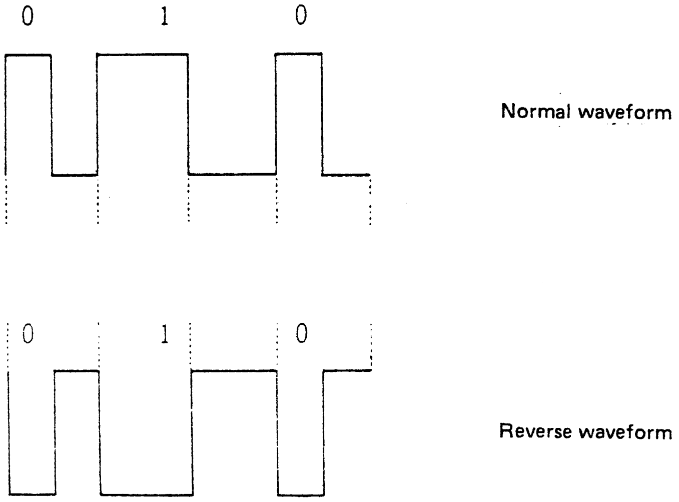


Fig. 6-4 Reverse Waveform

The type of a waveform is determined when synchronization is performed and then data read is performed. The waveform of the built-in microcassette is inverted.

(4) Bit judgement
To judge whether a bit is '0' or '1', the interval between the rise of the first pulse and that of the second is measured. If the measured value is above a specified value (approx. 750 µsec), the bit is judged to be logic '1' (Fig. 6-5)

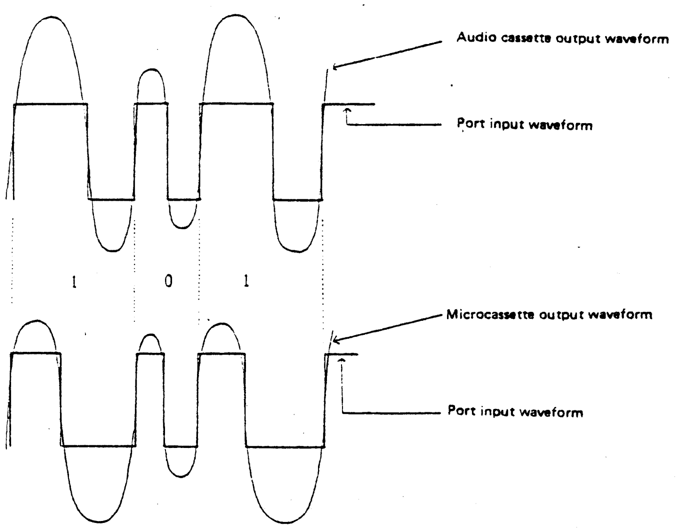


Fig. 6-5 Output Waveforms

6.3 I/O Ports

Table 6-1 lists the I/O ports related to the external cassette.
Table 6-2 lists the I/O ports related to the microcassette.

Table 6-1 I/O Ports Related to the External Cassette

	Port	Description
Master MCU	P12	Input. Connected to port P34 of the slave MCU, this port informs the master MCU of the slave MCU's error status.
Slave MCU	P3Ø	Output. This port is used for the cassette REM output. 1: Off Ø: On
	P32	Input. This port is used to input data from the external cassette. 1: High Ø: Low
	P33	Output. This port is used to output data to the external cassette. 1: High Ø: Low
	P34	Output. This port is connected to port P12 of the master MCU.

Table 6-2 I/O Ports Related to the Microcassette

	Port	Description
Master MCU	P12	Input. This port is connected to port P34 of the slave MCU and informs the master MCU of the slave MCU's error status.
	P17	Input. This port is used to input the counter status or to judge the plug-in option.
Slave MCU	P2Ø	Input. This port is used to input data (1: High, Ø: Low) or to judge the the write protection. The handling of the input contents of this port depends on the value of P45.
	P21	Output. This port is used to output data to the microcassette. 1: High Ø: Low
	P42	Output. This port is used to turn the microcassette power ON/OFF. 1: ON Ø: OFF
	P43	Output. This port is used to set microcassette commands.
	P44	Output. This port supplies a serial clock for timing the microcassette commands. 1: High Ø: Low
	P45	Output. This port is used to select the P2Ø input. Ø: RS-232C 1: Microcassette
	P46	Input. This port is used to input the counter status when port P44 is Ø and the head switch status when it is 1.

6.4 Block format

Cassette data is recorded in blocks. Each block consists of the items listed in Table 6-3.

Table 6-3 Block Configuration

Field	Description
Synchronization field	Contains 80 bits of data '0'.
Preamble	Contains data FF, AA (2 bytes).
Block identification field	This field consists of 4 bytes. The function of each byte is as follows. Byte 0: Block identifier field indicating the type of the block. H: Header D: Data E: End of file (EOD) Bytes 1 and 2: Indicate the 2-byte block number and must be 0000 to FFFF. Byte 3: Block identification number. This is used to identify blocks which are written twice to improve a reliability. Values 00 through FF can be assigned to a block but the values actually used are 00 and 01.
Data field	Stores data. An 80-byte data field is assigned for header (the block identifier field begins with H) and EOF blocks (block identifier field begins with E). In all other cases, the data field size is defined by the header block.
BCC (Block Check Character) field	Performs CRC (Cyclic Redundancy Check) for the range from the beginning of the block to the BCC field. The two BCC bytes and CRC-CCITT are used for this check.
Postamble	Contains values AA, 00 (2 bytes).

6.5 File structure

Only sequential files are supported. Sequential file data is fixed-length and blocked. Each sequential file consists of an 80-byte header block (the length of the data field excluding the preamble, block identification field, BCC, and postamble), one or more data blocks (256 bytes each), and an EOF block. The block numbers assigned for each file begin with header block 00, followed by 01, 02 ... ending with the EOF block. Each block is written twice to improve reliability.

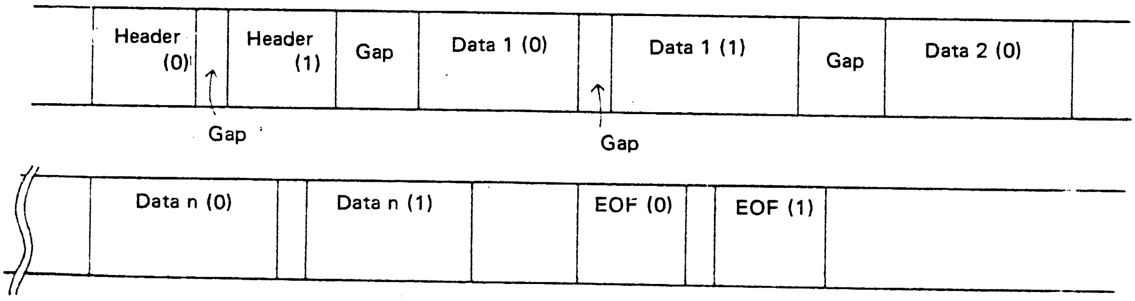


Fig. 6-6 Configuration of Sequential Files

A 5-sec tape feed (data FF) is provided at the beginning and end of each file as a gap to separate files.

6.6 Format of header and EOF blocks
The data format of the header block is shown in Table 6-4 and that of the EOF block is shown in Table 6-5.

Table 6-4 Format of Header Block

Column		Byte	Item	Description
from	to	size		
0	3	4	ID field	Data HDR1. Indicates, in ASCII code, that the block is a header
4	11	8	Filename	Stores the filename.
12	19	8	File type	Stores the file type.
20		1	Record type	This byte specifies the record type. The following record types can be specified. F : Fixed length V : Variable length 2 : Each fixed-length block is written twice. HX-20 currently supports only record type 2.
21		1	Interblock gap length	This byte specifies the interblock gap length "Δ": Interblock gap long enough for the tape to stop (long gap) "S": Interblock gap length not long enough for the tape to stop (short gap)
22	26	5	Block length	Indicates the data length of the block. Must be 00000 to 65535 (ASCII code).
27	31	4		Empty
32	37	6	Creation date	Indicates the date of file creation in "Month, Day, Year" format (ASCII code). Month, day, and year are represented by 2 bytes of data each.
38	43	6	Creation time	Indicates the time of file creation in "Hour, Minutes, Seconds" format (ASCII code). Hour, minute, and second are represented by 2 bytes of data each. Hour is the indicated by the 24-hour system (0 to 23).
44	49			Empty
50	51		Volume serial number	Indicates the tape volume number in ASCII code. (01 ~).
52	59	8	System name	Indicates the name of the system that created the file (ASCII code). "HX-20ΔΔΔ"
60	79	20		Empty

Table 6-5 Format of EOF Block

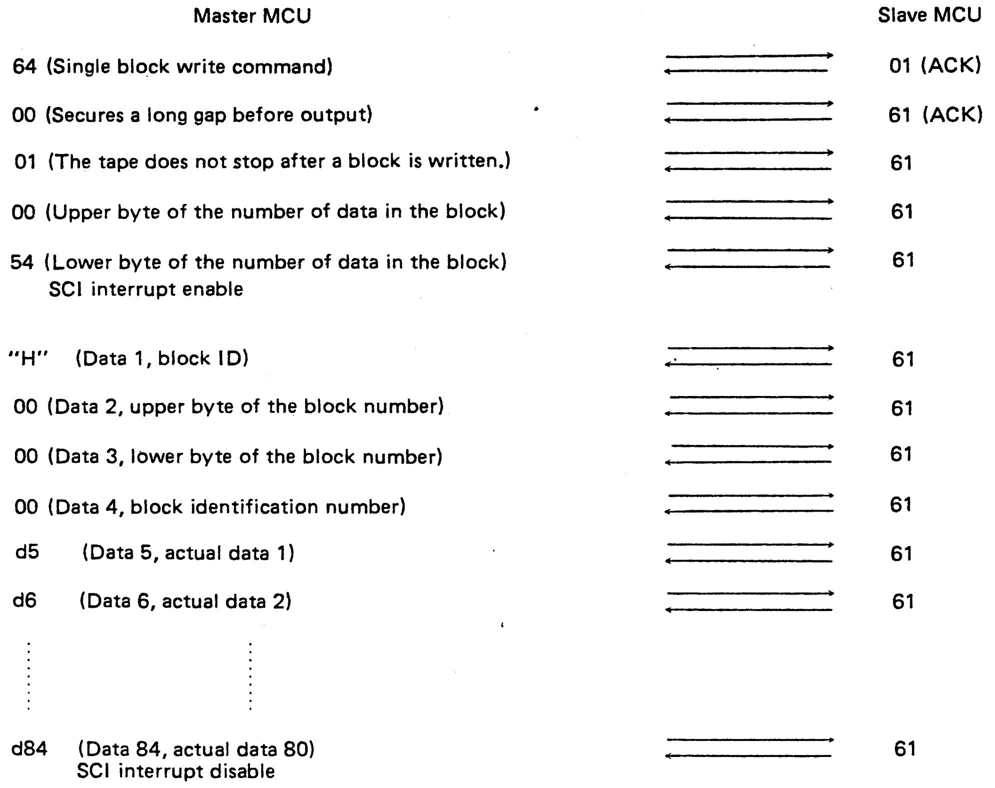
Column		Byte	Item	Description
from	to	size		
0	3	4	ID field	"EOFΔ".
4	79	66		Empty

6.7 Interblock Gaps

There are two types of interblock gaps: long and short. The length of an interblock gap depends on whether the tape will stops at the gap. An interblock gap of approx. 10 bytes (the length of tape require to write a single block twice) is secured between block where the tape will not stop. This is a short gap. An interblock gap of approx. 100 bytes is required when the tape stops between blocks. This type of gap (long gap) enables the motor of the tape drive to reach a constant rotation speed from a halt state. The length of the interblock gap is specified by the header.

6.8 Writing Blocks

Data is written to cassettes by the slave MCU in units of one block. Commands for block write are exchanged between the master and the slave MCUs as shown in Fig. 6-7. The master MCU must send the write data within 4 msec after receiving ACK from the slave. The tape drive must already be running when data is sent to the slave MCU.
The data sent consists of the block ID ("H") and the contents of the data block (84 bytes for the header). CRC calculations are performed solely by the slave MCU.



Master MCU — Slave MCU

64 (Single block write command) — 01 (ACK)

00 (Secures a long gap before output) — 61 (ACK)

01 (The tape does not stop after a block is written.) — 61

00 (Upper byte of the number of data in the block) — 61

54 (Lower byte of the number of data in the block) — 61
SCI interrupt enable

"H" (Data 1, block ID) — 61

00 (Data 2, upper byte of the block number) — 61

00 (Data 3, lower byte of the block number) — 61

00 (Data 4, block identification number) — 61

d5 (Data 5, actual data 1) — 61

d6 (Data 6, actual data 2) — 61

d84 (Data 84, actual data 80) — 61
SCI interrupt disable

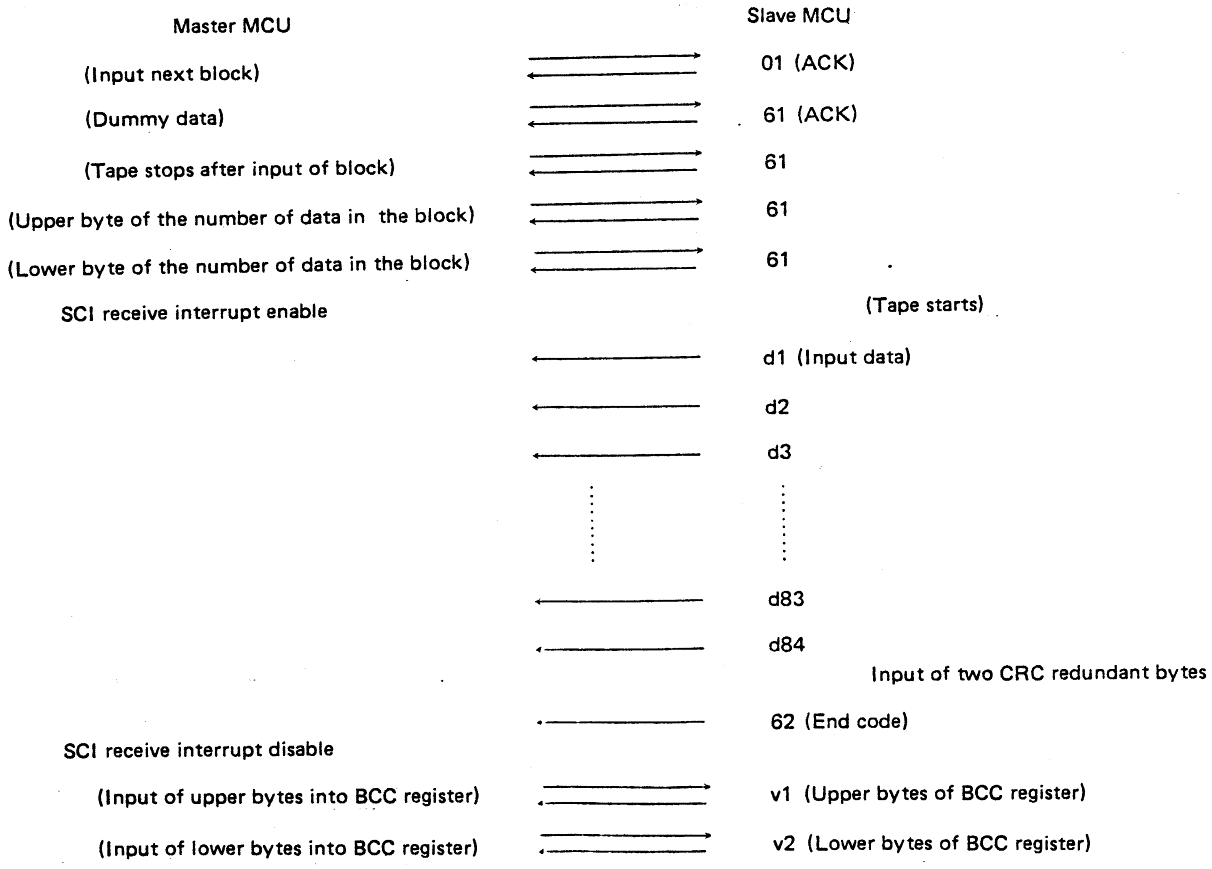
Fig. 6-7 Exchange of Commands for Write Operation for a Single Block (Header)

When the RIE (Receive Interrupt Enable) mask of the SCI (Serial Communication Interface) is opened, the main MCU uses the interrupt routine to transmit the data from "H" to d84 in Fig. 6-7 to the slave MCU. When master MCU receives data 61 from the slave MCU, an SCI interrupt is generated and the master MCU sends next data to the slave MCU.

The RIE mask is closed after one block has been transmitted. The master MCU can transmit data to the slave MCU without generating an SCI interrupt but the current transmission procedure uses the SCI interrupt.

6.9 Reading blocks

Command 28 (26, 27) is used to read a block from the external cassette. Command 68 (66, 67) is used to read a block from the microcassette. The slave MCU transmits to the master MCU the contents of the block, from the beginning of the block identification field to the beginning of the BCC. Redundant bytes used for the CRC check are not sent to the main MCU. When one block has been sent, the slave MCU sends a completion code (22 for the external cassette and 62 for the built-in microcassette) to the master MCU. When the master MCU receives the completion code, it inputs a BCC value to the slave MCU and evaluates the results of the CRC check. CRC check is performed for the range from the block identification field to the CRC redundant byte. If the result of the CRC check is 0, this indicates that the data write operation has been correctly performed. Next, the block number is checked. If block 4 is input when block 5 should be input, the next block must be input. If 6 is input, this means that the desired block has already passed. When a single block has been correctly input, this is taken as the completion of input processing. Otherwise, input processing is aborted or the input procedure for the next block is begun. The master MCU receives the data sent from the slave MCU via the SCI using SCI receive interrupt processing and stores this data in the specified buffer.



Master MCU
Slave MCU
(Input next block)
01 (ACK)
(Dummy data)
61 (ACK)
(Tape stops after input of block)
61
(Upper byte of the number of data in the block)
61
(Lower byte of the number of data in the block)
61
SCI receive interrupt enable
(Tape starts)
d1 (Input data)
d2
d3
d83
d84
Input of two CRC redundant bytes
62 (End code)
SCI receive interrupt disable
(Input of upper bytes into BCC register)
v1 (Upper bytes of BCC register)
(Input of lower bytes into BCC register)
v2 (Lower bytes of BCC register)

Fig. 6-8 Exchange of Commands for Read Operation for a Single Block (Header)

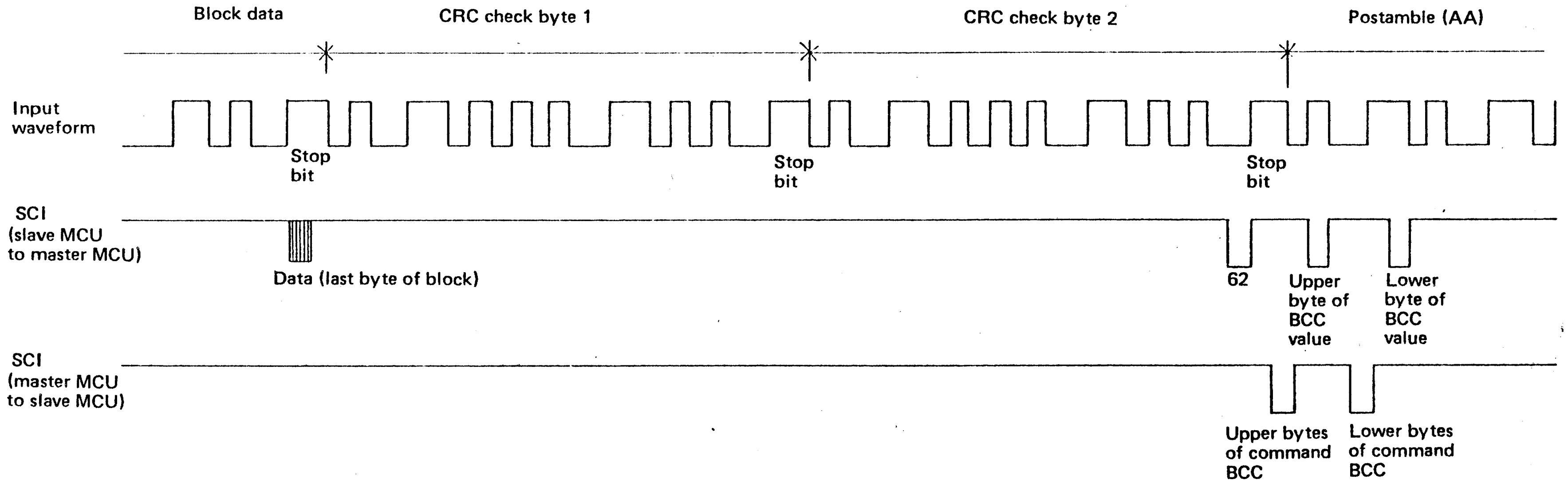


Fig. 6-9 Input Timing for Block Data

6.10 File output

Files are output to cassettes using the following three procedures.

(1) File open

Subroutine OPNWMS is used to open files for output to the built-in microcassette and subroutine OPNWCS performs the same function for the external cassette. When a file is opened for output, the header is output and internal preparations are made for data block output. Specification for the tape to stop after the head block has been output is included.

(2) Output of one byte to a tape file

Subroutine WRITMS outputs data to the built-in microcassette and subroutine WRITCS to the external cassette. Data is written to a buffer (256 bytes of data + block identification data). Actual output to the microcassette or external cassette is performed when the buffer becomes full.

(3) File close

Subroutine CLSMS closes the built-in microcassette file and subroutine CLSCS closes the external cassette file. If any data remains in the buffer when the file is closed, it is output as a data block. An EOF block is then output and the tape stops.

Double write

As a measure to improve reliability, the contents of the buffer are output twice (each block is written twice). This procedure is followed for all blocks (header, data and EOF).

6.11 File open

Files are input from cassettes using the following three procedures.

(1) File open

Subroutines SRCRCS and OPNRCS are used to open files for input from an the external cassette and subroutines SRCRMS and OPNRMS perform the same function for the built-in microcassette. These subroutines search a specified file by inputting a header from the tape and comparing this with the specified file. After the header of the specified file has been input the tape stops and data input is internally prepared.

(2) Input of one byte from a tape file

Subroutine READMS inputs one byte of data from the microcassette and subroutine READCS from the external cassette. Data is fetched one byte at a time from the 256-byte buffer. When the buffer is empty, the next block is written to it from the tape and data fetch continues.

(3) File close

Subroutine CLSMS closes the microcassette file and subroutine CLSCS closes the external microcassette file. The tape stops when one of these subroutines is called. When a file is closed, the corresponding input device is released.

6.12 Functions unique to the built-in microcassette

Fast forward and rewind of the microcassette are performed by the slave MCU in response to commands sent from the master MCU. The slave MCU also starts and stops the motor and reads the tape counter value. The following 4 subroutines are provided.

1. MCSMAN: Performs the operations of the manual operation mode.
2. REWMCS: Rewinds the tape to the beginning.
3. SEKMCS: Winds the tape to the specified tape counter value.
4. CNTMCS: Sets or reads the microcassette tape counter value.

Counter read

The main MCU controls the counter during data input or output. The slave MCU controls the counter at all other times. If there is no change in the counter signal for a specified length of time, it is judged that either no tape is set in the drive or that the tape has been wound to the BOT or EOT position. The tape then stops. Port P17 of the main MCU is used to input the tape counter status. This port value indicates whether the tape counter signal is high or low. The tape counter value is indicated by number of changes in the tape counter signal. When data is being input or output, the main MCU inputs the tape counter signal and performs sampling using a TOF interrupt (0.1-sec interval). The slave MCU controls the counter when fast forward or rewind is being performed.

6.13 Notes on I/O

(1) Polynominals generated for CRC check

The default value ($X^{16} + X^{12} + X^5 + 1$) for polynominal expressions generated for CRC check is set by the slave MCU after reset. This value can be modified by using slave MCU command 48. If the polynominal expression generated at the time of input is different from that generated at the time of output, the system assumes that a CRC error has occurred and no data can be input.

(2) Interblock gaps

When the REM terminal is used for data output to an external cassette, data write will not be correctly performed if the tape drive takes too much time to reach constant running speed from a fully stopped state. When using a tape recorder where such a condition occurs, the interblock gap must be lengthened (slave MCU command 21).

(3) Number of input data in a block

When one of slave MCU commands 28 or 68 (input one block) is input, if the first data input is H or E (header or EOF block), 84 bytes is assumed as the length of the data field of the block and the number of data specified by the command is ignored.

6.14 External cassette subroutines

Subroutine name	Entry point	Description
PONFCS	FF46	Turns ON/OFF the remote (REM) terminal. Parameters: At Entry (A): 0: Turns the REM terminal OFF 1: Turns the REM terminal ON. (Bit 0 is used.) At Return (C): Abnormal I/O flag (A): Return codes 00: Normal (00 is always set in the current version.) Registers retained (B) and (X) Subroutines referenced SNSCOM and CHKRS Variables used None
OPNRCS	FF43	Opens the cassette file for input and searches the specified file until it is found. Parameters: At Entry (X): Starting address of a data packet Data packet 1. Interblock stop mode (1 byte) 00: Tape stops at the interblock gap. 01: Tape does not stop at the interblock gap. FF: According to the header specification. 2. Starting address of input buffer (two bytes, high- to low-byte order) Input buffer size is 260 bytes. 3. 8-byte filename (ASCII code) 4. 8-byte file type (ASCII code) NOTE: If "*" is specified in the character string of a data packet filename, matching terminates at this asterisk position. "*" can also be used in a file type. A file whose filename and type match the specified filename and type is assumed to be the specified file. For example, if the filename is "FILE" and any file type is acceptable, the filename should be specified as "FILEΔΔΔΔ" and the file type as "*ΔΔ". To specify the first file in the tape, both filename and type should be "*ΔΔΔΔΔΔΔ". At Return (C): Abnormal I/O flag (A): Return codes 00: Normal 85: File error (Z): According to the value of (A).

		Registers retained None Subroutines referenced SNSCOM, CRDBHD, and CRDBEF Variables used R1, R2, R3, R4 and R5
SRCRCS	FF40	Opens the cassette file when the first file found is the specified file. Returns the found filename.
		Parameters: At Entry (X): Top address of data packet Data packet 1. Interblock stop mode Same as for subroutine OPNRCS. 2. Starting address of the input buffer Same as for subroutine OPNRCS. 3. Filename Same as for subroutine OPNRCS. 4. File type Same as for subroutine OPNRCS. 5. Found filename (8 bytes) 6. Found file type (8 bytes) Note: The function of "*" in the specification of filename and type is the same as for subroutine OPNRCS. At Return (C): Abnormal I/O flag (A): Return codes 00: Normal 85: File error 8B: The file found is not the specified file. (Z): According to the value of (A). Registers retained None Subroutines referenced SNSCOM, CRDBHD and CRDBEF Variables used R0, R1, R2, R3, R4 and R5
READCS	FF3D	Inputs one byte of data from the external cassette. Input data is fetched from the 256-byte buffer one byte at a time. When the buffer becomes empty, the next block is automatically written to the buffer.
		Parameters: At Entry None At Return (C): Abnormal I/O flag (A): 1-byte input data (B): Return codes 00: Normal 01: End of file (EOF) 84: The input file is not open. 81: Read error (Z): According to the value of (B).

		Registers retained (X) Subroutines referenced CRDBLK Variables used R0, R1, R2, R3, R4 and R5
OPNWCS	FF3A	Opens the external cassette file for output.
		Parameters: At Entry (X): Top address of a data packet Data packet 1. Interblock stop mode (1 byte) 00: Tape stops at the interblock gap. 01: Tape does not stop at the interblock gap. 2. Starting address of output buffer (buffer size is 260 bytes) 3. 8-byte filename (ASCII code) 4. 8-byte file type (ASCII code) At Return (C): Abnormal I/O flag (A): Return codes 00: Normal 88: File is already open. 91: Output error (Z): According to the value of (A). Registers retained None Subroutines referenced CWRHED Variables used R0, R1, R2, R3, R4 and R5
WRITCS	FF37	Outputs one byte of data to the external cassette. Output data is written to the 260-byte buffer. When the buffer becomes full, data is automatically written to the file.
		Parameters: At Entry (A): 1-byte output data At Return (C): Abnormal I/O flag (B): Return codes 00: Normal 94: File is not open. 91: Output error (Z): According to the value of (B). Registers retained (A) and (X) Subroutines referenced CWRBLK

		Variables used R0, R1, R2, R3, R4 and R5
CLSCS	FF34	Closes the external cassette file. When an output file is closed, any data remaining in the buffer is output to the cassette followed by an EOF block. When an input file is closed, input operation simply terminates.
		Parameters: At Entry None At Return (C): Abnormal I/O flag (A): Return codes 00: Normal 87: File is not open. 91: Output error (Z) According to the value of (A). Registers retained None Subroutines referenced WRTCCS, CWRHED and SNSCOM Variables used R0, R1, R2, R3, R4 and R5

6.15 Built-in microcassette subroutines

Subroutine name	Entry point	Description
MCSMAN	FF0D	Performs FF (fast forward) and REW (rewind), etc., according to the keyboard input and displays the tape counter value on the LCD. The keys used for the manual operation mode are as follows. PF1: FF PF2: Slow forward PF3: Stop PF4: REW PF5: Quit. returns from the subroutine PF6: Counter reset This subroutine preserves the contents of the virtual screen while the HX-20 is in the manual operation mode.
		Parameters: At Entry None At Return (C): Abnormal I/O flag (A): Return codes 00: Normal 80: Microcassette is not mounted. (Z): According to the value of (A). Registers retained None Subroutines referenced SNSCOM, KEYIN, KEYSTS, DSPLCN, BINDEC and LRECV Variables used None
OPNRMS	FF0A	Opens the microcassette file for input and searches the specified file until it is found. (see subroutine OPNRCS.)
		Parameters: At Entry and Return Same as subroutine OPNRCS except that return code 80 is also used. Registers retained Same as subroutine OPNRCS Variables used Same as subroutine OPNRCS Subroutines referenced MWRHED
SRCRMS	FF07	Opens the microcassette file. The function of this subroutine is the same as that of subroutine SRCRCS.
		Parameters: At Entry and Return Same as subroutine SRCRCS except that return code 80 is also used. Registers retained Same as subroutine SNSCOS Variables used Same as subroutine SNSCOS Subroutines referenced SNSOOM, MRDBHD and MRDBEF
READMS	FF04	Inputs one byte of data from the microcassette. The function of this subroutine is the same as that of subroutine READCS.

		Parameters: At Entry and Return Same as subroutine READCS except that return code 80 is also used. Registers retained Same as subroutine READCS Variables used Same as subroutine READCS Subroutines referenced WRTMCS, MWRHED and SNSCOM
OPNWMS	FF01	Opens the microcassette file.
		Parameters: At Entry and Return Same as subroutine OPNWCS except that return code 80 is also used. Registers retained Same as subroutine OPNWCS Variables used Same as subroutine OPNWCS Subroutines referenced MWRHED
WRITMS	FEFE	Outputs one byte of data to the microcassette. The function of this subroutine is the same as that of subroutine WRITCS.
		Parameters: At Entry and Return Same as subroutine WRITCS except that return code 80 is also used. Registers retained Same as subroutine WRITCS Variables used Same as subroutine WRITCS Subroutines referenced MWRBLK
CLSMS	FEFB	Closes the microcassette file. The function of this subroutine is the same as that of subroutine CLSCS.
		Parameters: At Entry and Return Same as subroutine CLSCS except that return code 80 is also used. Registers retained Same as subroutine CLSCS Variables used Same as subroutine CLSCS Subroutines referenced WRTMCS, MWRHED and SNSCOM
REWMCS	FEF5	Rewinds the microcassette tape to the beginning.
		Parameters: At Entry None At Return (C): Abnormal I/O flag (A): Return codes 00: Normal 80: Microcassette not mounted. (X): Tape counter value after rewind (-32768 to 32767) (Z): According to the value of (A) Registers retained None Subroutines referenced

		CHKMCS and SNSCOM Variables used R0
SEKMCS	FEF2	Winds the microcassette tape to the specified tape counter value.
		Parameters: At Entry (X): Specified value of the binary counter. (-32768 through 32767) At Return (C): Abnormal I/O flag (A): Return codes 00: Normal 80: Microcassette not mounted. (Z): According to the value of (A). (X): Counter value after wind Registers retained None Subroutines referenced CHKMCS and SNSCOM Variables used R0
CNTMCS	FEEF	Sets or reads the microcassette tape counter value.
		Parameters: At Entry (A): Specifies setting or reading of the tape counter value. 00: Reads the tape counter value. 01: Sets the tape counter value. (Any value other than 00 is taken as 01.) (X): Counter value (A≠00) Return parameters (C): Abnormal I/O flag (0 is always set on return.) (X): Counter value (A=00 at entry) Registers retained (B) Subroutines referenced None Variables used None

6.16 Work areas for external cassette

Address		Variable name	Byte count	Description
15D	1D5	CSMOD	1	Current mode Bits 0 and 1: Format (Bits 1 and 0) = (0, 0): EPSON format Other than (0, 0): Format other than EPSON format Bits 2 and 3: File open status (bit 3, bit 2) = (0, 0): File not open (0, 1): Open for input (1, 0): Open for output (1, 1): Undefined Bits 4 to 7: Undefined
1D6	1D7	CSBLNO	2	Block number
1D8	1D9	CSBCC	2	BCC register value (CRC check for a single block)
1DA	1DB	CSBLSZ	2	Unused
1DC	1DC	CSBSTP	1	Interblock gap tape stop mode 0: Tape stops at the interblock gap. 1: Tape does not stop at the interblock gap.
1DD	1DD	CSSTS	1	Error status (Logic '1' in any bit indicates an error.) Bit 0: EOF (EOF detected during input.) Bits 1 to 3: Underfined Bit 4: Underfined Bit 5: Write error Bit 6: Read error Bit 7: Buffer overflow
1DE	1DF	CSBFAD	2	Starting address of cassette buffer
1E0	1E1	CSBFBT	2	Ending address of cassette buffer plus 1
1E2	1E3	CSBFSZ	2	Cassette buffer size (in bytes)
1E4	1E5	CSBFIP	2	Pointer indicating the next address to be stored in the cassette buffer
1E6	1E7	CSBFOP	2	Pointer indicating the next address to be fetched from the cassette buffer
1E8	1E9	CSBFCM	2	Number of data in buffer
1EA	1EA	CSRDTR	1	Upper limit for the number of block input trials
1EB	1EB	CSRDCN	1	Number of block input trials

6.17 Work areas for built-in microcassette

Address	Variable name	Byte count	Description
1EC 1EC	MSMOD	1	Current mode Bits 0 and 1: Format (Bits 1 and 0) = (0, 0): EPSON format Other than (0, 0): Format other than EPSON format Bits 2 and 3: File open status (0, 0): File not open (0, 1): Open for input (1, 0): Open for output (1, 1): Undefined Bits 4 to 7: Undefined
1ED 1EE	MSGLNO	2	Block number
1EE 1F0	MSBCC	2	BCC register value (CRC check for a single block)
1F1 1F2	MSBLSZ	2	Unused
1F3 1F3	MSBSTP	1	Interblock gap tape stop mode 0: The tape stops at the interblock gap. 1: The tape does not stop at the interblock gap.
1F4 1F4	MSSTS	1	Error status (Logic '1' in any bit indicates an error.) Bit 0: EOF (EOF detected during input.) Bits 1 through 3: Underfined Bit 4: Counter not updated Bit 5: Write error Bit 6: Read error Bit 7: Buffer overflow
1F5 1F6	MSBFAD	2	Starting address of microcassette buffer
1F7 1F8	MSBFBT	2	Ending address of microcassette buffer +1
1F9 1FA	MSBFSZ	2	Microcassette buffer size (in bytes)
1F9 1FC	MSBFIP	2	Pointer indicating the next address to be stored in the buffer
1FD 1FE	MSBFOP	2	Pointer indicating the next address to be fetched from the buffer
1FF 200	MSBFCM	2	Number of data in buffer
201 201	MSRDIR	1	Upper limit for the number of block input trials
202 202	MSRDCN	1	Number of block input trials
203 204	MSCNTR	2	Counter value
205 205	MSMNCM	1	Manual command currently being executed
206 206	MTOFCN	1	Sampling timeout counter for data I/O
207 207	MSPLMD	1	Counter pulse status (low or high)

6.18 Work areas for external cassette headers

Address	Variable name	Byte count	Description
2DØ 2DØ	CHBLID	1	'H'
2D1 2D2	CHBLNO	2	Block number (binary, Ø ...)
2D3 2D3	CHBLBU	1	Same block, block number (Ø, 1 ...)
2D4 2D7	CID	4	HDR
2D8 2DF	CFNAME	8	Filename
2EØ 2E7	CFTYPE	8	File type
2E8 2E8	CRTYPE	1	Record type (2: Double write)
2E9 2E9	CBMODE	1	Block mode S: Short gap Δ: Interblock gap stop
2EA 2EE	CBLNG	5	Block length (ΔΔ256: 256)
2EF 2F3		5	
2F4 2F9	CDATE	6	Data (MMDDYY)
2FA 2FF	CTIME	6	Time (HHMMSS)
3ØØ 3Ø5		6	
3Ø6 3Ø7	CVOLN	2	Volume number
3Ø8 3ØF	CSYSN	8	System name (HX-2ØΔΔΔ)
31Ø 323		2Ø	

6.19 Work areas for built-in microcassette headers

Address	Variable name	Byte count	Description
324 324	MHBLID	1	'H'
325 326	MHBLNO	2	Block number
327 327	MHBLBU	1	Same block, block number
328 32B	MID	4	HDR1
32C 333	MFNAME	8	Filename
334 33B	MFTYPE	8	File type
33C 33C	MRTYPE	1	Record type (2: Double write)
33D 33D	MBMODE	1	Block mode S: Short gap Δ: Interblock gap stop
33E 342	MBLNG	5	Block length (ΔΔΔ256: 256)
343 347		5	
348 34D	MDATE	6	Date (MMDDYY)
34E 353	MTIME	6	Time (HHMMSS)
354 359		6	
35A 35B	MVOLN	2	Volume number
35C 363	CSYSN	8	System name (HX-2ØΔΔΔ)
364 377		2Ø	
378 47B	CASBUF	26Ø	Buffer used by the microcassette

CHAPTER 7 MICROPRINTER

Page

7.1 General 7-1
7.2 Print Head and Solenoids 7-1
7.3 Ports 7-2
7.4 Slave MCU Commands 7-2
7.5 List of Printer Subroutines 7-4
7.6 Microprinter Work Areas 7-6

APPENDIX Sample Lists 7-7

7.1 General

The built-in microprinter is a dot matrix printer with a print width of 144 dots. Printing is performed by a single print head driven by four solenoids. Print mode is unidirectional and paper feed is performed each time the print head is returned. The I/O ports related to printing are connected to the slave MCU which controls printing. The bit patterns for printing, however, are supplied by the master MCU.

7.2 Print Head and Solenoids

The microprinter has one print head and four solenoids: A, B, C and D. Each solenoid prints 36 dots during a single pass of the print head. (Fig. 7-1). Only unidirectional printing is performed and line feed of one dot-line is performed when the head is returned (Fig. 7-2).

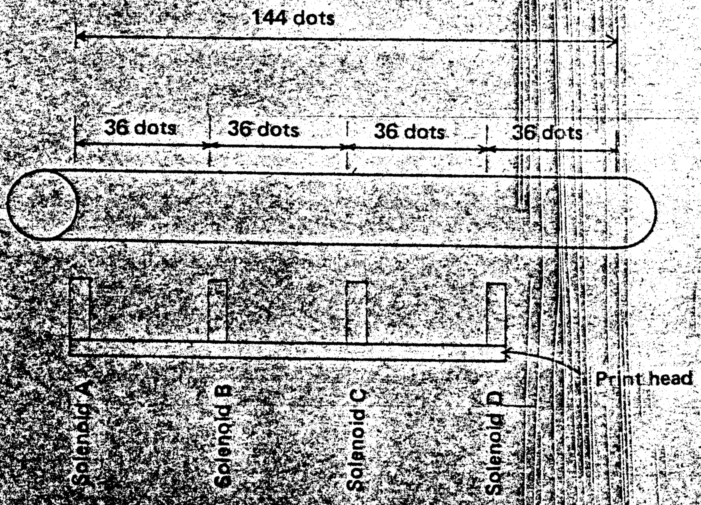


Fig. 7-1 Print Area of Each Solenoid

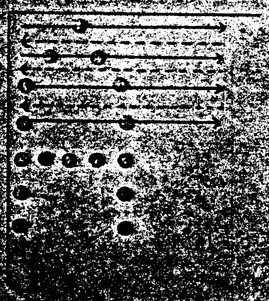


Fig. 7-2 Print Head Operation

Thus, to print a single 6 x 8-dot character pattern, the print head must make 8 passes in each direction.
When printing "ABCDEFGHIJKLMNOPQRSTUVWX", characters "ABCDEF" are printed by solenoid A, "GHIJKL" are printed by solenoid B, "MNOPQR" by solenoid C, and "STUVWX" by solenoid D.
The printer is controlled by the slave MCU, but actual printing is performed in response to commands sent from the master MCU.

7.3 Ports

The I/O ports related to the printer are as follows.

	Port	Input/ Output	Function		
Slave MCU	P1Ø	Output	Print solenoid 1	1: ON	Ø: OFF
	P11	Output	Print solenoid 2	1: ON	Ø: OFF
	P12	Output	Print solenoid 3	1: ON	Ø: OFF
	P13	Output	Print solenoid 4	1: ON	Ø: OFF
	P14	Output	Motor output	1: ON	Ø: OFF
	P15	Input	Reset signal input	1: High	Ø: Low
	P16	Input	Timing pulse	1: High	Ø: Low
	P17	Output	Motor break	1: Break ON	Ø: Break OFF

NOTE:
Commands must not be sent from the master MCU which will operate the above ports to supply current to the print solenoids for more than a few seconds or to supply a BREAK signal while motor output is specified (P14 is 1).

7.4 Slave MCU Commands

The slave MCU is provided with a command for printing 6 dots of print data. This command is sent from the master MCU 24 times to print one dot-line. Therefore, sending this command 48 times will print 2 dot-lines and sending it 192 (24 x 8) times will print one line of 6 x 8-dot character patterns.

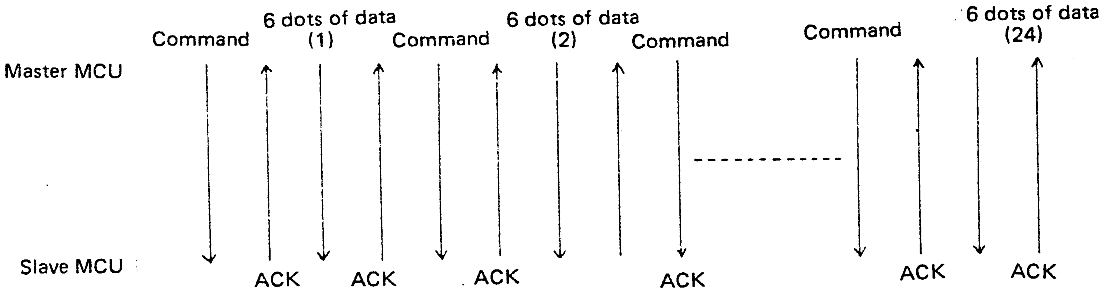
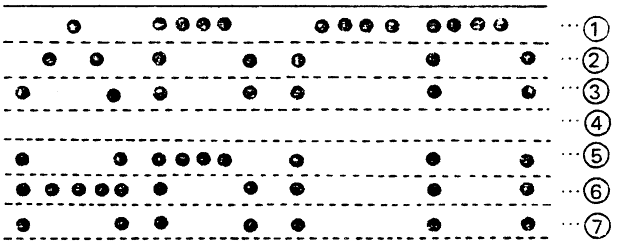


Fig. 7-3 Transmission of Slave MCU Command

If printing is resumed after being interrupted (the print head stops), a blank of one dot-line will occur. This is due to the automatic paper feed (one pitch) when the print head is returned and to the fact that the head stop and restart operation has not finished within the duration of the head's return pass across the page.



Lines ①, ② and ③ are printed normally. If printing is stopped after line ③, one dot line will be left blank by automatic line feed when the printing lines.

Fig. 7-4 One Blank Dot-line when Print Head Stops

After the slave MCU restarts printing on the printer and a new line is to be printed, if there are less than 24 bytes of data in the data buffer, printing is stopped automatically. When continuously printing a given print pattern, if an interrupt in command transmission from the master to the slave MCU of approx. 300msec occurs, data may be lost (Fig. 7-5).

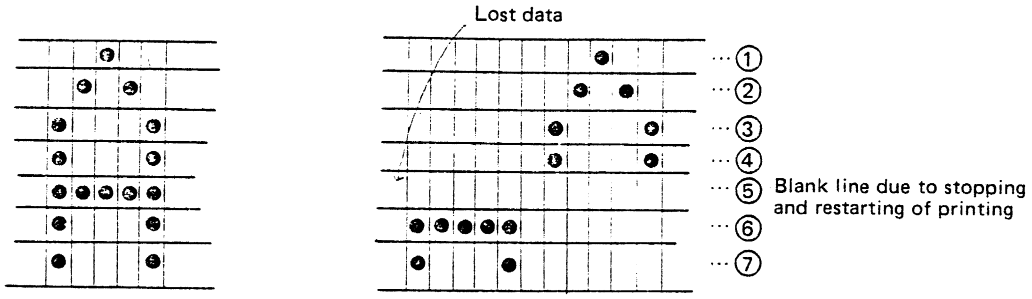


Fig. 7-5 Loss of Print Data

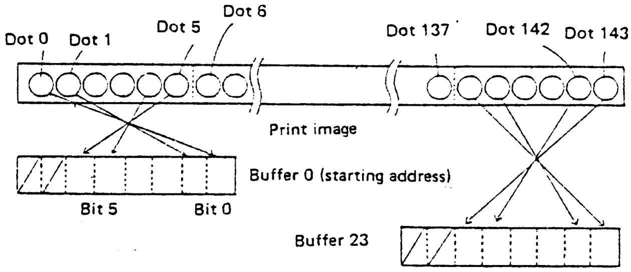
In Fig. 7-5, printing of an A pattern has been attempted. After the data on line 4 has been sent to the slave MCU and blank time has passed, data transmission is performed. Since there is only one byte of data in the slave MCU, printing is stopped. The data in the buffer at this time will be lost.

Printing is resumed when the contents of the buffer exceed 24 bytes. This results in lost print data, as shown in Fig. 7-5.

When printing a line of characters (subroutine "LNPRNT"), after 8 dot-lines of data have been sent, a 2-dot line feed command is sent from the master MCU. In this way, data loss due to timing is prevented (since the feed command processed by the slave MCU stops fetching of the dot pattern data to the buffer).

7.5 List of Printer Subroutines

Subroutine Name	Entry Point	Contents
CHPRNT	FF97	Outputs one character to the microprinter. All control codes (ØØ-IF) except CR(ØD) and LF (ØA) are ignored. For CR, the buffer column position is set to Ø (first position) and the contents of the buffer are cleared. For LF control codes, the contents of the buffer are printed. After printing, the buffer is cleared and the column position is set to Ø.
		Parameters: At Entry (A): Character code (ASCII) At Return (C) Abnormal I/O flag Registered retained (A), (B), (X) Subroutines referenced LNPRNT, CLRB Variables used None
LNPRNT	FF94	Outputs 1 line of characters on the microprinter. Checks for printer switch ON or OFF. If OFF, the output procedure is ignored. Prints 24 characters of the printer buffer contents (ASCII). After printing, the contents of the buffer remain unchanged.
		Parameters: At Entry (X): Starting address of buffer Buffer size: 24 bytes Data is in ASCII code. At Return (C) Abnormal I/O flag Registers retained (A), (B), (X) Subroutines referenced SNSCO, NDFEED, CHKSWT, CHKRS Variables used RØ, R1
PRTDOT	FF91	Prints one dot-line of bit-image data. One dot-line of bit-image print consists of 144 dots and is specified by the 24 bytes in the buffer. Data is entered into the buffer as follows.

Subroutine Name	Entry Point	Contents
		 Dot 0 Dot 1 Dot 5 Dot 6 Dot 137 Dot 142 Dot 143 Print image Buffer 0 (starting address) Bit 5 Bit 0 Buffer 23 Bit 6 and bit 7 of each byte of the buffer have no meaning. NOTE: If during the printing of an image an empty interval occurs until this subroutine is called, a 1-dot blank line will result.
		Parameters: At Entry (X): Starting address of buffer At Return (C) Abnormal I/O flag Registers retained (A), (B), (X) Subroutines referenced SNSCOW, CHKSWT, CHKRS Variables used RØH [Example] When the following is printed. ○●●○○○●●○○○○……… LDX BUFF JSR PRTDOT BUFF FCB $Ø6, $Ø3, ...
NDFEED	FF8F	Performs paper feed for n dot-lines.
		Parameters: At Entry (A): Number of dot-lines of line feed performed At Return (C): Abnormal I/O flag Registers retained (A), (B), (X) Subroutines referenced SNSCOW, CHKRS Variables used None

Subroutine Name	Entry Point	Contents
SCRCPY	FF8B	Copies the data displayed on the LCD on the microprinter. The width of the LCD is 120 dots and that of the printer, 144 dots. The data is left-justified and the remaining 24 dots remain blank.
		Parameters: At Entry None At Return (C): Abnormal I/O flag Registers protect (A), (B), (X) Subroutine referenced SNSCOW, SNSCOM, WRTP26, CHKSWT, LCDMOD Variables used None

7.6 Microprinter Work Areas

Addresses (From)	(To)	Variable Name	Byte Count	Contents
190	195	CHRPTN	6	Work Area for character font (for 1 character).
196	196	COLCNT	1	Data count in buffer (0~24 bytes)
197	1AE	CHRDAT	24	Buffer data for 1 line of characters.

```
    00001                            *
    00002                                   NAM     PRINT
    00003                                   TTL     --- PRINT FULL GRAPHIC PATTERN ---
    00004                                   OPT     LOAD
    00005                                   OPT     PAGE=55
    00006                            *
    00007                            * PRINT FULL GRAPHIC PATTERN TO INTERNAL MICRO PRINTER.
    00008                            * FILE NAME    'EX34'    BY  K.A
    00009                            *
    00010A 1000                             ORG     $1000
    00011                            *
    00012                            *
    00013           FF91   A         PRTDOT EQU     $FF91
    00014                            *
    00015                            *  PRINT PATTERN OF OBLIQUE LINES.
    00016                            *     .  .  .  .  .  .  .  .  .
    00017                            *      .  .  .  .  .  .  .  .  .
    00018                            *       .  .  .  .  .  .  .  .  .
    00019                            *     .  .  .  .  .  .  .  .  .
    00020                            *      .  .  .  .  .  .  .  .  .
    00021                            *       .  .  .  .  .  .  .  .  .
    00022                            *     .  .  .  .  .  .  .  .  .
    00023                            *
    00024                            *
    00025A 1000 86 08      A                LDA A   #8          * (A): REPAETING TIMES
    00026A 1002 C6 03      A         PRTR10 LDA B   #3          * (B): PATTERN NUMBER (3 2 1)
    00027                            *
    00028A 1004 CE 101F    A         PRTRPT LDX     #PATN1      * SET ADDRESS OF PRINT PATTERN
    00029A 1007 C1 03      A                CMP B   #3          * IF (B)=3, PATTERN 1
    00030A 1009 27 0A 1015                  BEQ     PRTR30
    00031A 100B CE 1037    A                LDX     #PATN2      * IF (B)=2, PATTERN 2
    00032A 100E C1 02      A                CMP B   #2
    00033A 1010 27 03 1015                  BEQ     PRTR30
    00034A 1012 CE 104F    A                LDX     #PATN3      * IF (B)=1, PATTERN 3
    00035A 1015 BD FF91    A         PRTR30 JSR     PRTDOT      * PRINT BY GRAPHIC IMAGE.
    00036A 1018 5A                          DEC B
    00037A 1019 26 E9 1004                  BNE     PRTRPT
    00038A 101B 4A                          DEC A               * FINISHED ?
    00039A 101C 26 E4 1002                  BNE     PRTR10
    00040                            *
    00041A 101E 39                          RTS
    00042                            *
    00043A 101F     09     A         PATN1  FCB     $09,$09,$09,$09,$09,$09
         A 1020     09     A
         A 1021     09     A
         A 1022     09     A
         A 1023     09     A
         A 1024     09     A
    00044A 1025     09     A                FCB     $09,$09,$09,$09,$09,$09
         A 1026     09     A
         A 1027     09     A
         A 1028     09     A
         A 1029     09     A
         A 102A     09     A
    00045A 102B     09     A                FCB     $09,$09,$09,$09,$09,$09
```

```
ERR   SEQ    LOC  OBJECT      PROGRAM  PRINT      --- PRINT FULL GRAPHIC PATTERN ---

           A 102C    09     A
           A 102D    09     A
           A 102E    09     A
           A 102F    09     A
           A 1030    09     A
      00046A 1031    09     A               FCB    $09,$09,$09,$09,$09,$09
           A 1032    09     A
           A 1033    09     A
           A 1034    09     A
           A 1035    09     A
           A 1036    09     A
      00047A 1037    12     A        PATN2  FCB    $12,$12,$12,$12,$12,$12
           A 1038    12     A
           A 1039    12     A
           A 103A    12     A
           A 103B    12     A
           A 103C    12     A
      00048A 103D    12     A               FCB    $12,$12,$12,$12,$12,$12
           A 103E    12     A
           A 103F    12     A
           A 1040    12     A
           A 1041    12     A
           A 1042    12     A
      00049A 1043    12     A               FCB    $12,$12,$12,$12,$12,$12
           A 1044    12     A
           A 1045    12     A
           A 1046    12     A
           A 1047    12     A
           A 1048    12     A
      00050A 1049    12     A               FCB    $12,$12,$12,$12,$12,$12
           A 104A    12     A
           A 104B    12     A
           A 104C    12     A
           A 104D    12     A
           A 104E    12     A
      00051A 104F    24     A        PATN3  FCB    $24,$24,$24,$24,$24,$24
           A 1050    24     A
           A 1051    24     A
           A 1052    24     A
           A 1053    24     A
           A 1054    24     A
      00052A 1055    24     A               FCB    $24,$24,$24,$24,$24,$24
           A 1056    24     A
           A 1057    24     A
           A 1058    24     A
           A 1059    24     A
           A 105A    24     A
      00053A 105B    24     A               FCB    $24,$24,$24,$24,$24,$24
           A 105C    24     A
           A 105D    24     A
           A 105E    24     A
           A 105F    24     A
           A 1060    24     A
      00054A 1061    24     A               FCB    $24,$24,$24,$24,$24,$24
           A 1062    24     A
```

```
ERR   SEQ   LOC  OBJECT       PROGRAM  PRINT       --- PRINT FULL GRAPHIC PATTERN ---

          A 1063      24     A
          A 1064      24     A
          A 1065      24     A
          A 1066      24     A
     00055                              *
     00056                              *
     00057                              *
     00058            0000   A                   END
***** TOTAL ERRORS      0
```

CHAPTER 8 ROM CARTRIDGE

Page

8.1 General 8-1
8.2 Configuration 8-1
8.3 Data Input Procedure 8-2
8.4 ROM File 8-3
8.5 Subroutines for ROM Cartridge 8-3
8.6 File Input Procedure 8-4
8.7 Header Format of ROM File 8-4
8.8 ROM Cartridge Subroutine Table 8-5
8.9 Work Area of ROM Cartridge 8-9

APPENDIX - Sample Lists 8-10

8.1 General

The ROM cartridge, which is provided as a plug-in option of the HX-20, can read 2K to 16K bytes of data from an external ROM memory via the I/O ports using its addressing counter and shift register. The addressing counter is incremental and its value can also be reset to 0.
The ROM cartridge is designed for an output-only file as a ROM file to allow data output in this file format.

8.2 Configuration

Table 8.1 shows the I/O ports related to the ROM cartridge.

	Port	I/O	Description
Master MCU	P17	Input	ROM data (1 bit)
	P266	Output	Shift/load select (0: Load; 1: Shift)
	P267	Output	Clock
Slave MCU	P20	Input	ROM cartridge interface judgment
	P46	Input	ROM cartridge interface judgment
	P42	Output	Shift register clear (0: OFF (Clear) 1: ON (Don't clear)
	P43	Output	Power supply (0: OFF 1: ON)
	P44	Output	Addressing counter clear (0: OFF (Clear); 1: ON (Don't clear))

The ROM cartridge is configured as shown in Fig. 8-1. One byte of ROM data at the address indicated by the addressing counter is input to the shift register, which in turn transfers the ROM data to the master MCU.

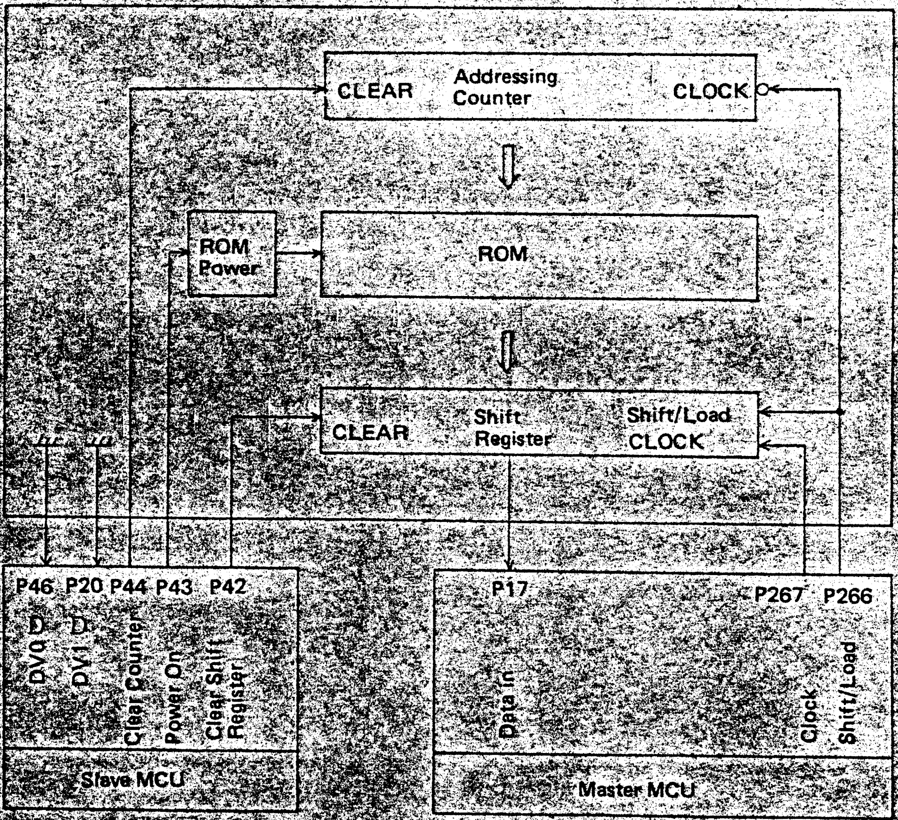


Fig. 8-1 Block Diagram of ROM Cartridge

8.3 Data Input Procedure

Only two types of instructions are applicable to the addressing counter: Clear (by setting the P44 of the slave MCU to "0") and Count-up. Data is fetched by the master MCU from the shift register by inputting one bit of data to the port P17 of the master MCU each time the data bits in the shift register are moved. Data input from the ROM cartridge is performed by the procedure as detailed below.

(1) The power supply of the ROM cartridge is turned ON.
The port P43 of the slave MCU is the power supply port to turn on or off the ROM cartridge. The master MCU instructs the slave MCU to issue a ROM Power ON command to turn on the power supply of the ROM cartridge.

(2) The addressing counter is cleared.
The addressing counter is automatically reset to 0 when the ROM Power ON command is issued to the ROM cartridge from the slave MCU.

(3) The addressing counter is incremented to the address from which data is to be read.
The counter counts up when the voltage level at the port P266 (bit 6 at address 26) of the master MCU changes from High to Low.

(4) When port P266 is at Low level, one byte of data at the address indicated by the addressing counter is loaded into the shift register at the leading edge of a CLOCK signal appearing at the P267 (bit 7 at address 26) of the master MCU. In this case, bit 7 is first loaded into the master MCU through port P17 (Data in).

(5) When port P266 is at High level, the contents of the shift register are shifted by one bit at the trailing edge of the CLOCK signal (P267). By repeating this operation 7 times, one byte of data can be fetched by the master MCU.

(6) If data input from the ROM cartridge is no longer required, the power supply of the ROM cartridge must be turned off by sending a command from the master MCU to the slave MCU to turn off the ROM power supply.

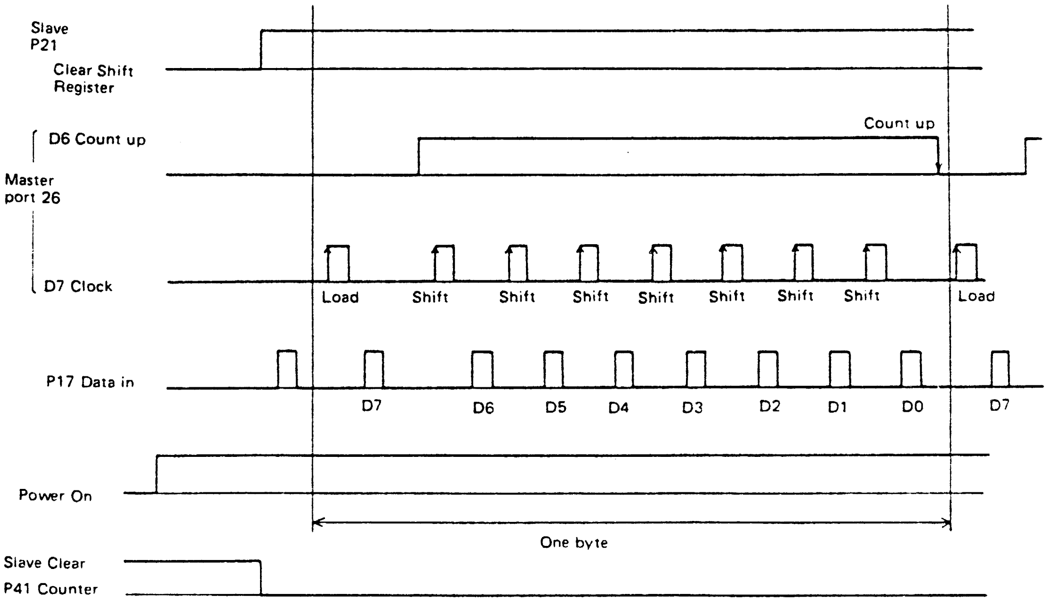


Fig. 8-2 Timing Chart of Data Input from ROM Cartridge

Note: If data is input after clearing the shift register, the data that is input to the master MCU is binary 0. If this Shift Register Clear operation is performed when the optional microcassette drive is connected to the HX-20, binary 1 is input.

8.4 ROM File

Data input from the optional ROM cartridge is supported in the form of data input from a ROM file. The ROM file consists of 32 headers and a data area. Each header may contain a maximum of 32 bytes of data as header information. The ROM file may only be accessed sequentially but not randomly.
Fig. 8-3 shows the structure of the ROM file.

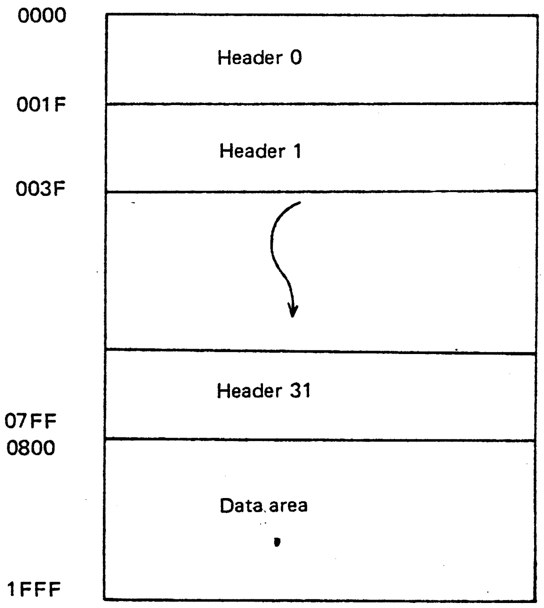


Fig. 8-3 Structure of ROM File

Headers are allocated as fixed areas from address 0000 in units of 32 bytes. Header 0 is from address 0000 to address 001F. A maximum of 32 headers can be set. The first one byte of each header represents the first letter of the filename as well as header information. If the first one byte of a header is "00", it indicates that the file with that header has been deleted. If "FF", it indicates that no subsequent header exists.
If the first one byte of header 2 is "FF", headers 0 and 1 are valid as headers. The contents of the header information are shown in Section 8.7 below.

8.5 Subroutines for ROM Cartridge

The following 4 subroutines are provided for the ROM cartridge:
(1) OPNPRM: Opens the ROM file.
(2) REDPRM: Inputs data from the ROM file in units of one byte.
(3) CLSPRM: Closes the ROM file.
(4) DIRPRM: Inputs the ROM file directory.

8.6 File Input Procedure

A ROM file is processed for data input as follows:

(1) Opening the ROM file
Subroutine "OPNPRM" is used to start the input of data from the ROM file.

(2) Data input
Data is read from the ROM file in units of one byte by subroutine "REDPRM."

(3) Closing the ROM file
Data input from the ROM file is terminated by subroutine "CLSPRM".

Note: Upon opening the ROM file, the ROM cartridge is energized. The ROM file must be closed soon after the data input has been completed particularly when an NMOS type PROM with high power consumption is used.

8.7 Header Format of ROM File

Columns	Bytes	Item	Description
Ø to 7	8	Filename	Filename (in ASCII codes.) Column Ø represents ID in addition to the filename. ØØ: File has been deleted. FF: No subsequent header exists.
8 to 15	8	File type	File type (in ASCII codes)
16 to 19	4	Starting address	The starting address of the ROM area secured as a file. The binary address value is expressed in 4-digit hexadecimal numbers (ASCII codes).
2Ø to 23	4	Ending address +1	The address next to the ending address of the ROM area secured as a file. The binary address value is expressed in 4-digit hexadecimal numbers (ASCII code).
24 to 29	6	Date	Month, day, and year each expressed in 2-digit ASCII codes.
3Ø to 31	2		Unused.

8.8 ROM Cartridge Subroutine Table

Subroutine name	Entry point	Description
OPNPRM	FEEC	ROM file input open °Parameters: At Entry (A): This parameter specifies whether or not the filename is to be returned. 01: Return the filename opened in the packet. 00: Do not return the filename opened in the packet. (X): Starting address of packet °Packet 1. Filename (8 bytes) 2. File type (8 bytes) 3. Filename (8 bytes) (Enter the filename opened when the filename is to be returned.) 4. File type (8 bytes) (Enter the file type opened when the filename is to be returned.) Note: In the filename specification for the packet, if the string specifying a filename contains an asterisk (*), the filename matching terminates at the point of the asterisk and the system assumes that both the filenames have matched. In BASIC Version 1.0, when the matching of the filename with an asterisk (*) terminates, the system assumes that both the file types have also matched. (Note that the ROM file open procedure differs from the cassette file open procedure.) At Return (C): Abnormal I/O flag (A): Return codes 00: Normal A0: ROM cartridge not mounted A1: File not found A2: File already open A4: Invalid header data format A5: Invalid header address format (Z): This parameter depends on the value of parameter (A). °Packet (filename, file type) °Registers retained: None °Subroutines referenced: "PRMPON", "PREDBY", "HEXBIN" and "CLSPRM" °Temporary variables used: R0, R1, and R2

Subroutine name	Entry point	Description
REDPRM	FEE9	Input of one byte from ROM file. °Parameters: At Entry None At Return (C) Abnormal I/O flag (Always ØØ) (A): Input data (B): Return codes ØØ: Normal Ø1: End of file A3: File not opened (Z): This parameter depends on the value of parameter (B). °Registers retained: (X) °Subroutines referenced: "ADSTEP" °Temporary variables used: None
CLSPRM	FEE6	ROM file close. °Parameters: At Entry None At Return (C): Abnormal I/O flag °Registers retained: (B) and (X) °Subroutines referenced: "CHKRS" and "SNSCOM" Note: An attempt to close an unopened ROM file is not regarded by the system as an error.
DIRPRM	FEE3	ROM file directory read. This subroutine specifies record number of the directory and inputs the record. °Parameters: At Entry (A): Directory record number from Ø through 63 (D) (X): Starting address of memory locations here the directory record is stored. The size of each record must be 32 bytes. At Return: (C): Abnormal I/O flag (A): Return codes ØØ: Normal AØ: ROM cartridge not connected A3: Invalid specification of the directory record number (Z): This parameter depends on the value of parameter (A). °Registers retained: None °Subroutines referenced: "PRMPON"' "ADSTEP", "PREDBY" and "CLSPRM" °Temporary variables used: None

8.9 ROM Cartridge Work Areas

Address (from)	(to)	Variable name	Bytes	Description
208	208	PRMSTS	1	Status of the ROM file Bit 0: File open status flag 0: File not opened 1: File opened Bits 1 ~ 6: Undefined Bit 7: Power supply for ROM 0: OFF, 1: ON
209	20A	STAPRS	2	ROM addressing counter
20B	20C	FTADRS	2	Starting address of file
20D	20E	EDADRS	2	Ending address of file + 1

```
ERR   SEQ    LOC   OBJECT       PROGRAM  ROMOPT      --- ROM CARTRIDGE INTERFACE ROUTINE ---

      00001                                     NAM     ROMOPT
      00002                                     TTL     --- ROM CARTRIDGE INTERFACE ROUTINE ---
      00003                              * FILE     'EX$7'  BY  K.A
      00004                                     OPT     PAGE=55
      00005                                     OPT     LOAD
      00006                              *
      00007                              *
      00008                              * COMMON DEFINITION
      00009                              *
      00010                              * MPU 6301 I/O PORT
      00011          0002  A             PORT1  EQU     $02          * I/O PORT 1 (ADDRESS)
      00012          0003  A             PORT2  EQU     $03          * I/O PORT 2 (ADDRESS)
      00013                              *
      00014                              * OTHER REGISTERS
      00015                              * REGISTER MEANINGS
      00016                              *
      00017                              * PORT1 $02
      00018                              *          0:R DATA SET READY (0:HIGH 1:LOW)
      00019                              *          1:R CLEAR TO SEND (0:HIGH 1:LOW)
      00020                              *          2:R PORT TO SLAVE P34 (SFLAG)
      00021                              *          3:R INTERRUPT FROM EXTERNAL PORT (0:INTERRUPT)
      00022                              *          4:R POWER FAIL (0:ABNORMAL)
      00023                              *          5:R KEY BOARD INTERRUPT FLAG (0:INTERRUPT)
      00024                              *          6:R PERIPHERAL STATUS (0:HIGH  1:LOW) (FROM SERIAL)
      00025                              *          7:R MICRO CASSETTE COUNTER / MICRO CASSETTE EXIST
      00026                              *
      00027                              *  $26
      00028                              *          0:W LCD COMAND/DATA 1
      00029                              *          1:W LCD COMAND/DATA 2
      00030                              *          2:W LCD COMAND/DATA 4
      00031                              *          3:W LCD CMMAND/DATA SELCTION (0:DATA 1:COMMAND)
      00032                              *          4:W KEY BOARD INTERRUPT MASK (0:CLOSE 1:OPEN)
      00033                              *          5:W PERIPHERAL CONTROL (TO SERIAL)
      00034                              *          6:W TO PLUG IN 1
      00035                              *          7:W TO PLUG IN 2 AND SLAVE P40
      00036                              *
```

```
    00038                          *   COMMON DEFINITION
    00039                          *
    00040                          * ZERO PAGE RAM
    00041A 004E                           ORG     $4E
    00042A 004E      0001   A      PWRFLG RMB     1            * BIT 0-3: CLOCK POWER ON MODE
    00043                          *                           *  $01:POWER ON BY CLOCK IN BASIC MODE
    00044                          *                           *  $02:POWER ON BY CLOCK IN APLLICATION MODE
    00045                          *                           * BIT 4-7: BEFOR POWER OFF, CALL PROCEDURE MODE
    00046                          *                           *  $01:BEFOR POWER OFF, CALL PROCEDURE IN
    00047                          *                           *.      BASIC MODE.
    00048                          *                           *  $02:BEFOR POWER OFF, CALL PROCEDURE IN
    00049                          *                           *       APLLICATION MODE.
    00050                          *
    00051A 004F      0001   A      P26    RMB     1            * VALUE OF ADDRESS $26
    00052                          * GENERAL REGISTERS USED BY I/O ROUTINE
    00053            0050   A      R0     EQU     *            * 2 BYTES REGISTER  (R0H,R0L)
    00054A 0050      0001   A      R0H    RMB     1
    00055A 0051      0001   A      R0L    RMB     1
    00056            0052   A      R1     EQU     *            * 2 BYTES REGISTER  (R1H,R1L)
    00057A 0052      0001   A      R1H    RMB     1
    00058A 0053      0001   A      R1L    RMB     1
    00059            0054   A      R2     EQU     *            * 2 BYTES REGISTER  (R2H,R2L)
    00060A 0054      0001   A      R2H    RMB     1
    00061A 0055      0001   A      R2L    RMB     1
    00062            0056   A      R3     EQU     *            * 2 BYTES REGISTER  (R3H,R3L)
    00063A 0056      0001   A      R3H    RMB     1
    00064A 0057      0001   A      R3L    RMB     1
    00065                          *
    00066A 007C                           ORG     $7C
    00067A 007C      0001   A      SIOSTS RMB     1            * SLAVE I/O STATUS   (EACH BIT  0:OFF,   1:ON)
    00068                          *                           * BIT 0: PRINTER
    00069                          *                           * BIT 1: EXTERNAL CASSETTE
    00070                          *                           * BIT 2: INTERNAL CASSETTE
    00071                          *                           * BIT 3: RS232C ON (READ)
    00072                          *                           * BIT 4: SPEAKER ON
    00073                          *                           * BIT 5: PROM CASSETTE
    00074                          *                           * BIT 6: BAR CODE READER
    00075                          *                           * BIT 7: BREAK SLAVE CPU (0:ON EXECUTE
    00076                          *                           *                              1:BROKEN BY INTERRUPT)
    00077A 007D      0001   A      MIOSTS RMB     1            * MAIN I/O STATUS  EACH BIT (0:OFF 1:ON)
    00078                          *                           * BIT 0: LCD ON READ/WRITE CHARACTERS
    00079                          *                           * BIT 1: NOW SENDING COMMAND TO SLAVE CPU
    00080                          *                           * BIT 2: NOW TRANSMITTING DATA TO SERIAL (1:ON)
    00081                          *                           * BIT 3: ON CLOCK INTERRUPT (1:ON)
    00082                          *                           * BIT 4: (POWER FAIL)
    00083                          *                           * BIT 5: (OFF POWER SWITCH)
    00084                          *                           * BIT 6: ON PAUSE KEY
    00085                          *                           * BIT 7: ON BREAK KEY
```

```
     00087                              * ROM CASSETTE WORK AREA
     00088A 0208                                ORG    $208
     00089            0208  A           PRWKTP EQU     *             * ROM WORK TOP
     00090A 0208      0001  A           PRMSTS RMB     1             * ROM STATUS (BIT7:POWER ON  1:ON 0:OFF
     00091                              *                            *             (BIT0:OPEN FLAG  1:OPEND  0:CLOSE
     00092A 0209      0002  A           STADRS RMB     2             * ROM ADDRESS COUNTER
     00093A 020B      0002  A           FTADRS RMB     2             * ADDRESS OF TOP OF FILE
     00094A 020D      0002  A           EDADRS RMB     2             * ADDRESS OF LAST OF FILE +1
```

```
 00097A 1000                       ORG     $1000
 00098                      *
 00099          FF2E  A     CHKPLG EQU     $FF2E
 00100          FF19  A     SNSCOM EQU     $FF19
 00101          FF16  A     CHKRS  EQU     $FF16
 00102          FED4  A     WRTP26 EQU     $FED4
 00103          FF2B  A     HEXBIN EQU     $FF2B
 00104                      *
 00105          0051  A     CMPRON EQU     $51        * ROM POWER ON COMMAND TO SLAVE MCU
 00106          0052  A     CMPROF EQU     $52        * ROM POWER OFF COMMAND TO SLAVE MCU
 00107                      *
 00108          012E  A     FILBYT EQU     $12E       * REST BYTES IN THE FILE (2 BYTES SIZE)
 00109                      *
 00110                      *
 00111                      * HEADER FORMAT OF PROM
 00112                      *  00 - 07 (DEC) : FILE NAME    (00: $00:DELETED   $FF:END OF HEADER)
 00113                      *  08 - 15       : FILE TYPE
 00114                      *  16 - 19       : TOP ADDRESS OF THE FILE
 00115                      *  20 - 23       : BOTTOM ADDRESS + 1
 00116                      *  24 - 29       : DATE
 00117                      *  30 - 31       : NOT USED
 00118                      *
 00119                      *
 00120                      *
 00121                      *  FUNCTION :  OPEN TO READ
 00122                      *  ON ENTRY
 00123                      *    (A)=READ MODE(0:NOT ANSWER FILE NAME
 00124                      *                  1:ANSWER FILE NAME )
 00125                      *    (X)=PACKET ADDRESS
 00126                      *          PACKET  0-7: FILE NAME
 00127                      *                  8-15: FILE TYPE
 00128                      *  ON EXIT
 00129                      *    (A)=RETURN CODE
 00130                      *                     $00:NORMAL
 00131                      *                     $A0:WITHOUT ROM CASSETTE
 00132                      *                     $A1:FILE IS NOT FOUND
 00133                      *                     $A2:ALREADY OPEN
 00134                      *                     $A3:DIRECTRY NUMBER ERROR
 00135                      *                     $A4:ROM FORMAT ERROR
 00136                      *                     $A5:ADDRESSING ERROR
 00137                      *    (C)=0
 00138                      *    (Z) DEPEND ON VALUE OF (A)
 00139                      *    PACKET
 00140                      *        16-23: FOUND FILE NAME(WHEN 'ANSWER FILE NAME' MODE)
 00141                      *        24-31: FOUND FILE TYPE( ..)
 00142                      *  REGISTER PRESERVE
 00143                      *    NONE
 00144                      *
 00145                      *  WORK AREA AS REGISTER
 00146                      *   R0: SAVE PACKET ADDRESS
 00147                      *   R1H: SAVE MODE WHEN OPEN PROCEDURE WAS CALLED (VALUE OF (A))
 00148                      *   R1L: THE FLAG WHETHER FOUND FILE NAME IS MATCHED
 00149                      *        (BIT 7:STOP TO COMPARE     0:CONTINUE TO COMPARE 1:STOP)
 00150                      *        (BIT 0-4:FLAG FILE NAME IS MATCHED (0:MATCHED,  OTHERS:NO)
 00151                      *   R2H: READ CHARACTER (READ BYTE ROUTINE)
```

```
00152                              *    R2L: HEADER NUMBER
00153                              *
00154A 1000 97 52     A           OPNPRM STA A   R1H        * SAVE MODE 'ANSWER FILE NAME OR NOT'
00155A 1002 DF 50     A                  STX     R0         * SAVE PACKET ADDRESS
00156                              *
00157A 1004 BD 10EC   A                  JSR     PRMPON     * WITH ROM CARTRIDGE ? (RESET ADDRESS COUNTER)
00158A 1007 26 7B 1084*                  BNE     OPNP67     * IF NONEZERO, ERROR DETECT.
00159A 1009 97 55     A                  STA A   R2L        * HEADER NUMBER = 0
00160A 100B 86 81     A                  LDA A   #$81       * SET OPEN AND POWER ON FLAG
00161A 100D B7 0208   A                  STA A   PRMSTS
00162                              *
00163                              * READ HEADER AND SEARCH FILE NAME
00164A 1010 5F                    OPNP20 CLR B              * (B): DATA COUNTER (0 - $0F)
00165A 1011 D7 53     A                  STA B   R1L        * FLAG (NAME IS MATCHED)
00166A 1013 DE 50     A           OPNP25 LDX     R0         * (X): PACKET ADDRESS
00167A 1015 3A                           ABX
00168A 1016 BD 10C2   A                  JSR     PREDBY     * READ ONE CHARACTER FROM THE ROM
00169A 1019 25 72 108D*                  BCS     OPNP80
00170A 101B 5D                           TST B              * ADDRESS = FIRST COLUMN OF FILE NAME ?
00171A 101C 26 07 1025                   BNE     OPNP26
00172A 101E 81 FF     A                  CMP A   #$FF       * NOT FOUND ?   (LAST DIRECTRY MARK= $FF)
00173A 1020 27 6C 108E                   BEQ     OPNP90
00174A 1022 4D                           TST A              * DELETED ?  (DELETED FILE MARK = $00)
00175A 1023 27 2A 104F                   BEQ     OPNP35
00176A 1025 7D 0052   A           OPNP26 TST     R1H        * 'ANSWER FILE NAME' MODE ?
00177A 1028 27 02 102C                   BEQ     OPNP27
00178A 102A A7 10     A                  STA A   16,X       * YES, STORE FILE NAME TO DATA PACKET.
00179A 102C 7D 0053   A           OPNP27 TST     R1L        * STOP TO COMPARE (FILE NAME IS MATCHED) ?
00180A 102F 2B 14 1045                   BMI     OPNP29
00181A 1031 36                           PSH A              * '*': MARK TO STOP TO COMPARE.
00182A 1032 86 2A     A                  LDA A   #'*
00183A 1034 A1 00     A                  CMP A   0,X
00184A 1036 32                           PUL A
00185A 1037 26 05 103E                   BNE     OPNP28
00186A 1039 72 8053   A                  OIM     #$80,R1L   * '*' MARK. SET 'STOP COMPARE' BIT
00187A 103C 20 07 1045                   BRA     OPNP29
00188                              *
00189A 103E A1 00     A           OPNP28 CMP A   0,X        * COPARE FILE NAME.
00190A 1040 27 03 1045                   BEQ     OPNP29
00191A 1042 7C 0053   A                  INC     R1L        * SET 'FILE NOT MATCHED' FLAG
00192                              *
00193A 1045 5C                    OPNP29 INC B              * FINISH TO COMPARE ?
00194A 1046 C1 10     A                  CMP B   #16        * FILE NAME AND FILE TYPE HAVE 16 BYTES LENGTH
00195A 1048 26 C9 1013                   BNE     OPNP25
00196                              * FILE NAME AND FILE TYPE ARE COPLETED TO COMPARE.
00197A 104A 7B 0F53   A                  TIM     #$F,R1L    * OK ?
00198A 104D 27 12 1061                   BEQ     OPNP50     *
00199                              * NO, COMPARE NEXT HEADER
00200A 104F 7C 0055   A           OPNP35 INC     R2L        * R2L: HEADER NUMBER (NEXT)
00201A 1052 D6 55     A                  LDA B   R2L        * ADDRESS OF HEADER = '32' * 'HEADER NUMBER'
00202A 1054 C1 40     A                  CMP B   #64        *
00203A 1056 2A 36 108E                   BPL     OPNP90     * LIMIT OF THE HEADER ($000 - $3FF)
00204A 1058 86 20     A                  LDA A   #32
00205A 105A 3D                           MUL
00206A 105B 18                           XGDX               *  (X) : NEXT ADDRESSING POINTER
```

```
    00207                          *
    00208A 105C BD 1155  A                JSR    ADSTEP    * SET ADDRESSING COUNTER TO FIRST COLUMN OF
    00209A 105F 20 AF 1010                BRA    OPNP20    *  THE HEADER.
    00210                          *
    00211                          * TOP ADDRESS AND LAST ADDRESS WHICH ARE SHOWN BY ASCII CODE ARE
    00212                          *  CONVERTED TO BINARY VALUE.
    00213A 1061 CE 0204  A         OPNP50 LDX    #PRWKTP-4
    00214A 1064 8D 5C 10C2         OPNP65 BSR    PREDBY    * (A,B) <--- ASCII CODED HEXADECIMAL VALUE.
    00215A 1066 36                        PSH A
    00216A 1067 8D 59 10C2                BSR    PREDBY
    00217A 1069 16                        TAB
    00218A 106A 32                        PUL A
    00219A 106B BD FF2B  A                JSR    HEXBIN    * CONVERT HEX TO BINARY.
    00220A 106E 26 15 1085                BNE    OPNP70    * ERROR ?
    00221A 1070 A7 07    A                STA A  FTADRS-PRWKTP+4,X
    00222A 1072 08                        INX
    00223A 1073 8C 0208  A                CPX    #PRWKTP
    00224A 1076 26 EC 1064                BNE    OPNP65
    00225                          *
    00226A 1078 EC 05    A                LDD    EDADRS-PRWKTP,X * 'EDADRS' <--- LAST ADDRESS
    00227A 107A A3 03    A                SUBD   FTADRS-PRWKTP,X * 'FTADRS' <--- TOP ADDRESS
    00228A 107C FD 012E  A                STD    FILBYT    * 'FILBYT' <---- DATA NUMBER IN THE FILE.
    00229                          *
    00230A 107F 86 81    A                LDA A  #$81      * SET 'OPENED FILE' FLAG.
    00231A 1081 A7 00    A                STA A  PRMSTS-PRWKTP,X
    00232A 1083 4F                        CLR A
    00233A 1084 39                 OPNP67 RTS
    00234                          *
    00235A 1085 86 A4    A         OPNP70 LDA A  #$A4      * FORMAT ERROR
    00236A 1087 36                 OPNP75 PSH A
    00237A 1088 BD 113D  A                JSR    CLSPRM    * ERROR CLOSE
    00238A 108B 32                        PUL A
    00239A 108C 16                        TAB              * SET (Z), (N)
    00240A 108D 39                 OPNP80 RTS
    00241                          *
    00242A 108E 86 A1    A         OPNP90 LDA A  #$A1      * RETURN CODE (FILE WAS NOT FOUND)
    00243A 1090 20 F5 1087                BRA    OPNP75
    00244                          *
    00245                          *
    00246                          *  FUNCTION :  READ ONE CHARACTER FROM ROM FILE
    00247                          *  ON ENTRY
    00248                          *    NONE PARAMETER
    00249                          *  ON EXIT
    00250                          *    (A)=READ DATA
    00251                          *    (B)=STATUS    $00:NORMAL  $01:END OF FILE
    00252                          *                  $A3:FILE NOT OPEN
    00253                          *    (C)=0
    00254                          *    (Z)=DEPEND ON VALUE OF (B)
    00255                          *  REGISTER PRESERVE
    00256                          *    (X)
    00257                          *
    00258A 1092 3C                 REDPRM PSHX             * SAVE (X)
    00259                          *
    00260A 1093 C6 A3    A                LDA B  #$A3      * PRESET ERROR CODE (FILE IS NOT OPEND)
    00261A 1095 B6 0208  A                LDA A  PRMSTS    * IS POWER ON ? (BIT0, BIT7 BOTH ON)
```

```
00262A 1098 2A 0D 10A7              BPL     REDP08
00263A 109A 47                      ASR A
00264A 109B 24 0A 10A7              BCC     REDP08
00265                        *
00266A 109D FC 020D  A       REDP05 LDD     EDADRS     * IS CURRENT ADDRESS BOTTOM IN THE FILE ?
00267A 10A0 B3 020B  A              SUBD    FTADRS
00268A 10A3 26 06 10AB              BNE     REDP10
00269                        *
00270A 10A5 C6 01    A              LDA B   #1         * EOF RETURN
00271A 10A7 4F               REDP08 CLR A
00272A 10A8 5D                      TST B              * SET (Z), (N), CLEAR (C).
00273A 10A9 38                      PULX
00274A 10AA 39                      RTS
00275                        * READ ON BYTES FROM FILE
00276             10AB   A   REDP10 EQU     *
00277A 10AB 18                      XGDX
00278A 10AC 09                      DEX
00279A 10AD FF 012E  A              STX     FILBYT     * SET 'REST DATA NUMBER IN THE FILE'
00280A 10B0 C6 A5    A              LDA B   #$A5       * PRESET 'ADDRESSING ERROR' FLAG
00281A 10B2 25 F3 10A7              BCS     REDP08
00282A 10B4 FE 020B  A              LDX     FTADRS     * NON ERROR,
00283A 10B7 3C                      PSHX
00284A 10B8 BD 1155  A              JSR     ADSTEP     * ROM ADDRESSING <---- +1 INCREMENT
00285A 10BB 38                      PULX
00286A 10BC 08                      INX
00287A 10BD FF 020B  A              STX     FTADRS     * ADDRESSING COUNTER <--- +1 INCREMENT
00288A 10C0 5F                      CLR B              * RETURN CODE = NORMAL
00289A 10C1 38                      PULX
00290                        *
00291                        *
00292                        *  ENTRY POINT 'READ NEXT ONE BYTE'
00293                        * ON ENTRY
00294                        *  PARAMETER : NONE
00295                        *  READ ONE BYTE AND INCREMENT ADDRESSING COUNTER
00296                        * ON EXIT
00297                        *  (A) READ CHARACTER
00298                        * REGISTER PRESERVE
00299                        *  (B), (X)
00300                        *  WORK AS REGISTER
00301                        *  R2H: COUNTER FOR 8 TIMES AND WORKAREA FOR READ DATA
00302                        *      R2H   C  BIT7                              BIT0
00303                        *                   0    0    0    0    0    0    0    1
00304                        *                             I
00305                        *                             I
00306                        *                             V
00307                        *              0    0    0    0    0    0    0    1    X
00308                        *                                                    X:READ BIT
00309                        *                             I
00310                        *                             I
00311                        *                             V
00312                        *              0    0    0    0    0    0    1    X    X
00313                        *                                                    X:READ BIT
00314                        *
00315                        *
00316A 10C2 37               PREDBY PSH B
```

```
    00317A 10C3 86 01     A                 LDA A   #$1        * 1: MARK FOR 8 TH TIME.
    00318A 10C5 97 54     A                 STA A   R2H
    00319A 10C7 5F                          CLR B
    00320A 10C8 C4 7F     A         REDP20  AND B   #$FF-$80   * BIT 7 LOW (D7)
    00321A 10CA 86 C0     A                 LDA A   #$C0       * BIT6,7 EFFECTIVE
    00322A 10CC BD FED4   A                 JSR     WRTP26     * CLOCK LOW (FIRST TIME: D6 LOW)
    00323A 10CF CA 80     A                 ORA B   #$80       * CLOCK HIGH (FIRST TIME :READ DATA
    00324A 10D1 BD FED4   A                 JSR     WRTP26     *                SECOND TIME:SHIFT DATA)
    00325                           *
    00326A 10D4 96 02     A                 LDA A   PORT1      * INPUT DATA (BIT7, BIT6 ,...)
    00327A 10D6 48                          ASL A
    00328A 10D7 79 0054   A                 ROL     R2H        * R2L:SHIFT ONE BIT WHICH WAS GET.
    00329A 10DA CA 40     A                 ORA B   #$40       * FOR D6:HIGH
    00330A 10DC 24 EA 10C8                  BCC     REDP20     * COMPLETE TO READ 8 BITS ?
    00331                           *
    00332A 10DE FC 0209   A                 LDD     STADRS     * ASSRESSING POINTER <--- +1 INCREMENT
    00333A 10E1 C3 0001   A                 ADDD    #1
    00334A 10E4 FD 0209   A                 STD     STADRS
    00335                           *
    00336A 10E7 96 54     A                 LDA A   R2H        * (A) <--- READ DATA
    00337A 10E9 33                          PUL B
    00338A 10EA 5D                          TST B              * CLEAR (C), SET (Z) FOR 'REDPRM' ROUTINE
    00339A 10EB 39                          RTS
    00340                           *
    00341                           *
    00342                           * POWER ON ROM
    00343                           *  PROCEDURE
    00344                           *  1:CHECK PLUGIN OPTION (ROM) ?
    00345                           *  2:CLEAR ADDRESSING COUNTER
    00346                           *  3:POWER ON
    00347                           * PARAMETER
    00348                           *   ON ENTRY    NONE
    00349                           *   ON EXIT
    00350                           *      (A):RETURN CODE    00:NORMAL  OTHERS:ERROR
    00351                           *      (C):I/O ERROR FLAG
    00352                           *      (Z):DEPEND ON VALUE OF (A)
    00353                           * REGISTER PRESERVE
    00354                           *       (X)
    00355                           *
    00356A 10EC BD FF2E   A         PRMPON JSR      CHKPLG     * CHECK PLUG-IN OPTION
    00357A 10EF 25 26 1117                 BCS      PRMP80
    00358A 10F1 16                         TAB
    00359A 10F2 26 23 1117                 BNE      PRMP80
    00360A 10F4 72 207C   A                OIM      #$20,SIOSTS * SLAVE ROM CASSETTE ON
    00361A 10F7 FD 0209   A                STD      STADRS     * ROM ADDRESS = 0    (A,B)=0
    00362A 10FA 86 C0     A                LDA A    #$C0
    00363A 10FC BD FED4   A                JSR      WRTP26     * SET D6,D7 LOW (COUNT, CLOCK)
    00364A 10FF 86 51     A                LDA A    #$51
    00365A 1101 BD FF19   A                JSR      SNSCOM     * SEND 'PROM ON COMMAND' TO SLAVE MCU.
    00366A 1104 25 11 1117                 BCS      PRMP80
    00367A 1106 3C                         PSHX
    00368A 1107 CE 0190   A                LDX      #400       * WAIT 2 M SEC
    00369A 110A 09                  PRMP20 DEX
    00370A 110B 26 FD 110A                 BNE      PRMP20
    00371A 110D FE 0208   A                LDX      PRMSTS     * SET POWER ON FLAG (ON BIT7)
```

```
00372A 1110 62 8000   A           OIM     #$80,0,X
00373A 1113 38                    PULX
00374A 1114 4F                    CLR A
00375A 1115 20 33 114A            BRA     CLSP10     * (JMP  CHKRS)
00376                      *
00377A 1117 86 A0     A    PRMP80 LDA A   #$A0       * WITHOUT ROM CASSETTE (ERROR)
00378A 1119 39                    RTS
00379                      *
00380                      *
00381                      *  FUNCTION :  READ DIRECTORY
00382                      *  ON ENTRY
00383                      *    (A):DIRECTORY NUMBER (FROM 0 TO 63)
00384                      *    (X):ADDRESS WHERE HEADER ARE STORED
00385                      *  ON EXIT
00386                      *    (A):RETURN CODE  $00: NORMAL
00387                      *                     $A0: WITHOUT ROM CASSETTE
00388                      *                     $A3: DIRECTRY NUMBER ERROR
00389                      *    (C)=0
00390                      *    (Z)=DEPEND ON VALUE OF (A)
00391                      *  REGISTER PRESERVE
00392                      *   NONE
00393                      *
00394A 111A 16             DIRPRM TAB                * SAVE DIRECTORY NUMBER
00395A 111B 86 A3     A           LDA A   #$A3       * (A) <--DIRECTORY ERROR FLAG   (PRESET)
00396A 111D C1 40     A           CMP B   #64        * IS DIRECTRY NO. LIMIT (00 - 63) OK ?
00397A 111F 24 29 114A            BCC     CLSP10
00398                      *
00399A 1121 DF 50     A           STX     R0         * SAVE ADDRESS OF DIRECTORY
00400                      *
00401A 1123 37                    PSH B
00402A 1124 8D C6 10EC            BSR     PRMPON     * POWER ON (CHECK PROM)
00403A 1126 33                    PUL B
00404A 1127 26 21 114A            BNE     CLSP10
00405                      *
00406A 1129 86 20     A           LDA A   #32        * CALCULATE HEADER ADDRESS (32 * 'NUMBER')
00407A 112B 3D                    MUL
00408A 112C 18                    XGDX
00409A 112D 8D 26 1155            BSR     ADSTEP     * SET ROM ADDRESS
00410A 112F C6 20     A           LDA B   #32
00411A 1131 DE 50     A           LDX     R0
00412A 1133 37             DIRP10 PSH B
00413A 1134 8D 8C 10C2*           BSR     PREDBY     * READ ONE CHARACTER
00414A 1136 A7 00     A           STA A   0,X
00415A 1138 08                    INX
00416A 1139 33                    PUL B
00417A 113A 5A                    DEC B
00418A 113B 26 F6 1133            BNE     DIRP10
00419                      *
00420                      *
00421                      *  FUNCTION :  CLOSE ROM CASSETTE
00422                      *  ON ENTRY
00423                      *    PARAMETER  NONE
00424                      *  ON EXIT
00425                      *    (C): I/O ERROR FLAG
00426                      *  REGISTER PRESERVE
```

```
ERR   SEQ    LOC  OBJECT        PROGRAM  ROMOPT     --- ROM CARTRIDGE INTERFACE ROUTINE ---

    00427                            *    (B),(X)
    00428                            *
    00429A 113D 7F 0208  A           CLSPRM CLR    PRMSTS    * SET ROM STATUS  'POWER OFF', 'CLOSED FILE'
    00430A 1140 86 52    A                  LDA A  #CMPROF   * SEND 'POWER OFF COMMAND' TO SLAVE MCU.
    00431A 1142 BD FF19  A                  JSR    SNSCOM
    00432A 1145 71 DF7C  A                  AIM    #$FF-$20,SIOSTS * SET FLAG ('ROM CASSETTE IS OFF')
    00433A 1148 86 00    A                  LDA A  #0        * (DO NOT CHANGE (C) BIT)
    00434A 114A 7E FF16  A           CLSP10 JMP    CHKRS     * RECOVER RS232 (OPEN TO READ RS232)
    00435                            *
    00436                            *
    00437                            *  FUNCTION :  SET PROM ADDRESS TO DESTINATED VALUE
    00438                            *  ON ENTRY
    00439                            *   (X)= TARGET ADDRESS
    00440                            *  ON EXIT
    00441                            *   (C): I/O ERROR FLAG
    00442                            *  REGISTER PRESERVE
    00443                            *   NONE
    00444                            *
    00445              114D  A       ADST00 EQU    *         * CASE OF (NEW ADDRRESS < CURRENT ADDRESS)
    00446A 114D 8D 9D 10EC                  BSR    PRMPON    * WITHOUT ROM ? (CLEAR ADDRESSING COUNTER)
    00447A 114F 26 20 1171                  BNE    ADST80    *  WITHOUT ?
    00448A 1151 5F                          CLR B            * IF ROM  (A):0
    00449A 1152 FD 0209  A                  STD    STADRS    * ROM ADDRESSING COUNTER <--- 0
    00450                            *
    00451                            *  ENTRY POINT OF 'ADSTEP' ROUTINE
    00452A 1155 3C                   ADSTEP PSHX             * (A,B)<---(X)
    00453A 1156 32                          PUL A
    00454A 1157 33                          PUL B
    00455A 1158 B3 0209  A                  SUBD   STADRS    * NEW ADDRESS >= CURRENT ADDRESS ?
    00456A 115B 27 14 1171                  BEQ    ADST80    * = ?
    00457A 115D 25 EE 114D                  BCS    ADST00
    00458                            *                         * CASE OF 'TARGET ADDRESS > CURRENT ADDRESS'
    00459A 115F FF 0209  A                  STX    STADRS    * SET NEW ADDRESS TO 'STADRS'
    00460A 1162 18                          XGDX             * (X)<--- STEP COUNT
    00461                            *
    00462A 1163 5F                   ADST30 CLR B            * COUNT UP ADDRESSING COUNTER
    00463A 1164 86 C0    A                  LDA A  #$C0
    00464A 1166 BD FED4  A                  JSR    WRTP26
    00465A 1169 C6 40    A                  LDA B  #$40
    00466A 116B BD FED4  A                  JSR    WRTP26
    00467A 116E 09                          DEX
    00468A 116F 26 F2 1163                  BNE    ADST30
    00469A 1171 39                   ADST80 RTS
    00470                            *
    00471              0000  A              END
***** TOTAL ERRORS      0
```

CHAPTER 9 LOAD MODULE

Page

9.1 General 9-1
9.2 Load Module (Machine Language) Format 9-2
9.3 Dump/Load Procedures 9-3
9.4 Binary Dump/Load Subroutine Table 9-4
9.5 Binary Dump/Load Work Area 9-7

APPENDIX - Sample Lists 9-8

9.1 General

The module format for output of data by the SAVEM command in BASIC or the W command in the Monitor is a special format called a "Binary Load Module format". One file is divided into a number of records each containing memory addresses and data (Fig. 9-1).

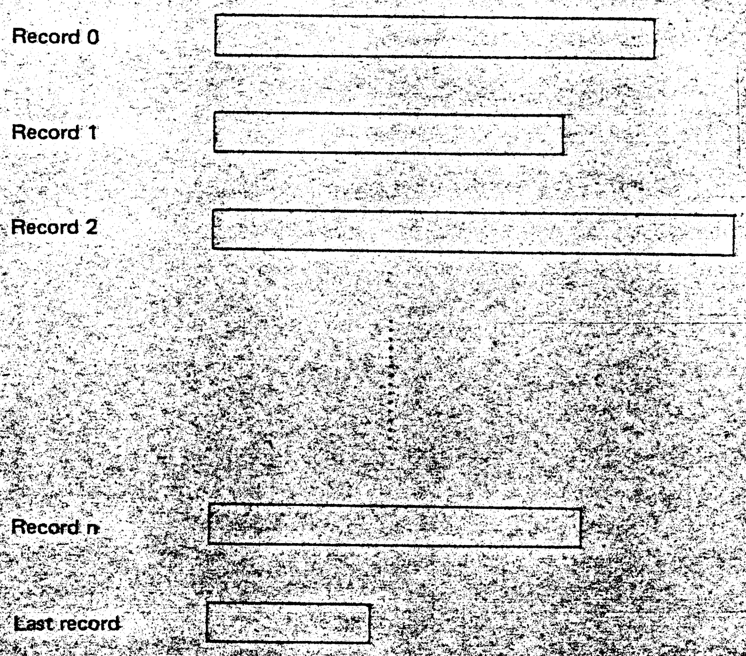


Fig. 9-1 Division of File into Records

Each record has a maximum length of 259 bytes and each data contained in the record is represented in binary numbers in units of one byte. The format of each record is shown below.

9.2 Load Module (Machine Language) Format

(1) Intermediate record

Column	Size (bytes)	Item	Description
0	1	Record length	Indicates the length of the data contained in the record in binary numbers (00 through FF).
1 ~ 2	2	Address	Indicates the address of the first data in the record in binary numbers 0000 through FFFF (in order of the upper and lower digits).
3	1	Data	Data 1. Namely, first data (00 through FF).
4	1	Data	Data 2
⸾			
n	1	Data	Data n (n must be a value in the range of 0 to 255.)
n + 1	1	Checksum	This value must be such that the low-order 8 bits of the sum of the data values in columns 0 through n + 1 become 0.

(2) Last record

Column	Size (bytes)	Item	Description
0	1	Record length	This value must always be 0.
1 ~ 2	2	Address	Indicates the entry point of a program in binary numbers (0000 through FFFF in order of the upper and lower digits).
3	1	Checksum	This value must be such that the low-order 8 bits of the sum of the data values in columns 0 through 3 become 0.

9.3 Dump/Load Procedures

9.3.1 I/O devices

The basic I/O routines support the following devices:

(1) Input
  - (a) External audio cassette
  - (b) Built-in microcassette
  - (c) ROM cartridge

(2) Output
  - (a) External audio cassette
  - (b) Built-in microcassette

9.3.2 Dump/load procedures

The memory contents in the binary load module format are transferred to and from an external storage as follows:

(1) Output to the external storage

(a) File opening

Subroutine "OPNDMP" is provided to open the specified file (device) for output. Subroutine "OPNWCS" is called if the specified file is an external audio cassette.

(b) Output of the memory contents

Subroutine "BIDUMP" is provided to output the memory contents in the binary load module format to the opened file and closes it upon completion of the dumping.

(2) Input from the external storage

(a) File opening

Subroutine "OPNLOD" is provided to open the specified file (device) for input. Subroutine "OPNPRM" is called if the specified file is a ROM cartridge.

(b) Loading into memory

Subroutine "BILOAD" is provided to store the input data in the binary load module format in the main memory and closes the file upon completion of the loading.

9.4 Binary Dump/Load Subroutine Table

Subroutine name	Entry point	Description
OPNDMP	FEEO	Binary memory dump open. This subroutine opens the file to be dumped in a binary absolute format and supports an external cassette and the built-in microcassette drive.
		Parameters: At Entry (X): Top address of a data packet (B): Device name 'M': Microcassette drive 'C': External audio cassette Packet 1. Interblock tape stop mode (1 byte) for external audio cassette or microcassette 00: Stop the tape between blocks. 01: Do not stop the tape between blocks. 2. Top address of buffer (2 bytes). The buffer size is 260 bytes. 3. Filename (8 bytes) 4. File type (8 bytes) 5. Dump start address (2 bytes) 6. Dump end address (2 bytes) 7. Offset value (2 bytes) 8. Program entry point (2 bytes) Note: The offset value is added to the dump start address, dump end address, or the program entry point as an unsigned binary number. Parameters: At Return (C): Abnormal I/O flag (A): Return code (This parameter is dependent on subroutines OPNNCS and OPNWMS.) Registers retained None Subroutines referenced OPNWMS, OPNWCS Variables to be used: R0, R1, R2, R3, R4, R5, R6, and R7
BIDUMP	FEDD	Binary memory dump. This subroutine dumps the memory contents in a binary absolute format to the file opened by subroutine OPNDMP and closes the file upon completion of the dumping.
		Parameters: At Entry None

		At Return Depend on subroutines WRTCS, WRTMS. Registers retained None Subroutines referenced WRTCS, WRTMS Variables used R0, R1, R2, R3, R4, R5, R6 and R7
OPNLOD	FEDA	Binary memory load. This subroutine opens the file to be loaded and loads the contents of the file dumped in binary absolute format, into memory.
		Parameters: At Entry (X): Top address of a data packet (B): Device name 'M': Microcassette 'C': External audio cassette 'P': ROM cartridge (A): Specifies whether or not the filename is to be returned. 00: Return the filename. 01: Do not return the filename. Packet 1. Interblock tape stop mode (1 byte) for external audio cassette or microcassette. 00: Stop the tape between blocks. 01: Do not stop the tape between blocks. FF: Depends on the header. 2. Top address of buffer (2 bytes) The buffer size is 260 bytes. 3. Filename (8 bytes) 4. File type (8 bytes) 5. Lower limit of the memory address to be loaded (2 bytes) 6. Upper limit of the memory address to be loaded (2 bytes) 7. Offset value (2 bytes) Note: The offset value is added to the address information of a file (load start address, load end address, or program entry point) as an unsigned binary number. The interblock tape stop mode is effective only for 'M' or 'C' but not for 'P'. If the return of a filename is specified by register (A), the filename is returned after the 19th byte of the packet. (In this case, the packet contents after the lower limit of the memory address are destroyed.) Since subroutines OPNRCS, SRCRCS, OPNRMS, SRCRMS, and OPNPRM are actually called, the packet depends on these subroutines.

		At Return (C): Abnormal I/O FLAG (A): Return codes 8C: Load area exceeds the specified memory space range. Other return codes depend on OPNRCS, SRCRCS, OPNRMS, SRCRMS and OPNPRM Registers retained None Subroutines referenced OPNRCS, SRCRCS, OPNRMS, SRCRMS, and OPNPRM Variables used R0, R1, R2, R3, R4, R5, R6, and R7
		Notes: Assuming that the upper- and lower-limit values of the memory addresses that can be loaded by the packet are μ and ℓ, respectively and that the address of a data to be loaded is α, the data can be loaded only when the following condition is satisfied. $\ell \leq \alpha \leq \mu$ If the address is not within this range, return code 8C (load area error) is output to interrupt the loading operation by force. The file is closed upon completion of the loading.
BILOAD	FED7	This subroutine loads the contents of the file opened by subroutine "OPNLOD" into the memory and closes the file upon completion of the loading.
		PaRAMETERS: At Entry (A): Specifies whether or not the contents of the file are to be loaded into the memory. 00: Load the contents of the file into the memory. 01: Check the load module format only. Do not load the file contents. At Return (X): Program entry point The value specified by the offset value is added to the value of the entry point recorded in the file. (A): Return codes 00: Normal 8C: Load area exceeds the specified memory space range. Others: Depend on the return codes of a file input routine. Registers retained None Subroutines referenced READMS, READCS, REDPRM Variables used R0, R1, R2, R3, R4, R5, R6 and R7

9.5 Binary Dump/Load Work Area

Address (From)	 (To)	Variable name	Bytes	Description
20Ø F	21Ø	DLTPAD	2	First dump address
211	212	DLBTAD	2	Last dump address
213	214	DLOFAD	2	Offset value
215	216	DLSTAD	2	Program entry point
217	217	DLDVID	1	Dump/load device
218	218	DLSTS	1	Status work area (dummy)
219	21A	DLDVIX	2	Table address of a dump/load routine

```
      *** 6301 CROSS MACROASSEMBLER   VER1.0 ***                         10/21/82    09:43:29
ERR   SEQ    LOC  OBJECT       PROGRAM  CLOCK       --- CLOCK SAMPLE PROGRAM ---

     00001                                    NAM    CLOCK
     00002                             *
     00003                             * DISPLAY CURRENT TIME ON THE PHISICAL SCREEN.
     00004                             *  MPU IS SLEEP IF CLOCK UPDATE IS NOT CAUSED.
     00005                             *
     00006                             * FILE NAME     'EX$B'   BY  K.A
     00007                                    TTL    --- CLOCK SAMPLE PROGRAM ---
     00008                                    OPT    PAGE=55
     00009                                    OPT    LOAD
     00010A 1000                              ORG    $1000
     00011                             *
     00012                             * SUBROUTINE ENTRY POINT
     00013            FFA9   A         SLEEP  EQU    $FFA9     * SLEEP CPU
     00014            FF4C   A         DSPLCH EQU    $FF4C     * DISPLAY ONE CHARACTER ON THE PHISICAL SCREEN
     00015            FF49   A         DSPLCN EQU    $FF49     * DISPLAY SOME CHARACTERS ON THE PHISICAL SCREE
     00016                             *
     00017                             *
     00018A 1000                              ORG    $1000
     00019                             *
     00020A 1000 C6 00       A                LDA B  #0        * CLEAR SCREEN
     00021A 1002 BD FF49     A                JSR    DSPLCN
     00022A 1005 86 FF       A                LDA A  #$FF      * ALARM INTERRUPT TIME
     00023A 1007 97 41       A                STA A  $41       *  = ANY TIME WHEN SECOND IS UPDATED.
     00024A 1009 97 43       A                STA A  $43
     00025A 100B 97 45       A                STA A  $45
     00026                             *
     00027A 100D 72 204B     A         CLCK10 OIM    #$20,$4B  * ENABLE ALARM INTERRUPT.
     00028A 1010 BD FFA9     A                JSR    SLEEP     * MCU IS SLEEP FOR SAVE POWER.
     00029A 1013 96 44       A                LDA A  $44       * LOAD 'HOUR'
     00030A 1015 16                           TAB              *  DISPLAY 'HOUR'
     00031A 1016 84 F0       A                AND A  #$F0      *  (HIGH ORDER)
     00032A 1018 47                           ASR A
     00033A 1019 47                           ASR A
     00034A 101A 47                           ASR A
     00035A 101B 47                           ASR A
     00036A 101C 8A 30       A                ORA A  #'0
     00037A 101E CE 0502     A                LDX    #$0502
     00038A 1021 37                           PSH B
     00039A 1022 BD FF4C     A                JSR    DSPLCH
     00040A 1025 32                           PUL A            * DISPLAY (LOW ORDER)
     00041A 1026 84 0F       A                AND A  #$0F
     00042A 1028 8A 30       A                ORA A  #'0
     00043A 102A BD FF4C     A                JSR    DSPLCH
     00044A 102D 86 3A       A                LDA A  #':       *
     00045A 102F BD FF4C     A                JSR    DSPLCH
     00046A 1032 96 42       A                LDA A  $42       * LOAD 'MINUTE'
     00047A 1034 16                           TAB              *  DISPLAY 'MINUTE'
     00048A 1035 84 F0       A                AND A  #$F0      *  (HIGH ORDER)
     00049A 1037 47                           ASR A
     00050A 1038 47                           ASR A
     00051A 1039 47                           ASR A
     00052A 103A 47                           ASR A
     00053A 103B 8A 30       A                ORA A  #'0
     00054A 103D CE 0802     A                LDX    #$0802
     00055A 1040 37                           PSH B
```

```
ERR   SEQ    LOC  OBJECT        PROGRAM  CLOCK       --- CLOCK SAMPLE PROGRAM ---

     00056A 1041 BD FF4C   A               JSR     DSPLCH
     00057A 1044 32                        PUL A               * DISPLAY (LOW ORDER)
     00058A 1045 84 0F      A              AND A   #$0F
     00059A 1047 8A 30      A              ORA A   #'0
     00060A 1049 BD FF4C   A               JSR     DSPLCH
     00061A 104C 86 3A      A              LDA A   #':         *
     00062A 104E BD FF4C   A               JSR     DSPLCH
     00063A 1051 96 40      A              LDA A   $40         * LOAD 'SECOND'
     00064A 1053 16                        TAB                 *  DISPLAY 'SECOND'
     00065A 1054 84 F0      A              AND A   #$F0        *  (HIGH ORDER)
     00066A 1056 47                        ASR A
     00067A 1057 47                        ASR A
     00068A 1058 47                        ASR A
     00069A 1059 47                        ASR A
     00070A 105A 8A 30      A              ORA A   #'0
     00071A 105C CE 0802   A               LDX     #$0802
     00072A 105F 37                        PSH B
     00073A 1060 BD FF4C   A               JSR     DSPLCH
     00074A 1063 32                        PUL A               * DISPLAY (LOW ORDER)
     00075A 1064 84 0F      A              AND A   #$0F
     00076A 1066 8A 30      A              ORA A   #'0
     00077A 1068 BD FF4C   A               JSR     DSPLCH
     00078A 106B 20 A0 100D                BRA     CLCK10-
     00079                           *
     00080                0000  A          END
***** TOTAL ERRORS    0

  ---- BINARY DUMP FORMAT OF OBJECT CODE ----

     13 10 00 C6 00 BD FF 49 86 FF 97 41 97 43 97 45 72 20 48 BD FF A9 BD
     13 10 13 96 44 16 84 F0 47 47 47 47 8A 30 CE 05 02 37 BD FF 4C 32 4A
     14 10 26 84 0F 8A 30 BD FF 4C 86 3A BD FF 4C 96 42 16 84 F0 47 47 47 62
     14 10 3A 47 8A 30 CE 08 02 37 BD FF 4C 32 84 0F 8A 30 BD FF 4C 86 3A 43
     12 10 4E BD FF 4C 96 40 16 84 F0 47 47 47 47 8A 30 CE 0B 02 37 40
     0D 10 60 BD FF 4C 32 84 0F 8A 30 BD FF 4C 20 A0 34
     00 10 00 F0
```

CHAPTER 10 FLOPPY DISK UNIT

Page

10.1 General 10-1
10.2 Disk Format 10-1
10.3 System Disk and Non-system Disk 10-2
10.4 Interface with DISK BASIC 10-3
10.5 EPSP (EPSON Serial Communication Protocol) 10-10
10.6 Function Table of Floppy Disk Unit 10-13

CHAPTER 10 FLOPPY DISK UNIT

10.1 General

The TF-20 Terminal Floppy is an intelligent floppy disk unit which is connected to the HX-20 through a serial communication interface and transfers the data stored in a floppy disk to the HX-20 according to the commands received from the HX-20.

When the TF-20 is connected to the HX-20, the DBASIC.SYS (Disk BASIC System, which is an extended portion of BASIC) is loaded from the floppy disk into the RAM of the HX-20 upon start of BASIC. The DBASIC.SYS loaded into the RAM operates together with the interpreter on the ROM until control is returned to the MENU again. It processes the data input/output to and from the floppy disk and newly added commands, statements and functions. The interpreter on the ROM handles the conventional functions of the HX-20.

In DISK BASIC, a maximum of two TF-20 units can be connected to the HX-20. The first TF-20 unit is used as disk drives "A:" and "B:" and the second unit as disk drives "C:" and "D:". To distinguish between the first and second units, the DIP switch located in the TF-20 must be used. The 4-pin DIP switch (bits 1 to 4) of the TF-20 is factory-set to all "ON" for drives "A:" and "B:". When connecting a second TF-20 unit to the HX-20, the DIP switch setting of the second unit must be changed to "bits 1, 2, 3, 4 = ON, ON, ON, OFF" to indicate that the unit is used as drives "C:" and "D:".

Daisy-chaining method is used to interconnect an HX-20 and a TF-20 or two TF-20 units via cable set #707 (for daisy chaning). TF-20 (disk 1) and TF-20 (disk 2) can be interconnected in any order. Fig. 10-1 shows how two TF-20 units are connected to the HX-20.

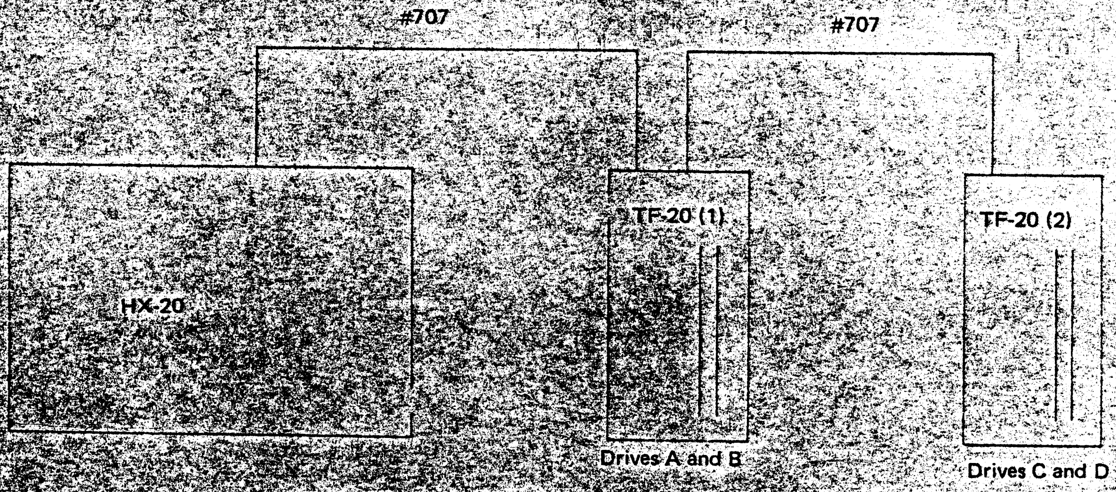


Fig. 10-1 Interconnection of HX-20 and Two TF-20 Units

10.2 Disk Format

Disk type:	Double-sided, double density (MFM)
No. of tracks:	80 tracks (40 tracks x 2 sides)
Track density:	48 TPI
No. of sectors:	16 sectors/track
Capacity per sector:	256 bytes

Total disk capacity:	320K bytes (256 x 16 x 80)
Access time between tracks:	15 ms

Tracks and sectors are logically structured as shown below.

No. of tracks:	40 tracks (0 to 39)
No. of sectors:	64 sectors/track (1 to 64)
Capacity per sector:	128 bytes

Table 10.1 shows the relationship between the physical and logical specifications.

Table 10.1 Relationship between Physical and Logical Specifications

	Physical specifications	Logical specifications
Track	One track on one side + one track on the other side	One track
Sector	One sector (256 bytes)	Two sectors (128 bytes x 2)

10.3 System Disk and Non-system Disk

The floppy disks used in DISK BASIC can be divided into a system disk and a non-system disk. Either of these two disks must be initialized by the physical format of the TF-20 for generation of correct directories.

All the floppy disks supplied by EPSON have been initialized before shipment so that they can be used as non-system disks. Floppy disks other than those supplied by EPSON and those disks in which a read or write error has occurred must be initialized by the FRMAT command. The system disk refers to the disk which contains a system program for DISK BASIC, and must be inserted into drive "A:" when DISK BASIC is to be booted. The system disk is mapped as follows.

Track 0	Sectors 1 and 2:	Cold-start loader (loads a system contained in the system disk into the memory of the TF-20.)
	Sectors 3 to 18:	Unused
	Sectors 19 to 46:	BDOS (Basic Disk Operating System)
	Sectors 47 to 64:	BIOS (Basic Input/Output System) for the HX-20
Track 1	Sectors 1 to 42:	TFDOS (communication program with the HX-20)
	Sectors 43 to 64:	Unused
Tracks 2 and 3	Sectors 1 to 64:	Unused
Track 4	Sectors 1 to 16:	Directory area (for 64 directories max.)
	Sectors 17 to 64:	File area
Tracks 5 to 38	Sectors 1 to 64:	File area (278K bytes max.)

Two files "BOOT80.SYS" and "DBAISC.SYS" are secured for the system in the system disk. Since these files are write-protected, their filenames are not displayed even by executing the FILES command. Note that the user cannot use the same filenames as these two files. To duplicate a system disk, either copy all the contents of the existing system disk to a new floppy disk by COPY utility, or execute the SYSGEN command for a non-system disk.
"SYSGEN" copies not only the system area of the disk but also copies the system file whose file type is "SYS".

10.4 Interface with DISK BASIC

The DISK BASIC is broadly divided into the following 3 modules:
(1) BASIC interpreter (ROM version: HX-20 side)
(2) DBASIC interpreter (DBASIC.SYS: HX-20 side)
This interpreter is an extended portion of BASIC which is loaded from a disk to the RAM of the HX-20 upon start of the BASIC and handles the data input/output to and from the disk and the processing of commands and statements, together with the BASIC interpreter described in (1) above. This module consists mainly of a portion connected to the BASIC interpreter (i.e., a BASIC driver) and a portion interfacing with the TFDOS (i.e., EPSP driver).
(3) TFDOS (TF-20 side)
The TFDOS which is resident on the RAM of the TF-20 receives commands from the HX-20, opens and reads or writes files using the BDOS or the BIOS for the HX-20, and returns data and error codes to the HX-20.

Of the above 3 modules, the BASIC driver and EPSP driver of the DBASIC interpreter are interfaced with each other through the BSCINT (BASIC interface), while the EPSP driver is interfaced with the TFDOS through the EPSP (EPSON Serial Communication Protocol) as shown in Fig. 10-2.

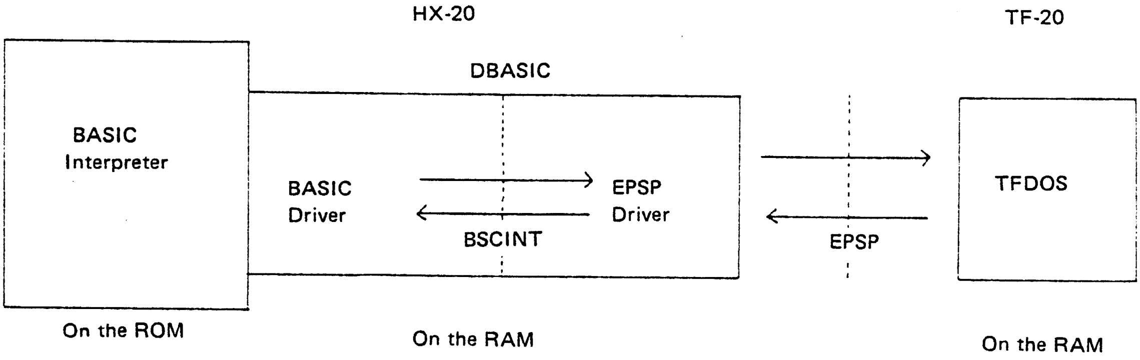


Fig. 10-2 Software Configuration of Disk BASIC

1Ø.4.1 BASIC interface (BSCINT)

1Ø.4.1.1 Functions of BSCINT

Interfacing of DBASIC with BASIC is supported by subroutine "BSCINT" (BASIC Interface) which has the following functions:

(1) File open
(2) File close
(3) Random read (128 bytes)
(4) Random write (128 bytes)
(5) File delete
(6) File rename
(7) File size calculation
(8) First directory search
(9) Next directory search
(1Ø) Direct write into disk (DSKO$, 128 bytes)
(11) Disk formatting (FRMAT)
(12) Disk system reset (RESET)
(13) System disk generation (SYSGEN)
(14) Disk free area calculation (DSKF)
(15) Direct read from disk (128 bytes)
(16) Disk all copy

1Ø.4.1.2 Subroutine call procedure

Subroutine "BSCINT" is called as follows:

(1) Setting the entry point for BSCINT
The contents at an address 3 bytes from addresses (ØA3E and ØA3F) are "JMP BSCINT" (see Fig. 1Ø.3). This means that the address specified by addresses (ØA3E and ØA3F) is the entry point of the subroutine that includes BSCINT error processing.

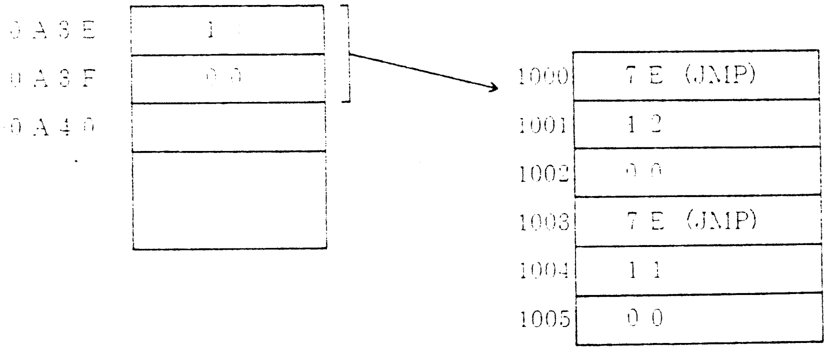


Fig. 10-3 BSCINT Entry Point

(2) Creation of a parameter packet
Parameters are created on memory, and are given in the order of the function code, return code, and data (see Fig. 1Ø-4). The data string has a length of one or more bytes. For details of the functions and parameters, refer to the BSCINT parameter packet table.

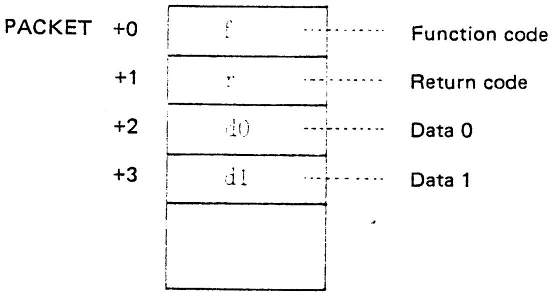


Fig. 10-4 Parameter Packet of Subroutine "BSCINT"

(3) Subroutine call

The first address of the parameter packet is set in the index register to call subroutine "BSCINT".

An example of opening a file is shown below:

[Example] File under the file descriptor "ABC.BAS" is opened in Sequential Output mode using drive "A".

```
          LDAA  =$7E      * (JMP instruction )
          STAA  BSENTR
          LDD   $A3E
          STD   BSENTR+1
          LDX   =CPOPC
          JSR   BSENTR
          LDAA  1,X
          BNE   ERROR
          RTS
           :
ERROR     EQU   *         * error procedure
           :
BSENTR    FCB   $7E       * (JMP BSCINT)
          RMB   2
CPOPC     FCB   $00
          FCB   $00
          FCB   $00
          FCC   /ABC△△△△△/
          FCC   /BAS/
```

10.4.2 BSCINT parameter packet table
*All packet data numbers are decimal numbers.

No.	Function	Packet data No.	Description
1	File open		Opens the file in the specified drive according to the filename, file type, and file mode.
		00	00 (function code)
		01	Return code (set at return)
		02	File number (set at return)
		03	Drive number ("A", "B", "C" or "D")
		04 ~ 11	Filename (8 characters. If the filename is less than 8 characters, left-justify the filename and fill blank code(s) (20) in the remaining space.)
		12 ~ 14	File type (3 characters. If the file type is less than 3 characters, left-justify the file type and fill blank codes (20) in the remaining space.)
		15	Modes 10_{16}: Sequential input (M.SQI) 30_{16}: Sequential output (M.SQO) 40_{16}: Random access (M.RND) If no file exists in M.SQI or M.SQO mode, a new file is created. If no file exists in M.SQI mode, it is assumed that an error has occurred. If a file exists in M.SQO mode, the previous file will be deleted.
2	File close		Closes the specified opened file.
		00	01 (function code)
		01	Return code (set at return)
		02	File number (i.e., the number returned at a file open)
3	Random read		Reads the specified record of a file. (One record consists of 128 bytes.)
		00	02 (function code)
		01	Return code (set at return)
		02	File number (i.e., the number returned at a file open)
		03 ~ 04	Record number (binary value in the range of 1 to 65535. Must be entered in the order of high- and low-order bytes.)
		05 ~ 06	Buffer address (must be entered in the order of high- and low-order bytes.)
4	Random write		Writes the specified record of a file. (One record consists of 128 bytes.)
		00	03 (function code)
		01	Return code (set at return)
		02	File number (i.e., the number returned at a file open)
		03 ~ 04	Record number (binary value in the range of 1 to 65535. Must be entered in the order of high- and low-order bytes.)
		05 ~ 06	Buffer address (must be entered in the order of high- and low-order bytes.)

No.	Function	Packet data No.	Description
5	File delete		Deletes the specified file.
		00	04 (function code)
		01	Return code (set at return)
		02	Unused
		03	Drive name ("A", "B", "C" or "D")
		04 ~ 11	Filename (8 characters. If the filename is less than 8 characters, left-justify the filename and fill blank codes (20) in the remaining space.)
		12 ~ 14	File type (3 characters. If the file type is less than 3 characters, left-justify the file type and fill blank codes (20) in the remaining space.)
6	File rename		Rename the existing file.
		00	05 (function code)
		01	Return code (set at return)
		02	Unused
		03	Drive name ("A", "B", "C" or "D")
		04 ~ 11	Filename before change (8 characters)
		12 ~ 14	File type before change (3 characters)
		15 ~ 22	Filename after change (8 characters)
		23 ~ 25	File type after change (3 characters)
7	File size calculation		Returns the number of records of the specified file. (One record consists of 128 bytes.)
		00	06 (function code)
		01	Return code (set at return)
		02	File number (i.e., the number returned at a file open)
		03 ~ 04	Maximum value of a record number (the value must be in the range of 0 to 65535. 0 indicates the null state.)
8	First directory search		Returns the FCB (file control block) address and directory code on the disk of the file for which the filename and file type were specified. If the filename and file type are all specified by character '?', it is assumed that file matching has been completed for all files.
		00	07 (function code)
		01	Return code (set at return)
		02	Unused
		03	Drive name ("A", "B", "C" or "D")
		04 ~ 11	Filename (8 characters)
		12 ~ 14	File type (3 characters)
		15	Directory code (set at return)
		16 ~ 47	Directory FCB (set at return)

No.	Function	Packet data No.	Description
9	Next directory search		Searches the next directory. (This function is performed next to the function No. 8 above.) The method of specifying the filename and file type is the same as function No. 8.
		00	08 (function code)
		01	Return code (set at return)
		02	Unused
		03	Drive name ("A", "B", "C" or "D")
		04 ~ 11	Filename (8 characters)
		12 ~ 14	File type (3 characters)
		15	Directory code (set at return)
		16 ~ 47	Directory FCB (set at return)
10	Direct write into disk (DSK0$)		Writes data into the specified tracks and sectors of floppy disk.
		00	09 (function code)
		01	Return code (set at return)
		02	Unused
		03	Drive name ("A", "B", "C" or "D")
		04	Track number (binary value in the range of 0 to 39_{10})
		05	Sector number (binary value in the range of 1 to 64_{10})
		06 ~ 07	Buffer address (must be entered in the order of high- and low-order bytes)
11	Disk formatting (FRMAT)		Formats the floppy disk in the specified drive.
		00	0A (function code)
		01	Return code (set at return)
		02	Unused
		03	Drive name ("A", "B", "C" or "D")
12	Disk system reset		Enables a disk replacement. When the disk system is reset, all the disks can be read or written and disk drive "A" is selected.
		00	0B (function code)
		01	Return code (set at return)
		02	Unused
		03	Drive name ("A", "B", "C" or "D")
13	System disk generation (SYSGEN)		Copies the system area and file of the system disk set in drive "A", to the disk set in drive "B". After copying, the disk in drive "B" can be used as a system disk.
		00	0C (function code)
		01	Return code (set at return)

No.	Function	Packet data No.	Description
14	Disk free area calculation (DSKF)		Provides the free area size of the disk in the specified drive in 2K-byte units.
		ØØ	ØD (function code)
		Ø1	Return code (set at return)
		Ø2	Unused
		Ø3	Drive name ("A", "B", "C" or "D")
		Ø4	Free area size (binary value in 2K-byte units set at return)
15	Direct read from disk (DSKI$)		Reads data from the specified tracks and sectors of a floppy disk.
		ØØ	ØE (function code)
		Ø1	Return code (set at return)
		Ø2	Unused
		Ø3	Drive name ("A", "B", "C" or "D")
		Ø4	Track number (binary value in the range of Ø to 39_{10})
		Ø5	Sector number (binary value in the range of 1 to 64_{10})
		Ø6 ~ 77	Buffer address (must be entered in the order of high- and low-order bytes. In this case, however, the message work area of EPSP driver routine is used.)
16	Disk all copy		Copies all the contents of the floppy disk in the specified drive to the disk in the other drive of the same floppy disk unit. (i.e., from "A" to "B", from "C" to "D")
		ØØ	ØF (function code)
		Ø1	Return code (set at return)
		Ø2	Unused
		Ø3	Drive name ("A", "B", "C" or "D") NOTE: With drives "A" and "B", disk copying must be from "A" to "B". With drives "C" and "D", disk copying must be from "C" to "D".

1Ø.4.3 BSCINT return codes

Code (Hex)	Meaning
ØØ	Normal completion of operation
Ø1	The specified file is not found.
Ø2	End of File (EOF) was detected during file input.
Ø3	The file already exists.
Ø4	The specified device is not found.
Ø5	No directory area exists.
Ø6	No disk area exists.
Ø7	The specified record number is incorrect.
Ø8	The disk is write-protected.
Ø9	The file is not opened.
ØA	The specified file number is incorrect.
ØB	The specified file mode is incorrect.
ØC	The specified file is already open.
ØD	The number of opened files is too many.
ØE	The specified file descriptor is incorrect.
ØF	An error has occurred during a read operation.
1Ø	An error has occurred during a write operation.

1Ø.5 EPSP (EPSON Serial Communication Protocol)

1Ø.5.1 EPSP functions

The EPSP is an interface between the EPSP driver and the TFDOS as described in Chapter 4. The EPSP on the TF-2Ø side has the following functions:

(1) Disk system reset
    Corresponds to Item (12) of paragraph 1Ø.4.1.

(2) File open
    Corresponds to Item (1) of paragraph 1Ø.4.1.

(3) File close
    Corresponds to Item (2) of paragraph 1Ø.4.1.

(4) First directory search
Corresponds to Item (8) of paragraph 10.4.1.

(5) Next directory search
Corresponds to Item (9) of paragraph 10.4.1.

(6) File delete
Corresponds to Item (5) of paragraph 10.4.1.

(7) File creation
By this function, the directory and memory are initialized and a file empty of data is created.

(8) Random read
Corresponds to Item (3) of paragraph 10.4.1.

(9) Random write
Corresponds to Item (4) of paragraph 10.4.1.

(10) File size calculation
Corresponds to Item (7) of paragraph 10.4.1.

(11) Disk all copy
Corresponds to Item (16) of paragraph 10.4.1.

(12) Direct write (128 bytes) into disk (DSKO$)
Corresponds to Item (10) of paragraph 10.4.1.

(13) Disk formatting (FRMAT)
Corresponds to Item (11) of paragraph 10.4.1.

(14) System disk generation (SYSGEN)
Corresponds to Item (13) of paragraph 10.4.1.

(15) Disk free area calculation (DSKF)
Corresponds to Item (14) of paragraph 10.4.1.

(16) Direct read (128 bytes) from disk (DSKI$)
Corresponds to Item (15) of paragraph 10.4.1.

(17) Disk boot
By this function, file "BOOT80.SYS" is booted to the HX-20 from the system disk in the disk drive A of the TF-20. In other words, this function opens file "BOOT80.SYS", reads 128 bytes of data only and transfers them to the HX-20.

(18) Load open
By this function, file "DBASIC.SYS" contained in the system disk in the drive A of the TF-20 is opened and then loaded into the RAM of the TF-20. After the loading, the file is relocated on the RAM of the TF-20 using a relocatable flag (one of the load open parameters) and an ending or starting address. Return code "FF" if the corresponding file is not found, or return code "00" if found, is returned to the HX-20 together with the file size of "DBASIC.SYS".

(19) Load close
This function indicates that the transfer of file "DBASIC.SYS" has been completed. In this case, the TF-20 does not perform any function.

(20) Read one block
By this function, the file "DBASIC.SYS" opned, read, and relocated in item (18) above is transferred to the HX-20 in units of 128 bytes.
Return code "FF" indicates the end of file (EOF).

10.5.2 Subroutne "OUTSRL"

Subroutine "OUTSRL" handles the data transmission/reception of EPSP as follows:

(1) Creation of a parameter packet

Parameters are given in the form of a packet as shown in Fig. 10-5.

PACK	FMT	Text format 00: Data transfer from the master (HX-20)
+1	DID	Terminal ID (Drive A or B: 31_{16} Drive C or D: 32_{16})
+2	SID	Master ID (20_{16}) (HX-20)
+3	FNC	Message function
+4	SIZ	Text length minus 1
+5	d0	Data 0
	d1	Data 1
	⋮	

Fig. 10-5 Parameter Packet of Subroutine "OUTSRL"

(2) Subroutine call

The first address of the parameter packet is set in the index register to call subroutine "OUTSRL" (entry point: FF70). For details of the EPSP, refer to Chapter 4. For details of the EPSP functions on the TF-20 side, refer to the next page.

°EPSP side

Open file
Drive :   "A", filename, file type: ABC, BAS
File mode: Sequential output "0"

```
OUTSRL  EQU  $FF70
        LDX  =PACKET
        JSR  OUTSRL          ... Routine for data output
          :                      to the serial interface

PACKET  EQU  *
FMT     FCB  $00, $30, $20, $0F, $0E
MSG     FCB  $00, $01, $01,
        FCC  /ABC△△△△△/
        FCC  /BAS/
```

10.6 Function Table of Floppy Disk Unit

FMT	DID	SID	FNC	SIZ	Text data No.	Description of function and text
						Terminal floppy reset
00	SS	MM	0E	00	00	XX
01	MM	SS	0E	00	00	Return code 00
						File open
00	SS	MM	0F	0E	00	High-order byte of FCB address in HX-20
					01	Low-order byte of FCB address in HX-20
					02	Drive code (1: Drive A or 2: Drive B)
					03 ~ 0A	Filename
					0B ~ 0D	File type
					0E	Extent number (Normally 0)
01	MM	SS	0F	00	00	Return code
						BDOS error (See Note at the end of this table.)
						FF: File not found.
						Codes other than the above: Normal
						File close
00	SS	MM	10	01	00	High-order byte of FCB address in HX-20
					01	Low-order byte of FCB address in HX-20
01	MM	SS	10	00	00	Return code (The same return code as that at file open.)
						First data search
00	SS	MM	11	0C	00	Drive code (1 or 2)
					01 ~ 08	Filename
					09 ~ 0B	File type
					0C	Extent number (Normally 0)
01	MM	SS	11	20	00	Return code (The same return code as that at file open.)
					01 ~ 20	Directory FCB entry (The FCB of the found directory is entered.)
						Next data search
00	SS	MM	12	00	00	XX
01	MM	SS	12	20	00	Return code (The same return code as that at file open.)
					01 ~ 20	Directory FCB entry (The FCB of the found directory is entered.)

FMT	DID	SID	FNC	SIZ	Text data No.	Description of function and text
						File creation
00	SS	MM	16	0E	00	High-order byte of FCB address in HX-20
					01	Low-order byte of FCB address in HX-20
					02	Drive code (1 or 2)
					03 ~ 0A	Filename
					0B ~ 0D	File type
					0E	Extent number (Normally 0)
01	MM	SS	16	00	00	Return code (The same return code as that at file open.)
						File rename
00	SS	MM	17	1F	00	Drive code (1 or 2)
					01 ~ 08	Filename before change (8 characters)
					09 ~ 0B	File type before change (3 characters)
					0C	Extent number
					0D ~ 0F	Unused
					10	Drive code (1 or 2)
					11 ~ 18	Filename after change (8 characters)
					19 ~ 1B	File type after change (3 characters)
					1C	Extent number
					1D ~ 1F	Unused
01	MM	SS	17	00	00	Return code (The same return code as that at file open.)
						Random data read
00	SS	MM	21	04	00	High-order byte of FCB address in HX-20
					01	Low-order byte of FCB address in HX-20
					02	R0
					03	R1 } Random record numbers
					04	R2
01	MM	SS	21	82	00	Extent number
					01	Current record number
					02 ~ 81	Read data (128 bytes)
					82	Return code BDOS error (See Note at the end of this table.) Codes other than the above: Normal

FMT	DID	SID	FNC	SIZ	Text data No.	Description of function and text
						Random data write
00	SS	MM	22	84	00	High-order byte of FCB address in HX-20
					01	Low-order byte of FCB address in HX-20
					02 ~ 81	Write data (128 bytes)
					82	R1
					83	R1 Random record numbers
					84	R2
01	MM	SS	22	02	00	Extent number
					01	Current record number
					02	Return code BDOS error (See Note at the end of this table.) Codes other than the above: Normal
						File size calculation
00	SS	MM	23	01	00	High-order byte of FCB address in HX-20
					01	Low-order byte of FCB address in HX-20
01	MM	SS	23	05	00	Extent number
					01	Current record number
					02	R0
					03	R1 Random record numbers
					04	R2
					05	Return code (Always 0)
						Disk all copy
01	SS	MM	7A	00	00	Drive code (1 or 2)
01	MM	SS	7A	02	00	High-order byte of currently copied track number
					01	Low-order byte of currently copied track number 0 to 39 FFFF : End
					02	Return code (BDOS error or 0)
						Direct write into disk
00	SS	MM	7B	82	00	Drive code (1 or 2)
					01	Track number (0 to 39)
					02	Sector number (1 to 64)
					03 ~ 82	Write data (128 bytes)
00	MM	SS	7B	00	00	Return code (BDOS error or 0)
						Disk formatting (FRMAT)
00	SS	MM	7C	00	00	Drive code (1 or 2)
01	MM	SS	7C	02	00	High-order byte of currently formatted track number
					01	Low-order byte of currently formatted track number 0 to 39 FFFF : End
					02	Return code (BDOS error or 0)

FMT	DID	SID	FNC	SIZ	Text data No.	Description of function and text
						New system disk generation (SYSGEN)
00	SS	MM	7D	00	00	XX
01	MM	SS	7D	02	00	0000 : Not end
					01	FFFF : End
					02	Return code (BDOS error or 0)
						Disk free area calculation (DSKF)
00	SS	MM	7E	00	00	Drive code (1 or 2)
01	MM	SS	7E	01	00	Free area size (in 2K-byte units)
					01	Return code (BDOS error or 0)
						Direct read from disk (DSKI$)
00	SS	MM	7F	02	00	Drive code (1 or 2)
					01	Track number (0 to 39)
					02	Sector number (1 to 64)
01	MM	SS	7F	80	00 ~ 7F	Read data (128 bytes)
					80	Return code (BDOS error or 0)
						Disk boot
00	SS	MM	80	00	00	Application ID (in BASIC 80_{16} ... BOOT80.SYS)
01	MM	SS	80	FF	00	Return code 00 : Normal FF : File not found.
					01 ~ FF	Read data
						Load open
00	SS	MM	81	0D	00 ~ 07	Filename (the filename of DISK BASIC is "DBASIC".)
					08 ~ 0A	File type (the file type of DISK BASIC is "SYS".)
					0B	Relocate flag 00: Do not relocate. 01: Relocate from the starting address. 02: Relocate from the ending address.
					0C ~ 0D	Ending or starting address
01	MM	SS	81	02	00	Return code 00: Normal FF: File not found.
					01	High-order byte of file size
					02	Low-order byte of file size

FMT	DID	SID	FNC	SIZ	Text data No.	Description of function and text
						Load close
00	SS	MM	82	00	00	XX
01	MM	SS	82	00	00	Return code (Always 0)
						Read one block
00	SS	MM	83	01	00	High-order byte of current record number
					01	Low-order byte of current record number
01	MM	SS	83	82	00	High-order byte of current record number
					01	Low-order byte of current record number
					02 ~ 81	Read data
					82	Return code (00: Normal; FF: End)

NOTE: The term "BDOS error" used in the above table refers to one of the following errors; a read error (error code: FA), a write error (error code: FB), a drive select error (error code: FC), and a write protect error (error code: FD or FE).

The format of the file control block (FCB) used by the floppy disk unit is as follows:

0	1 8	9	10	11	12	13	14	15	16 31	32	33	34	35
dr	FN	t1	t2	t3	ex	s1	s2	rc	DM	CR	ro	r1	r2

dr: Disk drive code (00 to 16) (Use of code 05 to 16 will result in an error.)
00: A file is assigned to the standard disk drive.
01: Disk and disk drive A are selected automatically.
02: Disk and disk drive B are selected automatically.

16: Disk and disk drive P are selected automatically.

FN: Filename consisting of a maximum of 8 characters (in ASCII codes).
If no filename is given by the user, blanks (20) will be filled as the filename.

t1, t2, t3: File format (in ASCII codes)
As ASCII codes, bits in the upper row are selected and high-order bits set to 0 are used. These bits when represented by t1, t2 and t3 are as follows.
t1=1: Read only file
t2=1: No system file, FILES list

ex: File extent (Normally 0)
This is a number to indicate the current location of the logical extent, and is normally set to 00 by the user. This number must be a value in the range of 0 to 31 when a file input/output is to be performed.

s1: Used within the system.

s2: Used within the system. s2 is set to 0 when a file is to be opened, created, or called for search.

rc: Record number of the logical extent indicated by "ex" and must be a value in the range of 0 to 128.

DM: A value set and used by the system.

cr: A value indicating the location of the record where data read/write is being performed in sequential file processing. This value is normally set to 0 by the user.

r0, r1, r2: Random record number indicated by a value in the range of 0 to 65535. r0, r1 and r2 are used to configure 24 bits. r0 indicates the low-order digit, r1 the high-order digit, and r2 an overflow.

CHAPTER 11 SLAVE MCU COMMANDS

Page

11.1 General 11-1
11.2 Commands for Slave MCU Control 11-2
11.3 Cancelling a Command 11-2
11.4 Slave MCU Command Transmission Subroutine 11-3
11.5 Commands to Slave MCU 11-4

APPENDIX Sample Lists 11-16

11.1 General

The interface between the master and slave MCUs consists of two signal lines. Serial communication is performed at 38.4K BPS. Slave MCU operations are performed in response to instructions (commands) sent from the main MCU. The master CPU uses the serial interface to communicate either with the slave MCU or externally.

The slave CPU supports the following functions:

(1) Operation of the microprinter
(2) Data reception via RS-232C port
(3) Data I/O for external cassette
(4) Data I/O and operation of built-in microcassette
(5) Output for piezoelectric speaker
(6) Control switches for serial, power supply and bar code reader power

11.2 Commands for Slave MCU Control

Commands are sent to the slave MCU via the 38.4K-BPS serial interface. Commands are one byte in length. However, for some commands, parameters are added. The standard communication procedure involves sending a command from the master MCU and receiving an ACK signal from the slave MCU in response. The sequence for commands sent with parameters is shown below.

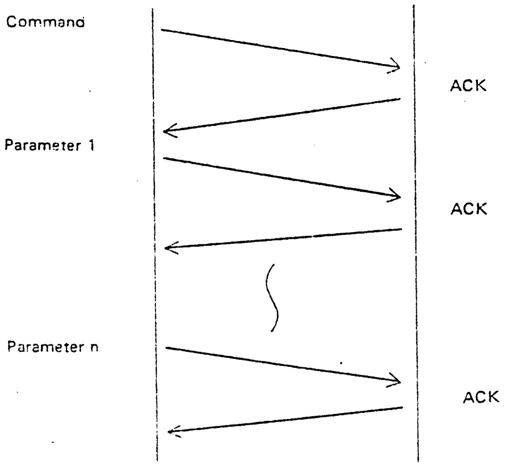


Fig. 11-1 Command Transmission Sequence

First, a 1-byte command is sent to the slave MCU. The SNSCOM subroutine (entry point FF19) is called to receive the ACK signal. For details of commands, see the command table.

For data reception from the RS-232C or cassette, the slave MCU sends serial input data to main MCU upon completion of command reception. Data received by the slave MCU under this condition are assumed to be commands and the current input mode is cancelled.

11.3 Cancelling a Command

The command being executed is cancelled if an overrun error occurs during serial communication. (For example, if overrun occurs when 100-line feed is specified for the microprinter, the current command is aborted and the system goes into WAIT status pending receipt of a fresh command.) If new data is received from main MCU while a command is being executed by the slave MCU, the data is set in the receive register but not processed. At this point, if new serial communication data is received, the data in the register is destroyed, causing an overrun error.

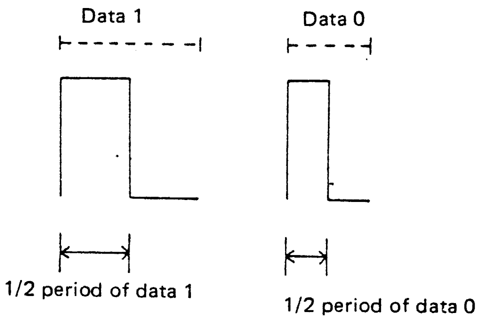
To cancel a command, the master MCU sends a series of BREAK commands to the slave MCU. Subroutine BREAKIO (entry point FFA3) is provided for this purpose.

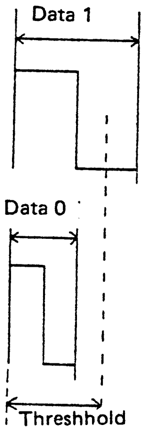
11.4 Slave MCU Command Transmission Subroutine

Subroutine name	Entry point	Description
SNSCOM	FF19	Transfers a command or 1 byte of data to the slave MCU via SCI. Parameters: At Entry (A) Transmit data (Command) At Return (C): Abnormal I/O flag (A): Return code (Transmit data from slave MCU) Registers retained (B), (X) Subroutines referenced None Variables used None

11.5 Commands to Slave MCU

Command	Master MCU data	Slave MCU response	Description
00	00	01 (ACK)	Slave MCU ready check. ACK is returned when the slave MCU is ready to receive a command. The slave MCU makes no response if it is not ready.
01	01	01 (ACK)	Sets the constants required by slave MCU in the field. The following values are set: Generated polynomial expressions, BCC register value, RS-232C bit rate, cassette (external or built-in microcassette), microcassette tape counter setting.
02	02	01 (ACK)	Initialization. The status of serial communication driver remains unchanged.
03	03 (Command) AA (Parameter)	01 (ACK)	Opens masks for special commands. Commands 06, 07, 08 and 0B cannot be executed unless the masks are opened. Any value other than AA indicates that the mask is closed.
04	04	01 (ACK)	Closes masks for special commands.
05	05 ah (Upper byte of address) al (Lower byte of address)	01 (ACK) 01 (ACK) d (Data)	Reads slave MCU memory. NAK (0F) is returned in response to 05 if the mask is not open.
06	06 ah (Upper byte of address) al (Lower byte of address) d (Data)	01 (ACK) 01 (ACK) 01 (ACK) 01 (ACK)	Stores data to the memory address specified by the slave MCU. 0F (NAK) is returned and command execution is aborted if the mask is not opened.
07	07 ah (Upper byte of address) al (Lower byte of address) d (Data)	01 (ACK) 01 (ACK) 01 (ACK) 01 (ACK)	Performs logical OR operation for the data at the memory address specified by the slave MCU and the specified data and stores the result in the specified address. 0F (NAK) is returned and command execution is aborted if the mask is not opened.
08	08 ah (Upper byte of address) al (Lower byte of address) d (Data)	01 (ACK) 01 (ACK) 01 (ACK) 01 (ACK)	Performs logical AND operation for the data at the memory address specified by the slave MCU and the specified data and stores the result in the specified address. 0F (NAK) is returned and command execution is aborted if the mask is not opened.

Command	Master MCU data	Slave MCU response	Description
Ø9	Ø9	Ø1 (ACK)	Unused (In version 2, bar-code reader power ON)
ØA	ØA	Ø1 (ACK)	Unused (In version 2, bar-code reader power OFF)
ØB	ØB ah (Upper byte of address) al (Lower byte of address)	Ø1 (ACK) Ø1 (ACK) Ø1 (ACK)	Sets the program counter to a specified value. (Jumps execution to a specified address.) ØF is returned and command execution is aborted, if the mask is not opened.
ØC	ØC	Ø2 (ACK for BREAK)	BREAK. Terminates processing and sets the system to command WAIT status.
ØD	ØD AA	Ø1 (ACK) Ø1 (ACK)	Cuts OFF power supply. Command execution is aborted if parameter AA is omitted.
ØE ~ ØF			Undefined
1Ø	1Ø d (Data)	Ø1 (ACK) Ø1 (ACK)	Activates the built-in printer. Prints out 6-dot data (Bit Ø to bit 5). One dot-line is printed by repeating this command procedure 24 times.
11	11 d (Number of lines)	Ø1 (ACK) Ø1 (ACK)	Feeds the specified number of dot lines to the built-in printer.
12	12	Ø1 (ACK)	Activates built-in printer motor for approx. 1.2 sec. (Paper feed operation)
13 ~ 1F			Undefined
2Ø	2Ø	21 (ACK)	Executes external cassette ready check. Code 21 is returned when the external cassette is ready. The external cassette makes no response if it is not ready.
21	21 d1 (Upper byte of time (MCU clocks) of 1/2 cycle for data '1') d2 (Lower byte of time (MCU clocks) of 1/2 cycle for data '1') d3 (Upper byte of time (MCU clocks) of 1/2 cycle for data 'Ø')	Ø1 (ACK) 21 (ACK) 21 (ACK) 21 (ACK)	Sets constants for the external cassette.  Data 1 Data 0 1/2 period of data 1 1/2 period of data 0

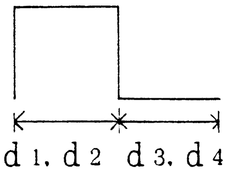
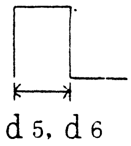
Command	Master MCU data	Slave MCU response	Description
	d4 (Lower byte of time (MCU clocks) of 1/2 cycle for data 'Ø')	21 (ACK)	The times (in MCU clock pulses) for 1/2 cycle for data '1' and for data 'Ø' are set as constants. The bit judgment threshold value for data read is also set as the number of MCU clocks (Fig. below).
	d5 (Upper byte of bit judgment threshold value between cycle times for '1' and '2')	21 (ACK)	This data represents the interblock gap length in tape stop mode (long gap) as the number of times that data FF is written to the tape.
	d6 (Lower byte of bit judgment threshold value)	21 (ACK)	
	d7 (Upper byte of interblock gap length (in bytes) in stop mode (tape head stops between blocks))	21 (ACK)	
	d8 (Lower byte of interblock gap length in stop mode)	21 (ACK)	
22	22	Ø1 (ACK)	Turns the external cassette REM terminal ON.
23	23	Ø1 (ACK)	Turns the external cassette REM terminal OFF.
24	24	Ø1 (ACK)	Writes 1 block of data in EPSON format. After synchronizing pattern is sent, the number of bytes specified as the block length is written followed by 2 CRC bytes. For output data, only the number of bytes specified as the block length are required. If data has not been received from the master MCU when the slave attempts to write data to the cassette, the slave MCU returns 2F, activates the speaker (88ØHz for 1 sec.) and terminates cassette output. Block write start mode values are as follows (d1) ØØ: 125-byte gap before the block (default valve). Ø1: 15-byte gap before the block. FF: 625-byte gap before the block. Block write start mode value (ØØ, Ø1 or FF) is used as the block write end mode value at the completion of block write operation. In ØØ and FF modes, the REM terminal is turned after completion of block write operation.
	d1 (Block write start mode)	21 (ACK)	
	d2 (Block write end mode)	21 (ACK)	
	d3 (Upper byte of block length)	21 (ACK)	
	d4 (Lower byte of block length)	21 (ACK)	
	W1 (Output data)	22 (ACK) (2F(NAK))	
	∫		
	Wm (Output data)	22 (ACK) (2F(NAK))	

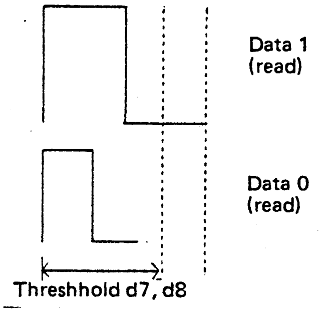
Command	Master MCU data	Slave MCU response	Description
25	25 d1 (Upper byte of number of FF patterns) d2 (Lower byte of number of FF patterns)	01 (ACK) 21 (ACK) 21 (ACK)	Outputs number of FF patterns specified by d1 and d2 to the external cassette. Writing of data FF is unrelated to blocking.
26	26 d1 (Block read start mode) d2 (Block read end mode) d3 (Upper byte of block length) d4 (Lower byte of block length)	01 (ACK) 21 (ACK) 21 (ACK) 21 (ACK) 21 (ACK) W1 W2 W3 ∫ W84	Inputs files from an external cassette. Searches header block (EPSON format) and sends the contents of this block to the master MCU. Header block always begins with data H. In actual practice, however, d1 is ignored. REM is turned OFF after reading 1 block if d2 is 00. If d2 is 01, REM is left ON. If an error occurs during transmission of block data, data transmission is terminated and P34 (connected to P12 of the master MCU) is turned ON. Two CRC bytes are placed at the end of the block but are not transmitted.
27	27 d1 (Block read start mode) d2 (Block read end mode) d3 (Upper byte of block length) d4 (Lower byte of block length)	01 (ACK) 21 (ACK) 21 (ACK) 21 (ACK) 21 (ACK) W1 W2 W3 ∫ W84	Inputs files from an external cassette. Seaches EOF block (EPSON format) and sends the contents of this block to the master MCU. EOF block always begins with data E. Parameters and execution result are identical to those for command 26.
28	28 d1 (Block read start mode) d2 (Block read end mode) d3 (Upper byte of block length) d4 (Lower byte of block length)	01 (ACK) 21 (ACK) 21 (ACK) 21 (ACK) 21 (ACK) W1 W2 W3 ∫ W260	Inputs files from an external cassette. Inputs the next block (EPSON format) and sends the data to the master MCU. The block may beging with any data. Parameters and execution result are identical to those for command 26.
29			Undefined
2A			Undefined

Command	Master MCU data	Slave MCU response	Description
2B	2B d1 (Specifies the pulse mode)	01 (ACK) 21 (ACK)	Specifies the input signal for the external cassette and built-in microcassette. d1 Bit 3: When logic '1', the microcassette input signal is as defined by bit 2. When logic '0', the microcassette input signal is judged at input. Bit 2: When logic '1', the microcassette input signal is reversed. When logic '0', the microcassette input signal is normal. Bit 1: When logic '1', the external cassette input signal is defined by bit 0. When logic '0', the external cassette input signal is judged at input. Bit 0: When logic '1', the external cassette input signal is reversed. When logic '0', the external cassette input signal is normal. NOTE: In versions 1 and 2, the slave MCU assumes (Bit 3, Bit 2) = (1, 1) when bit 3 is logic '0'.
30	30 d1 (Tone) d2 (Duration)	01 (ACK)	Specifies the tone and duration and sounds the piezoelectric speaker. The specifications for tone are as follows: 0 = pause, 1, 2, 3 ... correspond to C, D, E ... . Values 1 to 28_{10} represent a 4-octave major scale (13 = 880Hz) and values 29 to 56_{10} a scale each tone of which is a half tone higher than that represented by 1 to 28. Duration is specified with 1 = 0.1 sec., 2 = 0.2 sec, etc. 0 specifies a pause (command not executed).
31	31 d1 (Upper byte of frequency specification) d2 (Lower byte of frequency specification) d3 (Upper byte of duration specification) d4 (Lower byte of duration specification)	01 (ACK) 31 (ACK) 31 (ACK) 31 (ACK) 31 (ACK)	Specifies the frequency and duration and sounds the piezoelectric speaker. Frequency is specified as the number of MCU clocks corresponding to 1/2 cycle. Example: 349_{10} for 880Hz. Specification of duration: 1 = 400µsec. (256 MCU clocks)
32	32	01 (ACK)	Sounds the speaker for 0.03 sec at tone 6 using command 30.
33	33	01 (ACK)	Sounds the speaker for 1 sec at tone 20 using command 30.

Command	Master MCU data	Slave MCU response	Description
34	34 d s1 d 11 d s2 d 12 d sn d 1n FF	01 (ACK) 31 (ACK) 31 (ACK) 31 (ACK) 31 (ACK) 31 (ACK) 31 (ACK) 31 (ACK)	Sets melody data in the slave MCU buffer. Buffer size is 48 bytes. The data set here can be output to the speaker using command 35. The format for data is the same as for command 30, i.e., tone, duration. As a pair, these data repeatedly specify the tone and duration. Due to the buffer size, the maximum number of data is 46_{10}. Data must end with FF. The data set in the buffer remains unchanged unless it is rewritten by command 34 or destroyed by a printer command. (This is because this buffer is also used by printer.)
35	35	01 (ACK)	Sounds the piezoelectric speaker in accordance with the melody data specified in command 34.
36 ~ 3F			Undefined
40	40	01 (ACK)	Turns the serial driver ON. RTS is set to low (OFF).
41	41	01 (ACK)	Turns the serial driver OFF.
42	42 d1 (Upper byte of bit rate) d2 (Lower byte of bit rate) d3 (Word length) d4 (Mode)	01 (ACK) 41 (ACK) 41 (ACK) 41 (ACK) 41 (ACK)	Selects RS-232C mode. Bit rate corresponds to bit time specified as the number of CPU clock cycles (for example, 800_{16}: 300 BPS). Word length (excluding parity bits) may be set at 5, 6, 7 or 8 bits. The significance of each bit of mode data (d4) is as follows. Bit 0 : Bit 1 : Number of stop bits (1 or 2) Bit 2: '0': Carrier check '1': No carrier check Bit 3: Controls RTS output. '0': low '1': high Bit 4: Undefined Bit 5: Undefined Bit 6 : Parity control. Bit 7 : '00': Even parity '01': Odd parity '10' or '11': None
43	43	V	Inputs RS-232C status maintained by the slave MCU. The significance of each bit of the status code is as follows. (Logic '1' indicates an error.)

Command	Master MCU data	Slave MCU response	Description
43	43	V	Bit 0: Carrier detect Bit 1: Parity Bit 2: Overrun Bit 3: Framing Bit 4: Undefined Bit 5: Undefined Bit 6: Undefined Bit 7: Undefined Error status bits are reset by a clear command (44) or when input is resumed (command 45).
44	44	01 (ACK)	Clears RS-232C error status.
45	45	01 (ACK) V1 V2	Starts RS-232C input. Input data is sent to the master MCU. If the word length of the data (including the parity bits) is less than 8 bits, the remaining bits (from MSB) are padded with data 0 (right-justified). P34 (connected to master MCU P12) is reset (logic '1')when input starts. P34 is set (logic '1') if an error (framing error, carrier OFF, etc.) occurs. Data reception terminates upon receipt of a new command from the master MCU.
46	46	01 (ACK)	Terminates RS-232C input initiated by command 45. (This is not the only way of terminating such input.)
47			Undefined
48	48 d1 (Upper byte of polynomial expression) d2 (Lower byte of polynomial expression)	01 (ACK) 41 (ACK) 41 (ACK)	Sets the polynomial expression used for CRC check. This polynomial expression can also be used for cassette files. Default value is 8408 ($1+X^5+X^{12}+X^{16}$).
49	49 d1 (Upper byte of BCC register value) d2 (Lower byte of BCC register value)	01 (ACK) 41 (ACK) 41 (ACK)	Sets BCC register values for CRC check. This value is used as the initial value when CRC calculation is performed at RS-232C input. However, the data in BCC register is lost when I/O operations to a cassette are performed.
4A	4A	V	Inputs upper byte of BCC register value.

Command	Master MCU data	Slave MCU response	Description
4B	4B	V	Inputs lower byte of BCC register value.
4C	4C	01 (ACK)	Activates the serial driver. In contrast to command 40 which turns RTS OFF, this command does not affect the status of RTS.
4D	4D	01 (ACK)	RTS high/low specification. Only bit 0 is significant. 0: low, 1: high
4E			Undefined
4F			Undefined
50	50	V	Identifies the plug-in option. Bit states of P46 and P20 are returned. Bit 0: Bit state of P46 Bit 1: Bit state of P20 Bit 2 to 7: 0 NOTE: Plug-in option power is turned OFF when this command is executed.
51	51	01 (ACK)	Turns power of plug-in ROM cartridge ON.
52	52	01 (ACK)	Turns power of plug-in ROM cartridge OFF.
53 ~ 5F			Undefined.
60	60	61 (ACK)	Executes ready check. Same MCU responds only if a microcassette command is executable. In all other cases, no response is sent.
61	61 d1 (Upper byte of signal low time of one cycle for data '1') d2 (Lower byte of signal low time of one cycle for data '1') d3 (Upper byte of signal high time of one cycle for data '1') d4 (Lower byte of signal high time of one cycle for data '1')	01 (ACK) 01 (ACK) 61 (ACK) 61 (ACK) 61 (ACK)	Sets the microcassette parameters. Parameters are specified using data d1 to d8.  Data 1 (write)  Data 0 (write)

Command	Master MCU data	Slave MCU response	Description
	d5 (Upper byte of time of 1/2 cycle for data 'Ø')	61 (ACK)	
	d6 (Lower byte of d5)	61 (ACK)	
	d7 (Upper byte of 'Ø', '1' bit judgment threshold value)	61 (ACK)	
	d8 (Lower byte of d7)	61 (ACK)	
62	62	Ø1 (ACK)	Specifies the number of gap bytes for each mode when stopping the microcassette between blocks.
	d1 (Upper byte of number of gap bytes)	61 (ACK)	
	d2 (Lower byte of number of gap bytes)	61 (ACK)	
63	63	Ø1 (ACK)	Advances the tape (in PLAY mode) for the specified number of bytes. The bit judgement threshold value is taken as the length of one bit and 9 bits are counted as one byte. This command does not perform data read.
	d1 (Upper byte of the number of bytes sent)	61 (ACK)	
	d2 (Lower byte of the number of bytes sent)	61 (ACK)	
64	64	Ø1 (ACK)	Outputs one block to microcassette in EPSON format. Output file and command format and execution result are identical to command 24 (block output to external cassette).
	d1 (Block write start mode)	61 (ACK)	
	d2 ((Block write end mode)	61 (ACK)	
	d3 (Upper byte of block length)	61 (ACK)	
	d4 (Lower byte of block length)	61 (ACK)	
	W1 (Data)	61 (ACK)	
	∫	(6F (NAK))	
	Wm (Data)	61 (ACK)	
65	65	Ø1 (ACK)	Outputs the number of bytes of data FF specified by d1 and d2 to the microcassette. Result is the same as command 25.
	d1 (Upper byte of number of bytes)	61 (ACK)	
	d2 (Lower byte of number of bytes)	61 (ACK)	
66	66	Ø1 (ACK)	Inputs files from microcassette. Command operation and parameters are identical to command 26.
	d1 (Block read start mode)	61 (ACK)	
	d2 (Block read end mode)	61 (ACK)	
	d3 (Upper byte of block length)	61 (ACK)	
	d4 (Lower byte of block length)	61 (ACK)	
		W1	

Command	Master MCU data	Slave MCU response	Description
		W2 ∫ W84	
67	67 d1 (Block read start mode) d2 (Block read end mode) d3 (Upper byte of block length) d4 (Lower byte of block length)	01 (ACK) 61 (ACK) 61 (ACK) 61 (ACK) 61 (ACK) W2 ∫ W84	Inputs files from microcassette. Command operation and parameters are identical to command 27.
68	68 d1 (Block read start mode) d2 (Block read end mode) d3 (Upper byte of block length) d4 (Lower byte of block length)	01 (ACK) 61 (ACK) 61 (ACK) 61 (ACK) 61 (ACK) W1 W2 ∫ W260	Inputs files from microcassette. Command operation and parameters are identical to command 28.
69			Undefined
6A			Undefined
6B			Undefined
6C			Undefined
6D	6D d1 (Upper byte of counter value) d2 (Lower byte of counter value)	01 (ACK) 61 (ACK) 61 (ACK)	Sets microcassette counter value in the slave MCU. The counter value is a 16-bit signed hexadecimal number.
6E	6E	V	Fetches microcassette counter value. Sends the upper 8 bits of counter value to the master MCU.
6F	6F	V	Fetches microcassette counter value. Sends the lower 8 bits of counter value of the master MCU.

Command	Master MCU data	Slave MCU response	Description
70	70	V	Executes microcassette write protect check. In write enable status, '0' is returned to the master MCU. In write protect status, 'FF' is returned to MCU.
71	71 d1 (Upper byte of counter value) d2 (Lower byte of counter value)	01 (ACK) 61 (ACK) 61 (ACK)	Rewinds microcassette tape to the tape counter value specified by d1 and d2. Speed of rewind is same as that of fast forward.
72	72 d1 (Upper byte of counter value) d2 (Lower byte of counter value)	01 (ACK) 61 (ACK) 61 (ACK)	Advances the microcassette tape (fast forward) to the counter value specified by d1 and d2.
73	73	01 (ACK)	Causes the microcassette to rewind up to the beginning of tape (fast rewind).
74	74	V	Inputs microcassette status to the slave MCU. Status is a one-byte code. The significance of each bit is as follows. (Logic '1' indicates an error.) Bit 0: Tape read error Bit 1: Undefined Bit 2: Header or EOF block not found Bit 3: Delay in data transmission from master MCU during data output Bit 4: Write protect Bit 5: Head error Bit 6: Microcassette not connected Bit 7: Undefined
75	75	01 (ACK)	Clears the microcassette status register.
76	76	01 (ACK) (6F (NAK))	Loads the microcassette head. If an error occurs during loading, the slave MCU returns '6F'.
77	77	01 (ACK) (6F (NAK))	Unloads the microcassette head. If an error occurs during unloading, the slave MCU returns '6F'.
78	78	01 (ACK) (6F (NAK))	Rewinds the microcassette tape. Rewind operation continues until the next command is received.
79	79	01 (ACK) (6F (NAK))	Advances the microcassette tape (fast forward). Fast forward continues until the next command is received.

Command	Master MCU data	Slave MCU response	Description
7A	7A	01 (ACK) 6F (NAK))	Advances the microcassette tape (slow forward). Slow forward continues until the next command is received.
7B	7B	01 (ACK) 6F (NAK))	Stops microcassette tape forward and rewind operations.
7C			Undefined
7D			Undefined
7E ~ 7F			Undefined
80	80 d1 (Upper byte of address) d2 (Lower byte of address) d3 (Bit position)	01 (ACK) 0F (NAK)) 01 (ACK) 01 (ACK) 01 (ACK)	Causes master MCU PLG2 port (Address 26, bit 5) value to be stored in the specified bit in the slave MCU. The PLG2 port value is stored in the bit specified by d3 at the slave MCU address specified by d1 and d2. This operation continues until the next command is received. As this is a special command, the mask must be opened prior to execution (command 03). This command will not be accepted if the mask has not been opened.
81	81 d1 (Upper byte of address) d2 (Lower byte of address) d3 (Bit position)	01 (ACK) (0F (NAK)) 01 (ACK) 01 (ACK) 01 (ACK)	Stores the value of the specified bit in the slave MCU to P12 of the master MCU. The slave MCU address is specified by d1 and d2 and the bit position is specified by d3. If any of the data at the position specified by d3 (1) is '1', '1' will be stored in P12. In all other cases, '0' will be stored in P12. Like command 80, this command is a special command.
81 ~ 8F			Undefined
90 ~ FF			Undefined

```
      *** 6301 CROSS MACROASSEMBLER   VER1.0 ***                                    10/21/82    09:42:21

ERR   SEQ    LOC  OBJECT        PROGRAM  SLAVE       --- SEND SLAVE COMMAND ---

    00001                            *
    00002                                   NAM    SLAVE
    00003                                   TTL    --- SEND SLAVE COMMAND ---
    00004                                   OPT    LOAD
    00005                                   OPT    PAGE=55
    00006                            *  FILE NAME 'EX$6'    BY K.A
    00007                            *
    00008                            * SEND COMMAND TO SLAVE MCU
    00009                            * SEND MELODY PATTERN TO SLAVE MCU AND SEND COMMAND TO PLAY MELODY.
    00010                            *
    00011A 1000                             ORG    $1000
    00012                            *
    00013                            *
    00014           0011   A         TRCSR  EQU    $11
    00015           0012   A         SRDR   EQU    $12
    00016           0013   A         STDR   EQU    $13
    00017                            *
    00018                            * SET SLAVE MCU THE MELODY (YANKEE DODLE)
    00019A 1000 CE 1022    A         PLAY   LDX    #MELTBL    * (X):THE ADDRESS WHERE MELODY DATA ARE STORED.
    00020A 1003 86 34      A                LDA A  #$34       * SEND DATA TO SLAVE MCU.
    00021A 1005 8D 0E 1015                  BSR    SNDSLV     * COMMAND 34: SET MELODY DATA
    00022A 1007 A6 00      A         SLV10  LDA A  0,X        * SET DATA
    00023A 1009 8D 0A 1015                  BSR    SNDSLV
    00024A 100B 08                          INX
    00025A 100C 81 FF      A                CMP A  #$FF       * LAST DATUM IS $FF.
    00026A 100E 26 F7 1007                  BNE    SLV10
    00027                            *
    00028                            * PLAY MELODY.
    00029A 1010 86 35      A                LDA A  #$35
    00030A 1012 8D 01 1015                  BSR    SNDSLV
    00031                            *
    00032A 1014 39                          RTS
    00033                            *
    00034                            *
    00035                            *
    00036                            * SUBROUTINE
    00037                            *  SEND COMMAND TO SLAVE MCU
    00038                            *  PARAMETER
    00039                            *   ON ENTRY
    00040                            *    (A): COMMAND
    00041                            *   ON EXIT
    00042                            *    (A): RECIEVED CODE
    00043                            *   REGISTER PRESERVE  (X),(B)
    00044                            *
    00045A 1015 7B 2011    A         SNDSLV TIM    #$20,TRCSR * TX READY ?
    00046A 1018 27 FB 1015                  BEQ    SNDSLV
    00047A 101A 97 13      A                STA A  STDR       * SEND COMMAND
    00048                            * RECIVE FROM SLAVE MCU
    00049A 101C 7D 0011    A         SNDS10 TST    TRCSR      * RX READY ?
    00050A 101F 2A FB 101C                  BPL    SNDS10
    00051A 1021 39                          RTS
    00052                            *
    00053                            *
    00054                            * MELODY TABLE (YANKEE DOODLE)
    00055A 1022    29      A         MELTBL FCB    41,10,41,10,15,10,16,10
```

```
         A 1023    0A     A
         A 1024    29     A
         A 1025    0A     A
         A 1026    0F     A
         A 1027    0A     A
         A 1028    10     A
         A 1029    0A     A
    00056A 102A    29     A                   FCB     41,10,16,10,15,10,11,10
         A 102B    0A     A
         A 102C    10     A
         A 102D    0A     A
         A 102E    0F     A
         A 102F    0A     A
         A 1030    0B     A
         A 1031    0A     A
    00057A 1032    29     A                   FCB     41,10,41,10,15,10,16,10
         A 1033    0A     A
         A 1034    29     A
         A 1035    0A     A
         A 1036    0F     A
         A 1037    0A     A
         A 1038    10     A
         A 1039    0A     A
    00058A 103A    29     A                   FCB     41,20,13,10,11,10
         A 103B    14     A
         A 103C    0D     A
         A 103D    0A     A
         A 103E    0B     A
         A 103F    0A     A
    00059A 1040    FF     A                   FCB     $FF
    00060                           *
    00061          0000   A                   END
***** TOTAL ERRORS     0
```

CHAPTER 12 BAR-CODE READER

Page

12.1 General 12-1
12.2 Input/Output Ports related to the bar-code reader 12-1
12.3 Procedure for Data Input 12-2
12.4 Printing Bar Codes with MX-80 Series Printers 12-4

APPENDIX - Sample Lists 12-5

12.1 General

A bar code is a code which uses combinations of bars of varying thicknesses, designed to be read by an optical wand, and provides an effective means as a consumer product information code in inventory control, etc. (The current BASIC version of the HX-20 does not support the input/output of bar codes.)
This chapter describes the methods of inputting bar codes and printing them out using MX-80 series printers. (These functions will become available only with the external BASIC.)

12.2 Input/Output Ports related to the bar-code reader

Input/output ports related to the bar-code reader are shown in Table 1 below.

Table 1 Input/Output Ports Related to the Bar-Code Reader

MCU	Port	Direction	Function
Master MCU	P20	Input	Bar-code input signals (1: Mark (black), 0: Space (white))
Slave MCU	P35	Output	Bar-code reader power supply (0: On, 1: Off)
	P41	Output	Always 0

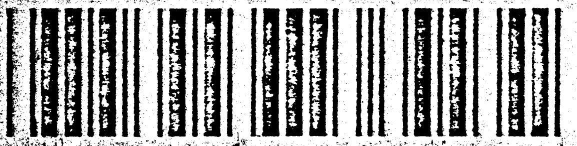


Fig. 12-1 Bar Codes

When bar codes are to be scanned with a bar-code reader, each bar (black) is input as binary "1" (mark) and a blank (white) between bars is input as binary "0" (space) to the P20 of the master MCU. A code is input by measuring the time duration of the black and white elements of the code.

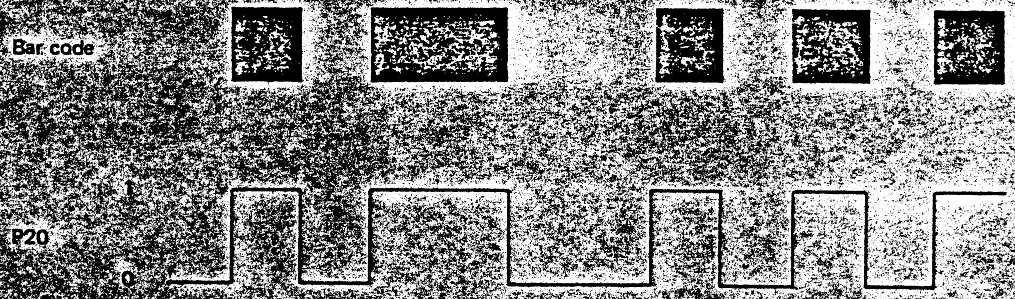


Fig. 12-2 Bar Code Scanned and Input Signal

12.3 Procedure for Data Input

12.3.1 Turning On the Power supply of the bar-code reader.
Before inputting data to the bar-code reader, its power supply must be turned on as follows:

```
 SNSCOM   EQU        $FF19
          LDA   A    #$03        * Opens the special command mask of
                                   the slave MCU.
          JSR        SNSCOM
          LDA   A    #$AA
          JSR        SNSCOM
          LDA   A    #$08        * Turns off the P35 of the slave MCU.
          JSR        SNSCOM
          LDA   A    #$00        * (Turns off the bit 5 at address 0006H.)
          JSR        SNSCOM
          LDA   A    #$6
          JSR        SNSCOM
          LDA   A    #$DF
          JSR        SNSCOM      * Special command of slave MCU
          LDA   A    #$04        * Closes the special command masks.
          JSR        SNSCOM
```

12.3.2 Data input
Data must be input only after the power supply of the bar-code reader has been turned On. Data input is accomplished by measuring the time duration of the binary 1 or binary 0 at the P20 of the master MCU as follows:
Here, it is assumed that the initial state of P20 is binary 1 (black).

(1) Time measurement of the binary 1 (black) state

(a) Set the bit 1 of TCSR to "0" (by specifying as a change of the input from "1" to "0"). ("AIM #$FD, TCSR").

(b) Wait until the bit 7 of TCSR becomes "1" (indicating that the edge detection has been completed). The period of FRC at this point is approximately 0.1 sec. In normal bar-code scanning, the thickness of any single bar in a bar code will not exceed this time interval of 0.1 sec. Time-out monitoring is performed by the OCR in the bar-code reader so that any bar exceeding 0.1 sec. in time duration may be detected as an error. Data setting in the OCR is performed as follows:

```
  LDD        FRC        * Sets the timing of the OCR interrupt
                          at 0.1 sec.
  STD        OCR        * 0.1 sec is judged.
  LDA    B   TCSR       * Clears the bit 6 (Output compare flag)
                          of TCSR.
  STA    A   OCR
```

Edge detection is performed as follows:

```
LOOP      LDA    A   TCSR       * When bit 7=1, it indicates that the
                                  edge detection has been completed.
          BMI        EDGE
          BIT    A   #$40       * Monitors the time-out condition.
          BEQ        LOOP
          JMP        TIMOUT     * Executes the time-out processing.
EDGE      LDD        ICR        * (A, B) ← Time duration of binary 1.
          SUBD       LSTTIM
          LDX        ICR        * Stores the time when the edge is
          STX        LSTTIM       detected.
            .
            .
            .
LSTTIM RMB            2
```

(2) Time measurement of the binary 0 (white) state
The time duration of binary 0 is measured by the same procedure as described above, except the bit 1 of TCSR is set to "1" (by specifying as a change of the input edge from "0" to "1").

12.3.3 Turning Off the power supply of the bar-code reader

Upon completion of the data input to the bar-code reader, the power supply of the bar-code reader must be turned off as follows:

```
SNSCOM    EQU        $FF19
          LDA    A   #$03       * Opens the special command mask of the
                                  slave MCU.
          JSR        SNSCOM
          LDA    A   #$AA
          JSR        SNSCOM
          LDA    A   #$07       * Turns On the P35 of slave MCU.
          JSR        SNSCOM
          LDA    A   #$00       * (Turns ON the bit 5 at address 0006H).
          JSR        SNSCOM
          LDA    A   #$06
          JSR        SNSCOM
          LDA    A   #$20
          JSR        SNSCOM
          LDA    A   #$04       * Closes the special command mask of the
          JSR        SNSCOM       slave MCU.
```

12.4 Printing Bar Codes with M-80 Series Printers

The method of printing bar codes is explained using the codes shown in Fig. 12-3 as an example. The codes in this figure are available in two types of bars differing in thickness or width and two types of blanks differing in width. Namely, a 0.3mm narrow bar and a 1.0mm wide bar and a 0.3mm narrow blank and 1.0mm wide blank. To print these bars at a height of 1.7cm with any MX-80 series printer, the following must be specified.

(1) Paper feed pitch: 4/216 inch (specified with ESC, "3", 4)
(2) Dot density   : 960 dots/line (specified with ESC, "L, n_1, n_2)

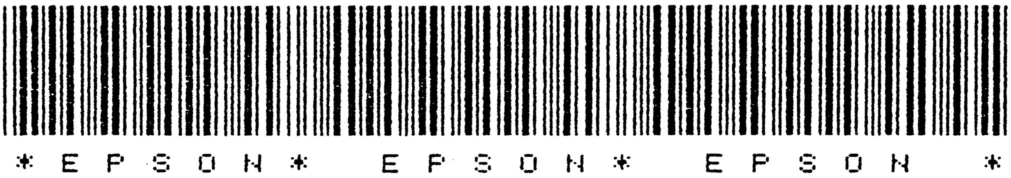


Fig. 12-3 Print Sample of Bar Codes with MX-80

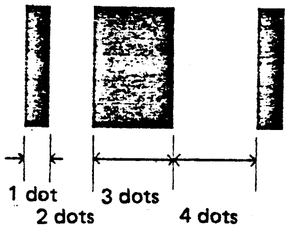


Fig. 12-4 Print Spacing with MX-80

The print spacing must be as shown in Fig. 12.4.

Narrow bar	1-dot space
Wide bar	3-dot space
Narrow blank	2-dot space
Wide blank	4-dot space

A sequence of 8 dots is printed 16 times to produce a 1.7cm long bar (see Fig. 12.5).

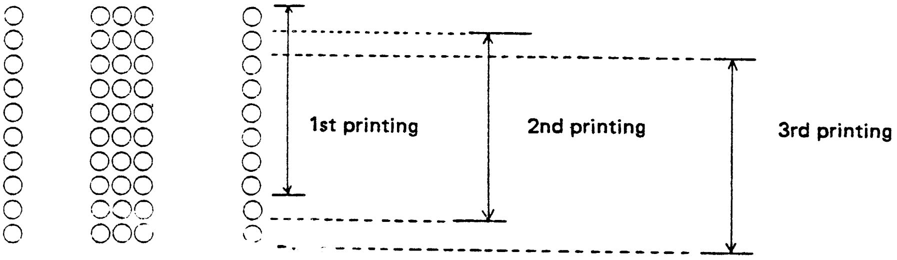
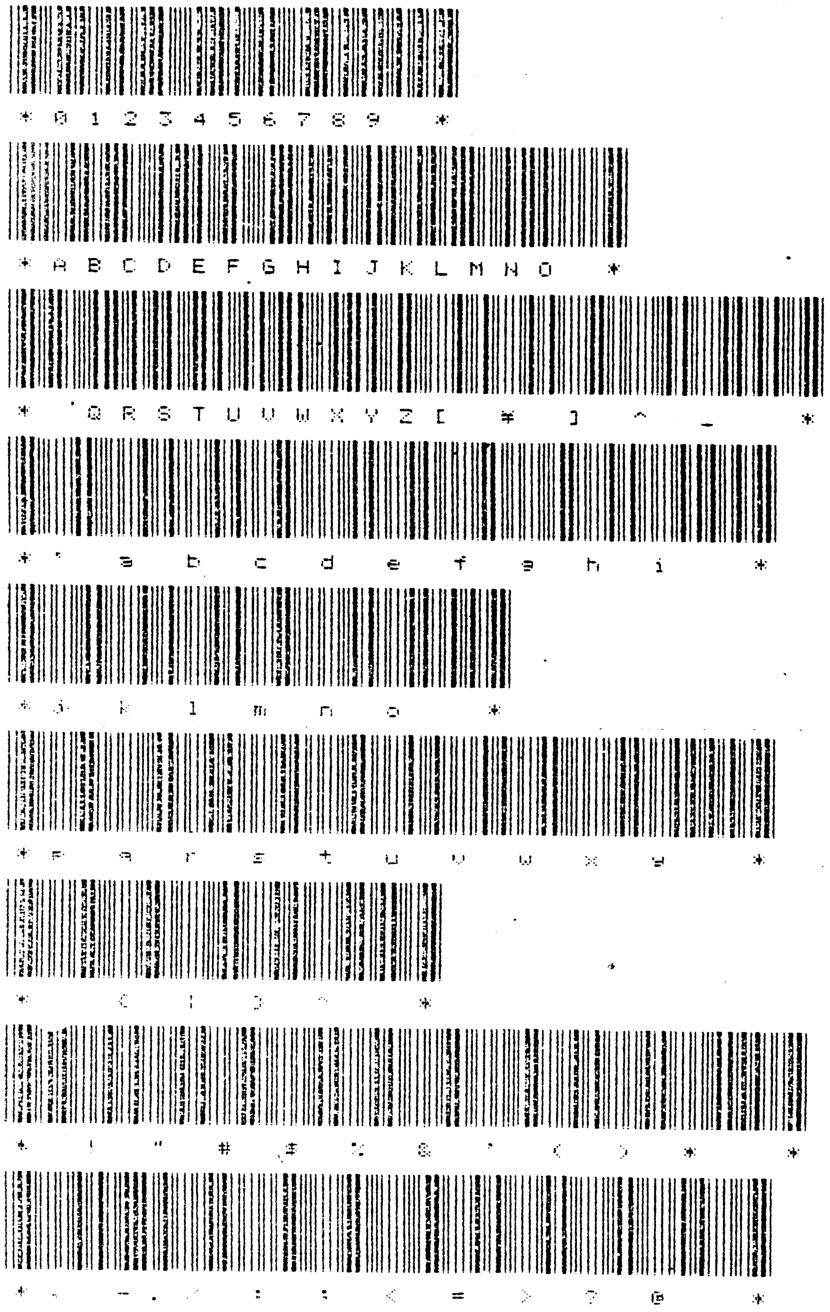


Fig. 12-5 Repetition of Graphic Printing



Bar Code Listing

12-5.

```
     00001                              NAM     BARCOD
     00002                              OPT     PAGE=55
     00003                              TTL     --- BARCODE READER READ SMAPLE ---
     00004                       * FILE NAME  'EXSD'    BY KOIKE
     00005                       *
     00006                       *
     00007                       *          BARCODE READER DECODE PROGRAM
     00008                       *              CREATIVE DATE : 1982/09/30 --- VER 0.1
     00009                       *
     00010                       *
     00011                       *
     00012           0003  A     PORT2  EQU     $3          * MAIN PORT2 ADDRESS
     00013           0006  A     PORT3  EQU     $6          * SLAVE PORT3 ADDRESS
     00014           0007  A     PORT4  EQU     $7          * SLAVE PORT4 ADDRESS
     00015           0008  A     TCSR   EQU     $8          * TIMER CONTROL STATUS REGISTER
     00016           0009  A     FRC    EQU     $9          * FREE RUNNING COUNTER
     00017           000B  A     OCR    EQU     $B          * OUTPUT COMPARE REGISTER
     00018           000D  A     ICR    EQU     $D          * INPUT CAPTURE REGISTER
     00019           007C  A     SIOSTS EQU     $7C         * SLAVE I/O STATUS
     00020           007D  A     MIOSTS EQU     $7D         * MAIN I/O STATUS
     00021           0076  A     MINVAL EQU     118         * MINIMUM WIDTH VALUE
     00022           C350  A     OVRVAL EQU     50000       * OVER FLOW VALUE
     00023                       *
     00024           FF16  A     RV232C EQU     $FF16       * SLAVE RS232C RECOVERY
     00025           FF19  A     SNSCOM EQU     $FF19       * SLAVE COMMUNICATION
     00026                       *
     00027           05E2  A     HKLOAD EQU     $5E2        * HOOK LOAD ADDRESS FOR BARCODE
     00028           063C  A     HKABTD EQU     $63C        * HOOK ABORT ADDRESS FOR BARCODE
     00029           0665  A     DCBTAD EQU     $665        * DCB POINTER FOR BARCODE
     00030           068C  A     ASCFLG EQU     $68C        * LOAD ASCII JUDGE FLAG
     00031           068F  A     OPTNTB EQU     $68F        * OPTION TABLE ADDRESS
     00032           8433  A     ERROR  EQU     $8433       * BASIC ERROR JUMP
     00033           8C70  A     FCERR  EQU     $8C70       * BASIC FC-ERROR JUMP
     00034           A6D0  A     LODCNT EQU     $A6D0       * BASIC CONTINUE LOADING ADDRESS
     00035           A9D8  A     ABTDO  EQU     $A9D8       * BASIC ABORT ADDRESS
     00036                       *
     00037                              TTL     TEST MAIN
     00038                       *
     00039                       *
     00040                       *          TEST MAIN
     00041                       *
     00042A 1700                        ORG     $1700
     00043                       *
     00044A 1700 86 01      A           LDA A   #$01
     00045A 1702 B7 173D    A           STA A   CHKDGT      * CHECK DIGIT JUDGE FLAG
     00046A 1705 86 01      A           LDA A   #$01
     00047A 1707 B7 173E    A           STA A   FULVER      * FULL ASCII JUDGE FLAG
     00048                       *
     00049A 170A BD 1BD3    A           JSR     PONBAR      * POWER ON BARCODE WAND
     00050                       *
     00051A 170D 86 02      A    MNST   LDA A   #$2         * GATE OPEN
     00052A 170F C6 07      A           LDA B   #PORT4
     00053A 1711 BD 1C0D    A           JSR     SPWRIT
     00054                       *
     00055A 1714 7F 17FC    A           CLR     ASCCNT
     00056A 1717 7F 068C  A               CLR    ASCFLG
     00057                         *
     00058A 171A BD 1801  A               JSR    RECBAR    * RECOGNITION BARCODE
     00059                         *
     00060A 171D B7 1738  A               STA A  ANSWER    * ERROR CODE BUFFER
     00061A 1720 25 06 1728               BCS    MN10
     00062A 1722 7F 1739  A               CLR    CARRY     * (C) BUFFER
     00063A 1725 7E 1730  A               JMP    MN20
     00064                         *
     00065A 1728 86 FF    A        MN10   LDA A  #$FF
     00066A 172A B7 1739  A               STA A  CARRY
     00067A 172D 7E 1735  A               JMP    MNED
     00068                         *
     00069A 1730 B6 1738  A        MN20   LDA A  ANSWER
     00070A 1733 27 D8 170D               BEQ    MNST
     00071                         *
     00072A 1735 7E 1735  A        MNED   JMP    *         ERROR END
     00073                         *
     00074A 1738     0001 A        ANSWER RMB    1
     00075A 1739     0001 A        CARRY  RMB    1
     00076                         *
     00077                                TTL    WORK AREA
     00078                         *
     00079                         *
     00080                         *
     00081                         *
     00082A 173A     0001 A        BAR    RMB    1         * BAR BIT PATTERN
     00083A 173B     0001 A        SPACE  RMB    1         * SPACE BIT PATTERN
     00084A 173C     0001 A        DIRECF RMB    1         * SCAN DIRECTION FLAG
     00085A 173D     0001 A        CHKDGT RMB    1         * CHECK DIGIT FLAG
     00086A 173E     0001 A        FULVER RMB    1         * FULL ASCII VERSION JUDGE FLAG
     00087A 173F     0001 A        CHRJDG RMB    1         * INPUT ROUTINE FIRST JUDGE FLAG
     00088A 1740     0002 A        TIMOVC RMB    2         * TIMER OVER FLOW COUNTER
     00089A 1742     0002 A        TIMCT1 RMB    2         * TIMER FIRST COUNTER
     00090A 1744     0002 A        TIMCT2 RMB    2         * TIMER END COUNTER
     00091A 1746     0002 A        TIMCNT RMB    2         * TIMER COUNTER
     00092A 1748     0002 A        TIMSTC RMB    2         *
     00093A 174A     0002 A        STRTMG RMB    2         * START MARGIN
     00094A 174C     0002 A        SUMCHK RMB    2         * CHECK DIGIT SUM
     00095A 174E     0002 A        ZNKBZC RMB    2         * LAST BAR ZERO COUNTER VALUE
     00096A 1750     0002 A        ZNKBOC RMB    2         * LAST BAR ONE COUNTER VALUE
     00097A 1752     0002 A        THRSHB RMB    2         * BAR 1 OR 0 THRESH LEVEL
     00098A 1754     0002 A        ZNKSZC RMB    2         * LAST SPACE ZERO COUNTER VALUE
     00099A 1756     0002 A        ZNKSOC RMB    2         * LAST SPACE ONE COUNTER VALUE
     00100A 1758     0002 A        THRSHS RMB    2         * SPACE 1 OR 0 THRESH LEVEL
     00101A 175A     0001 A        ANSCTB RMB    1         * BUFFER
     00102A 175B     0001 A        FULHNT RMB    1         * FULL ASCII DOUBLE CHARACTER JUDGE
     00103A 175C     0001 A        ASCBF1 RMB    1         * DOUBLE CHARACTER FIRST BUFFER
     00104A 175D     0001 A        ASCBF2 RMB    1         * DOUBLE CHARACTER SECOND BUFFER
     00105A 175E     0001 A        BITCNT RMB    1         * CHARACTER BIT COUNTER
     00106A 175F     0001 A        ERRBF  RMB    1         * ERROR CODE BUFFER
     00107A 1760     0001 A        ZNKOVF RMB    1         * ZENKAI OVER FLOW
     00108A 1761     0002 A        SPWRBF RMB    2         * SLAVE WRITE BUFFER
     00109A 1763     0002 A        ANSTBA RMB    2         * PRE-ANSWER TABLE ADDRESS
     00110A 1765     0002 A        ANSADR RMB    2         *
     00111A 1767     0002  A        ANSASA RMB     2          * ANSWER TABLE ADDRESS
     00112A 1769     0002  A        FULTBA RMB     2          * FULL ASCII TABLE ADDRESS
     00113                          *
     00114A 176B     0048  A        ANSTBL RMB     72         * PRE-ANSWER TABLE
     00115A 17B3     0001  A        ANSCNT RMB     1          * PRE-ANSWER COUNTER
     00116                          *
     00117A 17B4     0048  A        ANSASC RMB     72         * ANSWER TABLE
     00118A 17FC     0001  A        ASCCNT RMB     1          * ANSWER COUNTER
     00119                          *
     00120A 17FD     00    A               FCB     0
     00121A 17FE     0001  A        STDIGT RMB     1          * START CHECK DIGIT
     00122A 17FF     00    A               FCB     0
     00123A 1800     0001  A        EDDIGT RMB     1          * END CHECK DIGIT
     00124                          *
     00125                          *
     00126                                 TTL     RECOGNITION
     00127                          *
     00128                          *
     00129                          *  FUNCTION :  RECOGNITION BARCODE
     00130                          *  CALL     :  JSR   RECBAR
     00131                          *  RETURN   :  (A)= ERROR CODE
     00132                          *                    000: NORMAL
     00133                          *                    100: SCAN SPEED SLOWER
     00134                          *                    101: SCAN SPEED FASTER
     00135                          *                X   102: SW BAD OPERATION -- VER 0.3
     00136                          *                X   103: TIMER OVER FLOW  -- VER 0.3
     00137                          *                    104: BUFFER OVER FLOW
     00138                          *                    105: NOT CODE 39
     00139                          *                    106: CHECK DIGIT ERROR
     00140                          *                    107: FULL ASCII CONVERTION ERROR
     00141                          *               (C)= BREAK STATUS.
     00142                          *                      0: NORMAL
     00143                          *                      1: BREAK
     00144                          *
     00145                          *
     00146                          *
     00147A 1801 96 03     A        RECBAR LDA A   PORT2      * WAND PAPER ON ?
     00148A 1803 85 01     A               BIT A   #1
     00149A 1805 27 13 181A                BEQ     REC50
     00150                          *
     00151A 1807    7B     A               FCB     $7B,$80,$7D * TIM  #$80 MIOSTS
          A 1808    80     A
          A 1809    7D     A
     00152A 180A 27 F5 1801                BEQ     RECBAR
     00153A 180C 0D                        SEC
     00154A 180D 20 01 1810                BRA     REC800
     00155                          *
     00156                          **********    RETURN PROCESS   ***********
     00157                          *
     00158A 180F 0C                 REC700 CLC                * (C) CLEAR
     00159                          *
     00160A 1810 86 00     A        REC800 LDA A   #0         * ANSWER COUNTER CLEAR
     00161A 1812 B7 17FC   A               STA A   ASCCNT
     00162                          *
     00163A 1815 0E                        CLI                * INTERRUPT ENABLE
```

```
 00164                                  *
 00165A 1816 B6 175F  A                       LDA A  ERRBF     * ERROR CODE
 00166                                  *
 00167A 1819 39                         REC900 RTS             * RETURN
 00168                                  *
 00169                                  ***************  MARGIN DETECT  ***************
 00170                                  *
 00171A 181A    7B    A                 REC50  FCB    $7B,$B0,$7D * TIM   #$B0 MIOSTS
      A 181B    B0    A
      A 181C    7D    A
 00172A 181D 27 03 1822                        BEQ    REC60
 00173A 181F 0D                                SEC
 00174A 1820 20 EE 181D                        BRA    REC800
 00175                                  *
 00176A 1822 96 08    A                 REC60  LDA A  TCSR      * ICF CLEAR
 00177A 1824 DC 0D    A                        LDD    ICR
 00178                                  *
 00179A 1826 DC 09    A                        LDD    FRC       * OVF CLEAR
 00180A 1828 FD 1742  A                        STD    TIMCT1
 00181                                  *
 00182A 182B C3 C350  A                        ADDD   #OVRVAL   * OCF CLEAR
 00183A 182E DD 0B    A                        STD    OCR
 00184A 1830 96 08    A                        LDA A  TCSR
 00185A 1832    72    A                        FCB    $72,$00,$B * OIM  #$0,OCR
      A 1833    00    A
      A 1834    0B    A
 00186                                  *
 00187A 1835    72    A                        FCB    $72,$02,$08 * OIM   #2 TCSR * IEDG=1
      A 1836    02    A
      A 1837    08    A
 00188                                  *
 00189A 1838 BD 1AD8  A                        JSR    TIMRED    * MARGIN READ
 00190                                  *
 00191A 183B 25 02 183F                        BCS    REC70     * OVER FLOW ?
 00192A 183D 20 1E 185D                        BRA    REC90
 00193                                  *
 00194A 183F 96 08    A                 REC70  LDA A  TCSR
 00195A 1841 2B 02 1845                        BMI    REC80     * ICF ?
 00196A 1843 20 FA 183F                        BRA    REC70
 00197                                  *
 00198A 1845 DC 0D    A                 REC80  LDD    ICR       * START VALUE
 00199A 1847 FD 1742  A                        STD    TIMCT1
 00200A 184A C3 C350  A                        ADDD   #OVRVAL   * OVER FLOW COUNTER
 00201A 184D DD 0B    A                        STD    OCR
 00202                                  *
 00203A 184F 96 08    A                        LDA A  TCSR      * OCF CLEAR
 00204A 1851    72    A                        FCB    $72,$00,$0B * OIM   #$00,OCR
      A 1852    00    A
      A 1853    0B    A
 00205A 1854    71    A                        FCB    $71,$FD,$08 * AIM   #$FD TCSR      * IEDG=0
      A 1855    FD    A
      A 1856    08    A
 00206                                  *
 00207A 1857 CC FFFF  A                 REC85  LDD    #$FFFF    * OVER FLOW COUNTER SET
 00208A 185A FD 1746  A                        STD    TIMCNT
```

```
ERR   SEQ    LOC  OBJECT        PROGRAM  BARCOD      RECOGNITION

     00209                              *
     00210A 185D FC 1746  A             REC90  LDD    TIMCNT     * MARGIN ENTRY
     00211A 1860 FD 174A  A                    STD    STRTMG
     00212                              *
     00213A 1863 BD 1AD8  A                    JSR    TIMRED     * START BAR READ
     00214                              *
     00215A 1866 24 02 186A                    BCC    REC110
     00216                              *
     00217A 1868 20 B0 181A             REC100 BRA    REC50
     00218                              *
     00219A 186A FC 1746  A             REC110 LDD    TIMCNT
     00220A 186D 83 0076  A                    SUBD   #MINVAL    * 118 SPEED OVER ?
     00221A 1870 25 A8 181A                    BCS    REC50
     00222                              *
     00223A 1872 FC 174A  A                    LDD    STRTMG     * MARGIN CHECK
     00224A 1875 04                            LSRD              * (MARGIN)/16>=(START BAR WIDTH)
     00225A 1876 04                            LSRD
     00226A 1877 04                            LSRD
     00227A 1878 04                            LSRD
     00228A 1879 B3 1746  A                    SUBD   TIMCNT
     00229A 187C 25 9C 181A                    BCS    REC50
     00230                              *
     00231                              ************  STRT CODE ( * ) READ   *************
     00232                              *
     00233A 187E FC 1746  A                    LDD    TIMCNT     * INITIAL VALUE ENTRY
     00234A 1881 FD 174E  A                    STD    ZNKBZC     * INITIAL NARROW BAR
     00235A 1884 FD 1754  A                    STD    ZNKSZC     *
     00236A 1887 05                            ASLD
     00237A 1888 FD 1750  A                    STD    ZNKBOC     * INITIAL WIDE VALUE ( X 2 )
     00238A 188B FD 1756  A                    STD    ZNKSOC     *
     00239                              *
     00240A 188E F3 1746  A                    ADDD   TIMCNT     * THRESH LEVEL ENTRY ( X 1.5 )
     00241A 1891 04                            LSRD
     00242A 1892 FD 1752  A                    STD    THRSHB     * THRESH LEVEL
     00243A 1895 FD 1758  A                    STD    THRSHS
     00244A 1898 86 FF    A                    LDA A  #$FF       * PRE-ANSWER COUNTER INITIAL
     00245A 189A B7 17B3  A                    STA A  ANSCNT
     00246                              *
     00247A 189D 86 08    A                    LDA A  #8         * REST 8 BIT OF START CODE
     00248A 189F BD 1B0B  A                    JSR    DTTOBT
     00249                              *
     00250A 18A2 27 02 18A6                    BEQ    REC120
     00251A 18A4 20 C2 1868                    BRA    REC100
     00252                              *
     00253A 18A6 B6 173A  A             REC120 LDA A  BAR        * NORMAL DIRECTION CHECK
     00254A 18A9 81 06    A                    CMP A  #6
     00255A 18AB 26 0C 18B9                    BNE    REC130
     00256A 18AD B6 173B  A                    LDA A  SPACE
     00257A 18B0 81 08    A                    CMP A  #8
     00258A 18B2 26 B4 1868                    BNE    REC100     * NOT START CODE
     00259A 18B4 7F 173C  A                    CLR    DIRECF     * L TO R DIRECTION SET
     00260A 18B7 20 0E 18C7                    BRA    REC140
     00261                              *
     00262A 18B9 81 0C    A             REC130 CMP A  #$C        * REVERSE DIRECTION CHECK
     00263A 18BB 26 AB 1868                    BNE    REC100     * NOT START CODE
```

```
    00264A 18BD B6 173B  A                    LDA A   SPACE
    00265A 18C0 81 01    A                    CMP A   #1
    00266A 18C2 26 A4 1868                    BNE     REC100
    00267A 18C4 B7 173C  A                    STA A   DIRECF    * R TO L DIRECTION SET
    00268                              *
    00269A 18C7    7B    A             REC140 FCB     $7B,$B0,$7D * TIM    #$B0 MIOSTS      * BREAK C
         A 18C8    B0    A
         A 18C9    7D    A
    00270A 18CA 27 04 18D0                    BEQ     REC150
    00271A 18CC 0D                            SEC
    00272A 18CD 7E 1810  A                    JMP     REC800
    00273                              *
    00274                              ***************   DATA READ - IN   ***************
    00275                              *
    00276A 18D0 0F                     REC150 SEI               * INTERRUPT MASK DISABLE
    00277                              *
    00278A 18D1 CC 0000  A                    LDD     #0
    00279A 18D4 FD 174C  A                    STD     SUMCHK    * CHECK DIGIT SUM AREA CLEAR
    00280A 18D7 86 FF    A                    LDA A   #$FF
    00281A 18D9 B7 17FE  A                    STA A   STDIGT    * START DIGIT INITIA
    00282A 18DC CE 176B  A                    LDX     #ANSTBL   * PRE-ANSWER TABLE ADDRESS
    00283A 18DF FF 1763  A                    STX     ANSTBA
    00284                              *
    00285A 18E2 BD 1AD8  A             REC160 JSR     TIMRED    * CHARACTER GAP READ
    00286                              *
    00287A 18E5 24 08 18EF                    BCC     REC170    * OVER FLOW ?
    00288A 18E7 86 64    A                    LDA A   #100      * SCAN SPEED SLOWER
    00289A 18E9 B7 175F  A                    STA A   ERRBF
    00290A 18EC 7E 180F  A                    JMP     REC700
    00291                              *
    00292A 18EF 86 09    A             REC170 LDA A   #9        * 1 CHARACTER DATA BIT CONVERT
    00293A 18F1 BD 1B03  A                    JSR     DTTOBT
    00294                              *
    00295A 18F4 B7 175F  A                    STA A   ERRBF
    00296A 18F7 27 03 18FC                    BEQ     REC175
    00297A 18F9 7E 180F  A                    JMP     REC700
    00298                              *
    00299A 18FC B6 17B3  A             REC175 LDA A   ANSCNT    * BUFFER OVER CHECK
    00300A 18FF 81 49    A                    CMP A   #73
    00301A 1901 25 08 190B                    BCS     REC180
    00302A 1903 86 68    A                    LDA A   #104      * BUFFER OVER FLOW ERROR
    00303A 1905 B7 175F  A                    STA A   ERRBF
    00304A 1908 7E 180F  A                    JMP     REC700
    00305                              *
    00306                              *************  BIT TO ASCII CODE CONVERT  **********
    00307                              *
    00308A 190B CE 1C5E  A             REC180 LDX     #SPCTBL   * SPACE TABLE
    00309A 190E F6 173B  A                    LDA B   SPACE     * (X)=(X)+(SPACE)X4
    00310A 1911 58                            ASL B
    00311A 1912 58                            ASL B
    00312A 1913 3A                            ABX
    00313                              *
    00314A 1914 B6 173C  A                    LDA A   DIRECF    * DIRECTION  L TO R ?
    00315A 1917 27 02 191B                    BEQ     REC190
    00316A 1919 08                            INX
    00317A 1917 08                          INX
    00318                            *
    00319A 191B EC 00     A          REC190 LDD     ,X         * SPACE TABLE DATA
    00320A 191D 27 0B 192A                  BEQ     REC200
    00321A 191F 2B 36 1957                  BMI     REC280     * SPECIAL CHARACTER
    00322A 1921    18     A                 FCB     $18        * XGDX          * (D) TO (X)
    00323A 1922 F6 173A   A                 LDA B   BAR
    00324A 1925 3A                          ABX                * (X)=(X)+(BAR)
    00325A 1926 A6 00     A                 LDA A   ,X         * BAR TABLE DATA
    00326A 1928 26 08 1932                  BNE     REC210
    00327                            *
    00328A 192A 86 69     A          REC200 LDA A   #105       * NOT CODE 39 ERROR
    00329A 192C B7 175F   A                 STA A   ERRBF
    00330A 192F 7E 180F   A                 JMP     REC700
    00331                            *
    00332                            ***************  CHECK DIGIT CALCULATE  **********
    00333                            *
    00334A 1932 16                   REC210 TAB                * (A) TO (B)
    00335A 1933 C1 41     A                 CMP B   #$41       * ALPHA ?
    00336A 1935 24 1C 1953                  BCC     REC250
    00337A 1937 C1 30     A                 CMP B   #$30       * NUMERIC ?
    00338A 1939 24 14 194F                  BCC     REC240
    00339A 193B C1 20     A                 CMP B   #$20       * SP ?
    00340A 193D 26 04 1943                  BNE     REC220
    00341A 193F C6 26     A                 LDA B   #38        * SP DIGIT
    00342A 1941 20 1B 195E                  BRA     REC290
    00343                            *
    00344A 1943 C1 2D     A          REC220 CMP B   #$2D       * - ?
    00345A 1945 26 04 194B                  BNE     REC230
    00346A 1947 C6 24     A                 LDA B   #36        * - DIGIT
    00347A 1949 20 13 195E                  BRA     REC290
    00348                            *
    00349A 194B C6 25     A          REC230 LDA B   #37        * . DIGIT
    00350A 194D 20 0F 195E                  BRA     REC290
    00351                            *
    00352A 194F C0 30     A          REC240 SUB B   #$30       * 0-9 DIGIT
    00353A 1951 20 0B 195E                  BRA     REC290
    00354                            *
    00355A 1953 C0 37     A          REC250 SUB B   #$37       * A-Z DIGIT
    00356A 1955 20 07 195E                  BRA     REC290
    00357                            *
    00358A 1957 84 7F     A          REC280 AND A   #$7F       * SPECIAL CHARACTER
    00359A 1959 7D 173A   A                 TST     BAR
    00360A 195C 26 CC 192A                  BNE     REC200     * NOT CODE 39 ERROR
    00361                            *
    00362A 195E 81 2A     A          REC290 CMP A   #$2A       * END CODE (*) ?
    00363A 1960 27 1E 1980                  BEQ     REC310
    00364                            *
    00365A 1962 FE 1763   A                 LDX     ANSTBA
    00366A 1965 A7 00     A                 STA A   ,X         * ANSWER ASCII ENTRY
    00367A 1967 08                          INX
    00368A 1968 FF 1763   A                 STX     ANSTBA     * NEXT ADDRESS SAVE
    00369A 196B FE 174C   A                 LDX     SUMCHK     * CHECK DIGIT SUM
    00370A 196E 3A                          ABX
    00371A 196F FF 174C   A                 STX     SUMCHK
    00372                          *
    00373A 1972 B6 17FE  A                LDA A  STDIGT     * START DIGIT ?
    00374A 1975 2A 03 197A                BPL    REC300
    00375A 1977 F7 17FE  A                STA B  STDIGT
    00376                          *
    00377A 197A F7 1800  A         REC300 STA B  EDDIGT     * NEW DIGIT ENTRY
    00378A 197D 7E 18E2  A                JMP    REC160
    00379                          *
    00380                          **********  DATA ARRANGEMENT AND CONVERTION  **********
    00381                          *
    00382A 1980 7A 17B3  A         REC310 DEC    ANSCNT     * LAST (*) BUN COUNTER DECREMENT
    00383                          *
    00384A 1983 0E                        CLI               * INTERRUPT ENABLE
    00385                          *
    00386A 1984 B6 173D  A                LDA A  CHKDGT
    00387A 1987 27 32 19BB                BEQ    REC340
    00388                          *
    00389A 1989 7A 17B3  A                DEC    ANSCNT     * CHECK DIGIT BUN COUNTER DECREMENT
    00390A 198C B6 173C  A                LDA A  DIRECF     * DIRECTION CHECK
    00391A 198F 26 0B 199C                BNE    REC320
    00392                          *
    00393A 1991 FC 174C  A                LDD    SUMCHK     * L TO R DIRECTION
    00394A 1994 B3 17FF  A                SUBD   EDDIGT-1   * CHECK SUM
    00395A 1997 FD 174C  A                STD    SUMCHK
    00396A 199A 20 0F 19AB                BRA    REC330
    00397                          *
    00398A 199C FC 174C  A         REC320 LDD    SUMCHK     * R TO L DIRECTION
    00399A 199F B3 17FD  A                SUBD   STDIGT-1   * CHECK SUM
    00400A 19A2 FD 174C  A                STD    SUMCHK
    00401A 19A5 B6 17FE  A                LDA A  STDIGT     * LAST DIGIT SHITEI
    00402A 19A8 B7 1800  A                STA A  EDDIGT
    00403                          *
    00404A 19AB BD 1BC1  A         REC330 JSR    DGTCAL     * CHECK DIGIT CALCULATE
    00405                          *
    00406A 19AE B1 1800  A                CMP A  EDDIGT     * DIGIT OK ?
    00407A 19B1 27 08 19BB                BEQ    REC340
    00408                          *
    00409A 19B3 86 6A    A                LDA A  #106       * CHECK DIGIT ERROR
    00410A 19B5 B7 179F  A                STA A  ERRBF
    00411A 19B8 7E 180F  A                JMP    REC700
    00412                          *
    00413                          ***************  ANSWER ASCII REARRANGEMENT  **********
    00414                          *
    00415A 19BB CE 176B  A         REC340 LDX    #ANSTBL    * PRE-ANSWER TABLE
    00416A 19BE CC 17B4  A                LDD    #ANSASC    * ANSWER TABLE
    00417A 19C1 FD 1767  A                STD    ANSASA
    00418                          *
    00419A 19C4 B6 173C  A                LDA A  DIRECF     * DIRECTION FLAG
    00420A 19C7 27 0B 19D4                BEQ    REC360
    00421A 19C9 F6 17B3  A                LDA B  ANSCNT     * R TO L DIRECTION
    00422A 19CC B6 173D  A                LDA A  CHKDGT
    00423A 19CF 26 02 19D3                BNE    REC350
    00424A 19D1 C0 01    A                SUB B  #1         * NONE CHECK DIGIT
    00425                          *
    00426A 19D3 3A                 REC350 ABX               * TOP DATA ADDRESS
    00427                           *
    00428A 19D4 FF 1763   A         REC360 STX     ANSTBA     * SOURCE ADDRESS
    00429A 19D7 B6 17B3   A                LDA A   ANSCNT
    00430A 19DA B7 175A   A                STA A   ANSCTB     * TRANSFER COUNTER
    00431                           *
    00432A 19DD B6 175A   A         REC370 LDA A   ANSCTB
    00433A 19E0 27 28 1A0A                 BEQ     REC400
    00434                           *
    00435A 19E2 FE 1763   A                LDX     ANSTBA     * SOURCE ADDRESS
    00436A 19E5 A6 00     A                LDA A   ,X
    00437A 19E7 FE 1767   A                LDX     ANSASA     * DESTINATION ADDRESS
    00438A 19EA A7 00     A                STA A   ,X
    00439A 19EC 08                         INX
    00440A 19ED FF 1767   A                STX     ANSASA
    00441                           *
    00442A 19F0 B6 173C   A                LDA A   DIRECF
    00443A 19F3 26 09 19FE                 BNE     REC380
    00444A 19F5 FE 1763   A                LDX     ANSTBA     * L TO R DIRECTION
    00445A 19F8 08                         INX                * NEXT ADDRESS
    00446A 19F9 FF 1763   A                STX     ANSTBA
    00447A 19FC 20 07 1A05                 BRA     REC390
    00448                           *
    00449A 19FE FE 1763   A         REC380 LDX     ANSTBA     * R TO L DIRECTION
    00450A 1A01 09                         DEX                * NEXT ADDRESS
    00451A 1A02 FF 1763   A                STX     ANSTBA
    00452                           *
    00453A 1A05 7A 175A   A         REC390 DEC     ANSCTB
    00454A 1A08 20 D3 19DD                 BRA     REC370
    00455                           *
    00456A 1A0A B6 173E   A         REC400 LDA A   FULVER
    00457A 1A0D 26 19 1A28                 BNE     REC500
    00458                           *
    00459                           ***************   END PROCESS   ***************
    00460                           *
    00461A 1A0F B6 17B3   A                LDA A   ANSCNT     * PRE-ANSWER COUNTER
    00462A 1A12 B7 17FC   A                STA A   ASCCNT     * ANSWER COUNTER
    00463                           *
    00464A 1A15 BD 13A9   A         REC410 JSR     BEEPOK     * OK BEEP
    00465                           *
    00466A 1A18 24 06 1A20                 BCC     REC420
    00467A 1A1A 7F 175F   A                CLR     ERRBF      * BREAK
    00468A 1A1D 7E 1810   A                JMP     REC800
    00469                           *
    00470A 1A20 7F 173F   A         REC420 CLR     CHRJDG     * NORMAL END
    00471A 1A23 4F                         CLR A              * (A) CLEAR
    00472A 1A24 0C                         CLC                * (C) CLEAR
    00473A 1A25 7E 1819   A                JMP     REC900
    00474                           *
    00475                           ***************   FULL ASCII CHECK   ***************
    00476                           *
    00477A 1A28 CE 17B4   A         REC500 LDX     #ANSASC
    00478A 1A2B FF 1765   A                STX     ANSADR     * SOURCE ADDRESS
    00479A 1A2E FF 1767   A                STX     ANSASA     * DESTINATION ADDRESS
    00480A 1A31 7F 175B   A                CLR     FULHNT
    00481A 1A34 7F 17FC   A                CLR     ASCCNT     * ANSWER COUNTER CLEAR
```

```
00482                              *
00483A 1A37 B6 17B3   A            REC510 LDA A  ANSCNT
00484A 1A3A 81 00     A                   CMP A  #0
00485A 1A3C 2F D7 1A15                    BLE    REC410     * END
00486                              *
00487A 1A3E 7A 17B3   A                   DEC    ANSCNT     * SOURCE COUNTER DECREMENT
00488A 1A41 B6 175B   A                   LDA A  FULHNT     * DOUBLE CHARACTER JUDGE
00489A 1A44 27 08 1A4E                    BEQ    REC520
00490A 1A46 B6 175D   A                   LDA A  ASCBF2     * SPECIAL CODE ( $,+,/,% )
00491A 1A49 B7 175C   A                   STA A  ASCBF1
00492A 1A4C 20 0C 1A5A                    BRA    REC530
00493                              *
00494A 1A4E FE 1765   A            REC520 LDX    ANSADR     * SOURCE ADDRESS
00495A 1A51 A6 00     A                   LDA A  ,X
00496A 1A53 B7 175C   A                   STA A  ASCBF1
00497A 1A56 08                            INX
00498A 1A57 FF 1765   A                   STX    ANSADR
00499                              *
00500A 1A5A 81 24     A            REC530 CMP A  #$24       * $
00501A 1A5C 26 08 1A66                    BNE    REC540
00502A 1A5E CE 1D9E   A                   LDX    #FULASC
00503A 1A61 FF 1769   A                   STX    FULTBA
00504A 1A64 20 27 1A8D                    BRA    REC580
00505                              *
00506A 1A66 81 2F     A            REC540 CMP A  #$2F       * /
00507A 1A68 26 08 1A72                    BNE    REC550
00508A 1A6A CE 1DB8   A                   LDX    #FULASC+26
00509A 1A6D FF 1769   A                   STX    FULTBA
00510A 1A70 20 1B 1A8D                    BRA    REC580
00511                              *
00512A 1A72 81 2B     A            REC550 CMP A  #$2B       * +
00513A 1A74 26 08 1A7E                    BNE    REC560
00514A 1A76 CE 1DD2   A                   LDX    #FULASC+52
00515A 1A79 FF 1769   A                   STX    FULTBA
00516A 1A7C 20 0F 1A8D                    BRA    REC580
00517                              *
00518A 1A7E 81 25     A            REC560 CMP A  #$25       * %
00519A 1A80 27 05 1A87                    BEQ    REC570
00520A 1A82 7F 175B   A                   CLR    FULHNT
00521A 1A85 20 3F 1AC6                    BRA    REC610
00522                              *
00523A 1A87 CE 1DEC   A            REC570 LDX    #FULASC+78 * %
00524A 1A8A FF 1769   A                   STX    FULTBA
00525                              *
00526A 1A8D B6 17B3   A            REC580 LDA A  ANSCNT     * SOURCE DATA END CHECK
00527A 1A90 81 00     A                   CMP A  #0
00528A 1A92 2F 32 1AC6                    BLE    REC610     * END DATA ENTRY
00529                              *
00530A 1A94 FE 1765   A                   LDX    ANSADR     * NEXT DATA PRE-READ
00531A 1A97 A6 00     A                   LDA A  ,X
00532A 1A99 B7 175D   A                   STA A  ASCBF2
00533A 1A9C 08                            INX
00534A 1A9D FF 1765   A                   STX    ANSADR     * NEXT SOURCE ADDRESS
00535                              *
00536A 1AA0 81 41     A                   CMP A  #$41       * ALPHA ?
```

```
SEQ    LOC  OBJECT        PROGRAM  BARCOD      RECOGNITION

00537A 1AA2 25 1D 1AC1                    BCS    REC600
00538                            *
00539A 1AA4 80 41     A                   SUB A  #$41
00540A 1AA6 16                            TAB
00541A 1AA7 7F 175B   A                   CLR    FULHNT
00542A 1AAA FE 1769   A                   LDX    FULTBA
00543A 1AAD 3A                            ABX                * DATA ADDRESS
00544                            *
00545A 1AAE A6 00     A                   LDA A  ,X          * CONVERTION DATA
00546A 1AB0 81 FF     A                   CMP A  #$FF
00547A 1AB2 26 08 1ABC                    BNE    REC590
00548                            *
00549A 1AB4 86 68     A                   LDA A  #107        * FULL ASCII ERROR
00550A 1AB6 B7 175F   A                   STA A  ERRBF
00551A 1AB9 7E 180F   A                   JMP    REC700
00552                            *
00553A 1ABC 7A 17B3   A          REC590 DEC    ANSCNT      * SOURCE COUNTER DECREMENT
00554A 1ABF 20 08 1AC9                    BRA    REC620
00555                            *
00556A 1AC1 86 01     A          REC600 LDA A  #1          * SINGLE DATA FLAG SET
00557A 1AC3 B7 175B   A                 STA A  FULHNT
00558                            *
00559A 1AC6 B6 175C   A          REC610 LDA A  ASCBF1      * PRE-READ DATA
00560                            *
00561A 1AC9 FE 1767   A          REC620 LDX    ANSASA      * FULL ASCII ENTRY
00562A 1ACC A7 00     A                 STA A  ,X
00563A 1ACE 08                          INX                * DESTINATION NEXT ADDRESS
00564A 1ACF FF 1767   A                 STX    ANSASA
00565A 1AD2 7C 17FC   A                 INC    ASCCNT      * DESTINATION COUNTER RENEW
00566A 1AD5 7E 1A37   A                 JMP    REC510
00567                            *
00568                                   TTL    SUBROUTIN
00569                            *
00570                            *
00571                            *   FUNCTION :  BAR OR SPACE WIDTH TIMER VALUE READ
00572                            *   CALL     :  JSR   TIMRED
00573                            *   RETURN   :  (C)= RETURN STATUS
00574                            *                         0: NORMAL
00575                            *                         1: OVER FLOW
00576                            *               TIMCNT= TIMER VALUE
00577                            *
00578                            *
00579A 1AD8 96 08     A          TIMRED LDA A  TCSR        * TIMER CONTROLL STATUS REGISTER
00580A 1ADA 85 40     A                 BIT A  #$40        * OVER FLOW CHECK
00581A 1ADC 26 24 1B02                  BNE    TIM100
00582A 1ADE 85 80     A                 BIT A  #$80        * ICF CHECK
00583A 1AE0 27 F6 1AD8                  BEQ    TIMRED
00584                            *
00585A 1AE2 DC 0D     A                 LDD    ICR         * TIMER READ
00586A 1AE4 FD 1744   A                 STD    TIMCT2
00587A 1AE7 B3 1742   A                 SUBD   TIMCT1
00588A 1AEA FD 1746   A                 STD    TIMCNT      * TIMER VALUE ENTRY
00589                            *
00590A 1AED FC 1744   A                 LDD    TIMCT2      * START VALUE RENEW
00591A 1AF0 FD 1742   A                 STD    TIMCT1
```

```
      00592                                  *
      00593A 1AF3 C3 C350   A                        ADDD    #OVRVAL     * OVER FLOW COUNTER SET
      00594A 1AF6 DD 0B     A                        STD     OCR
      00595A 1AF8 96 08     A                        LDA A   TCSR
      00596A 1AFA    72     A                        FCB     $72,$00,$0B * OIM   #$00,OCR
           A 1AFB    00     A
           A 1AFC    0B     A
      00597                                  *
      00598A 1AFD    75     A                        FCB     $75,$02,$08 * EIM   #2 TCSR * EDGE CONVERT
           A 1AFE    02     A
           A 1AFF    08     A
      00599A 1B00 0C                                 CLC                 * CARRY CLEAR
      00600                                  *
      00601A 1B01 39                         TIM900  RTS                 * RETURN
      00602                                  *
      00603A 1B02 CC FFFF   A                TIM100  LDD     #$FFFF      * OVER FLOW
      00604A 1B05 FD 1746   A                        STD     TIMCNT
      00605A 1B08 0D                                 SEC                 * (C) SET
      00606A 1B09 20 F6 1B01                         BRA     TIM900
      00607                                  *
      00608                                  *
      00609                                  *   FUNCTION :  BAR DATA READ AND BIT CONVERT
      00610                                  *   CALL     :  JSR    DTTOBT
      00611                                  *                      (A)= BIT NUMBER
      00612                                  *   RETURN   :  (A)= RETURN STATUS
      00613                                  *                         0: NORMAL
      00614                                  *                       100: SCAN SPEED SLOWER
      00615                                  *                       101: SCAN SPEED FASTER
      00616                                  *               BAR   = BAR BIT ANSWER
      00617                                  *               SPACE = SPACE BIT ANSWER
      00618                                  *
      00619                                  *
      00620A 1B0B B7 175E   A                DTTOBT  STA A   BITCNT      * BIT COUNTER
      00621A 1B0E 7F 173A   A                        CLR     BAR
      00622A 1B11 7F 173B   A                        CLR     SPACE
      00623                                  *
      00624A 1B14 B6 175E   A                DTT10   LDA A   BITCNT      * BIT END CHECK
      00625A 1B17 26 05 1B1E                         BNE     DTT20
      00626                                  *
      00627A 1B19 7C 17B3   A                        INC     ANSCNT      * END  ENTRY CHARACTER RENEW
      00628A 1B1C 4F                                 CLR A               * NORMAL RETURN
      00629                                  *
      00630A 1B1D 39                         DTT900  RTS                 * RETURN
      00631                                  *
      00632A 1B1E B6 175E   A                DTT20   LDA A   BITCNT      * BAR , SPACE CHECK
      00633A 1B21 85 01     A                        BIT A   #$1
      00634A 1B23 26 05 1B2A                         BNE     DTT30
      00635A 1B25 78 173B   A                        ASL     SPACE       * WHEN SPACE
      00636A 1B28 20 03 1B2D                         BRA     DTT40
      00637                                  *
      00638A 1B2A 78 173A   A                DTT30   ASL     BAR         * WHEN BAR
      00639                                  *
      00640A 1B2D BD 1AD8   A                DTT40   JSR     TIMRED      * WIDTH READ
      00641                                  *
      00642A 1B30 24 04 1B36                         BCC     DTT50
```

```
     00643                                *
     00644A 1B32 86 64     A                     LDA A   #100      * SCAN SPEED SLOWER ERROR
     00645A 1B34 20 E7 1B1D                      BRA     DTT900
     00646                                *
     00647A 1B36 FC 1746   A              DTT50  LDD     TIMCNT
     00648A 1B39 83 0076   A                     SU3D    #MINVAL   * 118  SPEED OVER CHECK
     00649A 1B3C 24 04 1B42                      BCC     DTT60
     00650                                *
     00651A 1B3E 86 65     A                     LDA A   #101      * SCAN SPEED FASTER
     00652A 1B40 20 DB 1B1D                      BRA     DTT900
     00653                                *
     00654A 1B42 B6 175E   A              DTT60  LDA A   BITCNT
     00655A 1B45 85 01     A                     BIT A   #$1       * BAR OR SPACE CHECK
     00656A 1B47 26 2E 1B77                      BNE     DTT80     * BAR
     00657                                *
     00658                                **************    SPACE    ******************
     00659                                *
     00660A 1B49 FC 1746   A                     LDD     TIMCNT    *
     00661A 1B4C B3 1758   A                     SUBD    THRSHS    * COMPARE WITH SPACE THRESH
     00662A 1B4F 22 0F 1B60                      BHI     DTT70
     00663                                *
     00664A 1B51 FC 1746   A                     LDD     TIMCNT    * WHEN SPACE 0
     00665A 1B54 FD 1754   A                     STD     ZNKSZC    * LAST SPACE ZERO COUNTER ENTRY
     00666A 1B57 F3 1756   A                     ADDD    ZNKSOC
     00667A 1B5A 04                              LSRD              * / 2
     00668A 1B5B FD 1758   A                     STD     THRSHS    * NEW SPACE THRESH
     00669A 1B5E 20 43 1BA3                      BRA     DTT110
     00670                                *
     00671A 1B60 FC 1746   A              DTT70  LDD     TIMCNT    * WHEN SPACE 1
     00672A 1B63 FD 1756   A                     STD     ZNKSOC    * LAST SPACE ONE COUNTER ENTRY
     00673A 1B66 F3 1754   A                     ADDD    ZNKSZC
     00674A 1B69 04                              LSRD              * / 2
     00675A 1B6A FD 1758   A                     STD     THRSHS    * NEW SPACE THRESH
     00676                                *
     00677A 1B6D B6 1738   A                     LDA A   SPACE     * SPACE BIT SET
     00678A 1B70 8A 01     A                     ORA A   #1
     00679A 1B72 B7 1738   A                     STA A   SPACE
     00680A 1B75 20 2C 1BA3                      BRA     DTT110
     00681                                *
     00682                                **************    BAR    *****************
     00683                                *
     00684A 1B77 FC 1746   A              DTT80  LDD     TIMCNT
     00685A 1B7A B3 1752   A                     SU3D    THRSHB    * COMPARE WITH BAR THRESH
     00686A 1B7D 22 0F 1B8E                      BHI     DTT90     *
     00687                                *
     00688A 1B7F FC 1746   A                     LDD     TIMCNT    * WHEN BAR 0
     00689A 1B82 FD 174E   A                     STD     ZNKBZC    * LAST BAR ZERO COUNTER ENTRY
     00690A 1B85 F3 1750   A                     ADDD    ZNKBOC
     00691A 1B88 04                              LSRD              * / 2
     00692A 1B89 FD 1752   A                     STD     THRSHB    * NEW BAR THRESH
     00693A 1B8C 20 15 1BA3                      BRA     DTT110
     C0694                                *
     00695A 1B8E FC 1746   A              DTT90  LDD     TIMCNT    * WHEN BAR 1
     00696A 1B91 FD 1750   A                     STD     ZNKBOC    * LAST BAR ONE COUNTER ENTRY
     00697A 1B94 F3 174E   A                     ADDD    ZNKBZC
```

```
ERR   SEQ    LOC  OBJECT       PROGRAM  BARCOD     SUBROUTIN

      00698A 1B97 04                          LSRD              * / 2
      00699A 1B98 FD 1752  A                  STD    THRSHB     * NEW BAR THRESH
      00700                            *
      00701A 1B9B B6 173A  A                  LDA A  BAR        * BAR BIT SET
      00702A 1B9E 8A 01    A                  ORA A  #1
      00703A 1BA0 B7 173A  A                  STA A  BAR
      00704                            *
      00705A 1BA3 7A 175E  A           DTT110 DEC    BITCNT
      00706A 1BA6 7E 1B14  A                  JMP    DTT10
      00707                            *
      00708                            *
      00709                            *  FUNCTION :  OK BEEP ON
      00710                            *  CALL     :  JSR   BEEPOK
      00711                            *  RETURN   :  (C)= BREAK STATUS
      00712                            *                    0: NORMAL
      00713                            *                    1: BREAK
      00714                            *
      00715                            *
      00716A 1BA9 BD 1C53  A           BEEPOK JSR    SRWINT     * SLAVE SUPER VISOR MASK OPEN
      00717                            *
      00718A 1BAC 86 30    A                  LDA A  #$30       * SLAVE BEEP COMMAND
      00719A 1BAE BD FF19  A                  JSR    SNSCOM
      00720                            *
      00721A 1BB1 86 1C    A                  LDA A  #$1C       * SOUND LEVEL
      00722A 1BB3 BD FF19  A                  JSR    SNSCOM
      00723                            *
      00724A 1BB6 86 01    A                  LDA A  #$1        * SOUND LENGTH
      00725A 1BB8 BD FF19  A                  JSR    SNSCOM
      00726                            *
      00727A 1BBB 25 03 1BCD                  BCS    BEP900
      00728A 1BBD BD FF16  A                  JSR    RV232C     * SLAVE COMMUNICATION INITIAL
      00729                            *
      00730A 1BC0 39                   BEP900 RTS
      00731                            *
      00732                            *
      00733                            *  FUNCTION :  CHECK DIGIT CALCULATE
      00734                            *  CALL     :  JSR   DGTCAL
      00735                            *                    SUMCHK,+1=CHECK DIGIT SUM AREA
      00736                            *  RETURN   :  (A)= CHECK DIGIT
      00737                            *
      00738                            *
      00739A 1BC1 FC 174C  A           DGTCAL LDD    SUMCHK     * SUM CHECK
      00740A 1BC4 83 002B  A                  SUBD   #43
      00741A 1BC7 25 05 1BCE                  BCS    DGT10
      00742                            *
      00743A 1BC9 FD 174C  A                  STD    SUMCHK
      00744A 1BCC 20 F3 1BC1                  BRA    DGTCAL
      00745                            *
      00746A 1BCE C3 002B  A           DGT10  ADDD   #43
      00747A 1BD1 17                          TBA               * REST  (B) TO (A)
      00748A 1BD2 39                          RTS
      00749                            *
      00750                                   TTL    I/O SUBROUTINE
      00751                            *
      00752                            *
```

```
   00753                             *  FUNCTION :  BARCODE WAND POWER ON
   00754                             *  CALL     :  JSR   PONBAR
   00755                             *  RETURN   :  (C)= RETURN STATUS
   00756                             *                      0: NORMAL
   00757                             *                      1: BREAK
   00758                             *
   00759                             *
   00760A 18D3 86 20     A           PONBAR LDA A  #$20      * BARCODE WAND POWER ON
   00761A 18D5 C6 06     A                  LDA B  #PORT3
   00762A 18D7 BD 1C0D   A                  JSR    SPWRIT
   00763                             *
   00764A 18DA 25 06 18E2                   BCS    PON900    * BREAK CHECK
   00765A 18DC    72     A                  FCB    $72,$40,$7C * OIM   #$40 SIOSTS     * POWER ON STATUS
        A 18DD    40     A
        A 18DE    7C     A
   00766                             *
   00767A 18DF BD FF16   A                  JSR    RV232C    * SLAVE RS232C RECOVERY
   00768                             *
   00769A 18E2 39                    PON900 RTS              * RETURN
   00770                             *
   00771                             *
   00772                             *  FUNCTION :  BARCODE WAND POWER OFF
   00773                             *  CALL     :  (C)= RETURN STATUS
   00774                             *                      0: NORMAL
   00775                             *                      1: BREAK
   00776                             *
   00777                             *
   00778A 18E3 86 20     A           POFBAR LDA A  #$20      * BARCODE WAND POWER OFF
   00779A 18E5 C6 06     A                  LDA B  #PORT3
   00780A 18E7 CA 80     A                  ORA B  #$80
   00781A 18E9 BD 1C0D   A                  JSR    SPWRIT
   00782                             *
   00783A 18EC 25 06 18F4                   BCS    POF900    * BREAK CHECK
   00784A 18EE    71     A                  FCB    $71,$BF,$7C * AIM   #$BF SIOSTS     * POWER ON STATUS
        A 18EF    BF     A
        A 18F0    7C     A
   00785                             *
   00786A 18F1 BD FF16   A                  JSR    RV232C    * SLAVE RS232C RECOVERY
   00787                             *
   00788A 18F4 39                    POF900 RTS              * RETURN
   00789                             *
   00790                             *
   00791                             *  FUNCTION :  SLAVE PORT READ
   00792                             *  CALL     :  JSR   SPREAD
   00793                             *                    (B)= PORT ADDRESS
   00794                             *  RETURN   :  (A)= READ DATA
   00795                             *              (C)= RETURN STATUS
   00796                             *                      0: NORMAL
   00797                             *                      1: BREAK
   00798                             *
   00799                             *
   00800A 18F5 BD 1C53   A           SPREAD JSR    SRWINT    * SLAVE COMMUNICATION INITIAL
   00801A 18F8 25 12 1C0C                   BCS    SPR900    * ERROR
   00802                             *
   00803A 18FA 86 05     A                  LDA A  #5        * READ COMMMAND
```

```
00804A 1BFC BD FF19   A                JSR    SNSCOM
00805                           *
00806A 1BFF 4F                         CLR A              * PORT ADDRESS (H)
00807A 1C00 BD FF19   A                JSR    SNSCOM
00808                           *
00809A 1C03 17                         TBA                * PORT ADDRESS (L)
00810A 1C04 BD FF19   A                JSR    SNSCOM
00811A 1C07 25 03 1C0C                 BCS    SPR900      * ERROR
00812                           *
00813A 1C09 BD FF16   A                JSR    RV232C      * SLAVE RS232C RECOVERY
00814                           *
00815A 1C0C 39                  SPR900 RTS                * RETURN
00816                           *
00817                           *
00818                           *  FUNCTION :  SLAVE PORT DATA WRITE
00819                           *  CALL     :  JSR   SPWRIT
00820                           *                  (A)= OUTPUT DATA
00821                           *                  (B)= PORT ADDRESS
00822                           *  RETURN   :  (C)= RETURN STATUS
00823                           *                   0: NORMAL
00824                           *                   1: BREAK
00825                           *
00826                           *
00827A 1C0D FD 1761   A         SPWRIT STD    SPWRBF      * DATA SAVE
00828A 1C10 C4 7F     A                AND B  #$7F        * PORT ADDRESS
00829                           *
00830A 1C12 BD 1BF5   A                JSR    SPREAD      * PORT STATUS READ
00831A 1C15 25 3B 1C52                 BCS    SPW900      * ERROR
00832A 1C17 36                         PSH A              * DATA SAVE
00833                           *
00834A 1C18 B6 1762   A                LDA A  SPWRBF+1
00835A 1C1B 2B 11 1C2E                 BMI    SPWR10
00836                           *
00837A 1C1D B6 1761   A                LDA A  SPWRBF      * DATA RESET
00838A 1C20 88 FF     A                EOR A  #$FF
00839A 1C22 B7 1761   A                STA A  SPWRBF      * DATA INVERT
00840                           *
00841A 1C25 32                         PUL A
00842A 1C26 B4 1761   A                AND A  SPWRBF      * OUT DATA
00843A 1C29 B7 1761   A                STA A  SPWRBF
00844A 1C2C 20 07 1C35                 BRA    SPWR20
00845                           *
00846A 1C2E 32                  SPWR10 PUL A              * DATA SET
00847A 1C2F BA 1761   A                ORA A  SPWRBF      * OUT DATA
00848A 1C32 B7 1761   A                STA A  SPWRBF
00849                           *
00850A 1C35 BD 1C53   A         SPWR20 JSR    SRWINT      * SLAVE COMMUNICATION INITIAL
00851A 1C38 25 18 1C52                 BCS    SPW900      * ERROR
00852                           *
00853A 1C3A 86 06     A                LDA A  #6          * WRITE COMMAND
00854A 1C3C BD FF19   A                JSR    SNSCOM
00855                           *
00856A 1C3F 4F                         CLR A              * PORT ADDRESS (H)
00857A 1C40 BD FF19   A                JSR    SNSCOM
00858                           *
```

```
    00859A 1C43 17                         TBA                * PORT ADDRESS (L)
    00860A 1C44 BD FF19   A                JSR     SNSCOM
    00861                          *
    00862A 1C47 B6 1761   A                LDA A   SPWRBF     * DATA OUTPUT
    00863A 1C4A BD FF19   A                JSR     SNSCOM
    00864A 1C4D 25 03 1C52                 BCS     SPW900     * ERROR
    00865                          *
    00866A 1C4F BD FF16   A                JSR     RV232C     * SLAVE RS232C RECOVERY
    00867                          *
    00868A 1C52 39                 SPW900 RTS                 * RETURN
    00869                          *
    00870                          *
    00871                          *  FUNCTION :  SLAVE COMMUNICATION INITIAL
    00872                          *  CALL     :  JSR   SRWINT
    00873                          *  RETURN   :  (C)= RETURN STATUS
    00874                          *                    0: NORMAL
    00875                          *                    1: BREAK
    00876                          *
    00877                          *
    00878A 1C53 86 03     A        SRWINT LDA A   #3         * SLAVE SUPER VISOR MASK OPEN
    00879A 1C55 BD FF19   A               JSR     SNSCOM
    00880                          *
    00881A 1C58 86 AA     A               LDA A   #$AA
    00882A 1C5A BD FF19   A               JSR     SNSCOM
    00883                          *
    00884A 1C5D 39                        RTS                 * RETURN
    00885                          *
    00886                                 TTL     SPACE TABLE
    00887                          *
    00888                          ***************************************************
    00889                          *                                                 *
    00890                          *       SPACE  TABLE                              *
    00891                          *                                                 *
    00892                          ***************************************************
    00893                          *
    00894A 1C5E      0000  A       SPCTBL FDB     0,0        * 0000  ERROR
         A 1C60      0000  A
    00895                          *
    00896A 1C62      1C9E  A              FDB     BARTBL     * 0001  L TO R
    00897A 1C64      1D7E  A              FDB     BARTBL+$E0 * 1000  R TO L
    00898                          *
    00899A 1C66      1CDE  A              FDB     BARTBL+$40 * 0010  L TO R
    00900A 1C68      1D3E  A              FDB     BARTBL+$A0 * 0100  R TO L
    00901                          *
    00902A 1C6A      0000  A              FDB     0,0        * 0011  ERROR
         A 1C6C      0000  A
    00903                          *
    00904A 1C6E      1D1E  A              FDB     BARTBL+$80 * 0100  L TO R
    00905A 1C70      1CFE  A              FDB     BARTBL+$60 * 0010  R TO L
    00906                          *
    00907A 1C72      0000  A              FDB     0,0        * 0101  ERROR
         A 1C74      0000  A
    00908                          *
    00909A 1C76      0000  A              FDB     0,0        * 0110  ERROR
         A 1C78      0000  A
```

```
   00910                            *
   00911A 1C7A    A5     A                  FCB     $A5         * 0111  L TO R  % CODE
   00912A 1C7B    2A     A                  FCB     $2A         *               % DIGIT
   00913A 1C7C    A4     A                  FCB     $A4         *       R TO L  $ CODE
   00914A 1C7D    27     A                  FCB     $27         *               $ DIGIT
   00915                            *
   00916A 1C7E    1D5E   A                  FDB     BARTBL+$C0  * 1000  L TO R
   00917A 1C80    1CBE   A                  FDB     BARTBL+$20  * 0001  R TO L
   00918                            *
   00919A 1C82    0000   A                  FDB     0,0         * 1001  ERROR
        A 1C84    0000   A
   00920                            *
   00921A 1C86    0000   A                  FDB     0,0         * 1010  ERROR
        A 1C88    0000   A
   00922                            *
   00923A 1C8A    AB     A                  FCB     $AB         * 1011  L TO R  + CODE
   00924A 1C8B    29     A                  FCB     $29         *               + DIGIT
   00925A 1C8C    AF     A                  FCB     $AF         *       R TO L  / CODE
   00926A 1C8D    28     A                  FCB     $28         *               / DIGIT
   00927                            *
   00928A 1C8E    0000   A                  FDB     0,0         * 1100  ERROR
        A 1C90    0000   A
   00929                            *
   00930A 1C92    AF     A                  FCB     $AF         * 1110  L TO R  / CODE
   00931A 1C93    28     A                  FCB     $28         *               / DIGIT
   00932A 1C94    AB     A                  FCB     $AB         *       R TO L  + CODE
   00933A 1C95    29     A                  FCB     $29         *               + DIGIT
   00934                            *
   00935A 1C96    A4     A                  FCB     $A4         * 1110  L TO R  $ CODE
   00936A 1C97    27     A                  FCB     $27         *               $ DIGIT
   00937A 1C98    A5     A                  FCB     $A5         *       R TO L  % CODE
   00938A 1C99    2A     A                  FCB     $2A         *               % DIGIT
   00939                            *
   00940A 1C9A    0000   A                  FDB     0,0         * 1111  ERROR
        A 1C9C    0000   A
   00941                            *
   00942                                    TTL     BAR TABLE
   00943                            *
   00944                            ******************************************************
   00945                            *                                                    *
   00946                            *       BAR. TABLE                                   *
   00947                            *                                                    *
   00948                            ******************************************************
   00949                            *
   00950A 1C9E    00     A          BARTBL  FCB     0,0,0       * SPACE=0001  L TO R
        A 1C9F    00     A
        A 1CA0    00     A
   00951A 1CA1    51     A                  FCB     $51         * Q
   00952A 1CA2    00     A                  FCB     0
   00953A 1CA3    4E     A                  FCB     $4E         * N
   00954A 1CA4    54     A                  FCB     $54         * T
   00955A 1CA5    00     A                  FCB     0,0
        A 1CA6    00     A
   00956A 1CA7    4C     A                  FCB     $4C         * L
   00957A 1CA8    53     A                  FCB     $53         * S
```

```
      00958A 1CA9     00     A              FCB     0
      00959A 1CAA     50     A              FCB     $50         * P
      00960A 1CAB     00     A              FCB     0,0,0,0
           A 1CAC     00     A
           A 1CAD     00     A
           A 1CAE     00     A
      00961A 1CAF     4B     A              FCB     $4B         * K
      00962A 1CB0     52     A              FCB     $52         * R
      00963A 1CB1     00     A              FCB     0
      00964A 1CB2     4F     A              FCB     $4F         * O
      00965A 1CB3     00     A              FCB     0,0,0
           A 1CB4     00     A
           A 1CB5     00     A
      00966A 1CB6     4D     A              FCB     $4D         * M
      00967A 1CB7     00     A              FCB     0,0,0,0,0,0,0
           A 1CB8     00     A
           A 1CB9     00     A
           A 1CBA     00     A
           A 1CBB     00     A
           A 1CBC     00     A
           A 1CBD     00     A
      00968                          *
      00969A 1CBE     00     A              FCB     0,0,0       * SPACE=0001  R TO L
           A 1CBF     00     A
           A 1CC0     00     A
      00970A 1CC1     4D     A              FCB     $4D         * M
      00971A 1CC2     00     A              FCB     0
      00972A 1CC3     4F     A              FCB     $4F         * O
      00973A 1CC4     50     A              FCB     $50         * P
      00974A 1CC5     00     A              FCB     0,0
           A 1CC6     00     A
      00975A 1CC7     52     A              FCB     $52         * R
      00976A 1CC8     53     A              FCB     $53         * S
      00977A 1CC9     00     A              FCB     0
      00978A 1CCA     54     A              FCB     $54         * T
      00979A 1CCB     00     A              FCB     0,0,0,0
           A 1CCC     00     A
           A 1CCD     00     A
           A 1CCE     00     A
      00980A 1CCF     4B     A              FCB     $4B         * K
      00981A 1CD0     4C     A              FCB     $4C         * L
      00982A 1CD1     00     A              FCB     0
      00983A 1CD2     4E     A              FCB     $4E         * N
      00984A 1CD3     00     A              FCB     0,0,0
           A 1CD4     00     A
           A 1CD5     00     A
      00985A 1CD6     51     A              FCB     $51         * Q
      00986A 1CD7     00     A              FCB     0,0,0,0,0,0,0
           A 1CD8     00     A
           A 1CD9     00     A
           A 1CDA     00     A
           A 1CDB     00     A
           A 1CDC     00     A
           A 1CDD     00     A
      00987                          *
```

```
ERR   SEQ    LOC  OBJECT       PROGRAM  BARCOD      BAR TABLE

      00988A 1CDE     00     A                FCB     0,0,0       * SPACE= 0010   L TO R
           A 1CDF     00     A
           A 1CE0     00     A
      00989A 1CE1     47     A                FCB     $47         * G
      00990A 1CE2     00     A                FCB     0
      00991A 1CE3     44     A                FCB     $44         * D
      00992A 1CE4     4A     A                FCB     $4A         * J
      00993A 1CE5     00     A                FCB     0,0
           A 1CE6     00     A
      00994A 1CE7     42     A                FCB     $42         * B
      00995A 1CE8     49     A                FCB     $49         * I
      00996A 1CE9     00     A                FCB     0
      00997A 1CEA     46     A                FCB     $46         * F
      00998A 1CEB     00     A                FCB     0,0,0,0
           A 1CEC     00     A
           A 1CED     00     A
           A 1CEE     00     A
      00999A 1CEF     41     A                FCB     $41         * A
      01000A 1CF0     48     A                FCB     $48         * H
      01001A 1CF1     00     A                FCB     0
      01002A 1CF2     45     A                FCB     $45         * E
      01003A 1CF3     00     A                FCB     0,0,0
           A 1CF4     00     A
           A 1CF5     00     A
      01004A 1CF6     43     A                FCB     $43         * C
      01005A 1CF7     00     A                FCB     0,0,0,0,0,0,0
           A 1CF8     00     A
           A 1CF9     00     A
           A 1CFA     00     A
           A 1CFB     00     A
           A 1CFC     00     A
           A 1CFD     00     A
      01006                        *
      01007A 1CFE     00     A                FCB     0,0,0       * SPACE=0010   R TO L
           A 1CFF     00     A
           A 1D00     00     A
      01008A 1D01     43     A                FCB     $43         * C
      01009A 1D02     00     A                FCB     0
      01010A 1D03     45     A                FCB     $45         * E
      01011A 1D04     46     A                FCB     $46         * F
      01012A 1D05     00     A                FCB     0,0
           A 1D06     00     A
      01013A 1D07     48     A                FCB     $48         * H
      01014A 1D08     49     A                FCB     $49         * I
      01015A 1D09     00     A                FCB     0
      01016A 1D0A     4A     A                FCB     $4A         * J
      01017A 1D0B     00     A                FCB     0,0,0,0
           A 1D0C     00     A
           A 1D0D     00     A
           A 1D0E     00     A
      01018A 1D0F     41     A                FCB     $41         * A
      01019A 1D10     42     A                FCB     $42         * B
      01020A 1D11     00     A                FCB     0
      01021A 1D12     44     A                FCB     $44         * D
      01022A 1D13     00     A                FCB     0,0,0
```

```
ERR   SEQ    LOC  OBJECT      PROGRAM  BARCOD      BAR TABLE

          A  1D14    00     A
          A  1D15    00     A
     01023A  1D16    47     A                  FCB     $47        * G
     01024A  1D17    00     A                  FCB     0,0,0,0,0,0,0,0
          A  1D18    00     A
          A  1D19    00     A
          A  1D1A    00     A
          A  1D1B    00     A
          A  1D1C    00     A
          A  1D1D    00     A
     01025                          *
     01026A  1D1E    00     A                  FCB     0,0,0      * SPACE=0100  L TO R
          A  1D1F    00     A
          A  1D20    00     A
     01027A  1D21    37     A                  FCB     $37        * 7
     01028A  1D22    00     A                  FCB     0
     01029A  1D23    34     A                  FCB     $34        * 4
     01030A  1D24    30     A                  FCB     $30        * 0
     01031A  1D25    00     A                  FCB     0,0
          A  1D26    00     A
     01032A  1D27    32     A                  FCB     $32        * 2
     01033A  1D28    39     A                  FCB     $39        * 9
     01034A  1D29    00     A                  FCB     0
     01035A  1D2A    36     A                  FCB     $36        * 6
     01036A  1D2B    00     A                  FCB     0,0,0,0
          A  1D2C    00     A
          A  1D2D    00     A
          A  1D2E    00     A
     01037A  1D2F    31     A                  FCB     $31        * 1
     01038A  1D30    38     A                  FCB     $38        * 8
     01039A  1D31    00     A                  FCB     0
     01040A  1D32    35     A                  FCB     $35        * 5
     01041A  1D33    00     A                  FCB     0,0,0
          A  1D34    00     A
          A  1D35    00     A
     01042A  1D36    33     A                  FCB     $33        * 3
     01043A  1D37    00     A                  FCB     0,0,0,0,0,0,0,0
          A  1D38    00     A
          A  1D39    00     A
          A  1D3A    00     A
          A  1D3B    00     A
          A  1D3C    00     A
          A  1D3D    00     A
     01044                          *
     01045A  1D3E    00     A                  FCB     0,0,0      * SPACE= 0100  R TO L
          A  1D3F    00     A
          A  1D40    00     A
     01046A  1D41    33     A                  FCB     $33        * 3
     01047A  1D42    00     A                  FCB     0
     01048A  1D43    35     A                  FCB     $35        * 5
     01049A  1D44    36     A                  FCB     $36        * 6
     01050A  1D45    00     A                  FCB     0,0
          A  1D46    00     A
     01051A  1D47    38     A                  FCB     $38        * 8
     01052A  1D48    39     A                  FCB     $39        * 9
```

```
ERR   SEQ    LOC  OBJECT       PROGRAM  BARCOD      BAR TABLE

    01053A 1D49    00    A                   FCB    0
    01054A 1D4A    30    A                   FCB    $30        * 0
    01055A 1D4B    00    A                   FCB    0,0,0,0
          A 1D4C   00    A
          A 1D4D   00    A
          A 1D4E   00    A
    01056A 1D4F    31    A                   FCB    $31        * 1
    01057A 1D50    32    A                   FCB    $32        * 2
    01058A 1D51    00    A                   FCB    0
    01059A 1D52    34    A                   FCB    $34        * 4
    01060A 1D53    00    A                   FCB    0,0,0
          A 1D54   00    A
          A 1D55   00    A
    01061A 1D56    37    A                   FCB    $37        * 7
    01062A 1D57    00    A                   FCB    0,0,0,0,0,0,0
          A 1D58   00    A
          A 1D59   00    A
          A 1D5A   00    A
          A 1D5B   00    A
          A 1D5C   00    A
          A 1D5D   00    A
    01063                         *
    01064A 1D5E    00    A                   FCB    0,0,0      * SPACE=1000   L TO R
          A 1D5F   00    A
          A 1D60   00    A
    01065A 1D61    2D    A                   FCB    $2D        * -
    01066A 1D62    00    A                   FCB    0
    01067A 1D63    58    A                   FCB    $58        * X
    01068A 1D64    2A    A                   FCB    $2A        * *
    01069A 1D65    00    A                   FCB    0,0
          A 1D66   00    A
    01070A 1D67    56    A                   FCB    $56        * V
    01071A 1D68    20    A                   FCB    $20        * SP
    01072A 1D69    00    A                   FCB    0
    01073A 1D6A    5A    A                   FCB    $5A        * Z
    01074A 1D6B    00    A                   FCB    0,0,0,0
          A 1D6C   00    A
          A 1D6D   00    A
          A 1D6E   00    A
    01075A 1D6F    55    A                   FCB    $55
    01076A 1D70    2E    A                   FCB    $2E        * .
    01077A 1D71    00    A                   FCB    0
    01078A 1D72    59    A                   FCB    $59        * Y
    01079A 1D73    00    A                   FCB    0,0,0
          A 1D74   00    A
          A 1D75   00    A
    01080A 1D76    57    A                   FCB    $57        * W
    01081A 1D77    00    A                   FCB    0,0,0,0,0,0,0
          A 1D78   00    A
          A 1D79   00    A
          A 1D7A   00    A
          A 1D7B   00    A
          A 1D7C   00    A
          A 1D7D   00    A
    01082                         *
```

```
     01083A 1D7E    00    A              FCB    0,0,0      * SPACE=1000  R TO L
          A 1D7F    00    A
          A 1D80    00    A
     01084A 1D81    57    A              FCB    $57        * W
     01085A 1D82    00    A              FCB    0
     01086A 1D83    59    A              FCB    $59        * Y
     01087A 1D84    5A    A              FCB    $5A        * Z
     01088A 1D85    00    A              FCB    0,0
          A 1D86    00    A
     01089A 1D87    2E    A              FCB    $2E        * .
     01090A 1D88    20    A              FCB    $20        * SP
     01091A 1D89    00    A              FCB    0
     01092A 1D8A    2A    A              FCB    $2A        * *
     01093A 1D8B    00    A              FCB    0,0,0,0
          A 1D8C    00    A
          A 1D8D    00    A
          A 1D8E    00    A
     01094A 1D8F    55    A              FCB    $55        * U
     01095A 1D90    56    A              FCB    $56        * V
     01096A 1D91    00    A              FCB    0
     01097A 1D92    58    A              FCB    $58        * X
     01098A 1D93    00    A              FCB    0,0,0
          A 1D94    00    A
          A 1D95    00    A
     01099A 1D96    2D    A              FCB    $2D        * -
     01100A 1D97    00    A              FCB    0,0,0,0,0,0,0
          A 1D98    00    A
          A 1D99    00    A
          A 1D9A    00    A
          A 1D9B    00    A
          A 1D9C    00    A
          A 1D9D    00    A
     01101                       *
     01102                              TTL    FULL ASCII TABLE
     01103                       *
     01104                       ****************************************************
     01105                       *                                                  *
     01106                       *      FULL ASCII CONVERTION TABLE                 *
     01107                       *                                                  *
     01108                       ****************************************************
     01109                       *
     01110A 1D9E    01    A      FULASC FCB    $01        * $A= SOH
     01111A 1D9F    02    A             FCB    $02        * $B= STX
     01112A 1DA0    03    A             FCB    $03        * $C= ETX
     01113A 1DA1    04    A             FCB    $04        * $D= EOT
     01114A 1DA2    05    A             FCB    $05        * $E= ENQ
     01115A 1DA3    06    A             FCB    $06        * $F= ACK
     01116A 1DA4    07    A             FCB    $07        * $G= BEL
     01117A 1DA5    08    A             FCB    $08        * $H= BS
     01118A 1DA6    09    A             FCB    $09        * $I= HT
     01119A 1DA7    0A    A             FCB    $0A        * $J= LF
     01120A 1DA8    0B    A             FCB    $0B        * $K= VT
     01121A 1DA9    0C    A             FCB    $0C        * $L= FF
     01122A 1DAA    0D    A             FCB    $0D        * $M= CR
     01123A 1DAB    0E    A             FCB    $0E        * $N= SO
```

```
ERR   SEQ    LOC   OBJECT        PROGRAM   BARCOD      FULL ASCII TABLE

      01124A 1DAC    0F    A               FCB     $0F         * $O= SI
      01125A 1DAD    10    A               FCB     $10         * $P= DLE
      01126A 1DAE    11    A               FCB     $11         * $Q= DC1
      01127A 1DAF    12    A               FCB     $12         * $R= DC2
      01128A 1DB0    13    A               FCB     $13         * $S= DC3
      01129A 1DB1    14    A               FCB     $14         * $T= DC4
      01130A 1DB2    15    A               FCB     $15         * $U= NAK
      01131A 1DB3    16    A               FCB     $16         * $V= SYN
      01132A 1DB4    17    A               FCB     $17         * $W= ETB
      01133A 1DB5    18    A               FCB     $18         * $X= CAN
      01134A 1DB6    19    A               FCB     $19         * $Y= EM
      01135A 1DB7    1A    A               FCB     $1A         * $Z= SUB
      01136                     *
      01137A 1DB8    21    A               FCB     $21         * /A= !
      01138A 1DB9    22    A               FCB     $22         * /B=
      01139A 1DBA    23    A               FCB     $23         * /C= #
      01140A 1DBB    24    A               FCB     $24         * /D= $
      01141A 1DBC    25    A               FCB     $25         * /E= %
      01142A 1DBD    26    A               FCB     $26         * /F= &
      01143A 1DBE    27    A               FCB     $27         * /G= '
      01144A 1DBF    28    A               FCB     $28         * /H= (
      01145A 1DC0    29    A               FCB     $29         * /I= )
      01146A 1DC1    2A    A               FCB     $2A         * /J= *
      01147A 1DC2    2B    A               FCB     $2B         * /K= +
      01148A 1DC3    2C    A               FCB     $2C         * /L= ,
      01149A 1DC4    FF    A               FCB     $FF         * /M= -     ERROR
      01150A 1DC5    FF    A               FCB     $FF         * /N= .     ERROR
      01151A 1DC6    2F    A               FCB     $2F         * /O=/
      01152A 1DC7    30    A               FCB     $30         * /P= 0
      01153A 1DC8    31    A               FCB     $31         * /Q= 1
      01154A 1DC9    32    A               FCB     $32         * /R= 2
      01155A 1DCA    33    A               FCB     $33         * /S= 3
      01156A 1DCB    34    A               FCB     $34         * /T= 4
      01157A 1DCC    35    A               FCB     $35         * /U= 5
      01158A 1DCD    36    A               FCB     $36         * /V= 6
      01159A 1DCE    37    A               FCB     $37         * /W= 7
      01160A 1DCF    38    A               FCB     $38         * /X= 8
      01161A 1DD0    39    A               FCB     $39         * /Y= 9
      01162A 1DD1    3A    A               FCB     $3A         * /Z= :
      01163                     *
      01164A 1DD2    61    A               FCB     $61         * +A= SMALL A
      01165A 1DD3    62    A               FCB     $62         * +B= SMALL B
      01166A 1DD4    63    A               FCB     $63         * +C= SMALL C
      01167A 1DD5    64    A               FCB     $64         * +D= SMALL D
      01168A 1DD6    65    A               FCB     $65         * +E= SMALL E
      01169A 1DD7    66    A               FCB     $66         * +F= SMALL F
      01170A 1DD8    67    A               FCB     $67         * +G= SMALL G
      01171A 1DD9    68    A               FCB     $68         * +H= SMALL H
      01172A 1DDA    69    A               FCB     $69         * +I= SMALL I
      01173A 1DDB    6A    A               FCB     $6A         * +J= SMALL J
      01174A 1DDC    6B    A               FCB     $6B         * +K= SMALL K
      01175A 1DDD    6C    A               FCB     $6C         * +L= SMALL L
      01176A 1DDE    6D    A               FCB     $6D         * +M= SMALL M
      01177A 1DDF    6E    A               FCB     $6E         * +N= SMALL N
      01178A 1DE0    6F    A               FCB     $6F         * +O= SMALL O
```

```
ERR   SEQ    LOC  OBJECT      PROGRAM  BARCOD     FULL ASCII TABLE

     01179A 1DE1     70     A              FCB     $70        * +P= SMALL P
     01180A 1DE2     71     A              FCB     $71        * +Q= SMALL Q
     01181A 1DE3     72     A              FCB     $72        * +R= SMALL R
     01182A 1DE4     73     A              FCB     $73        * +S= SMALL S
     01183A 1DE5     74     A              FCB     $74        * +T= SMALL T
     01184A 1DE6     75     A              FCB     $75        * +U= SMALL U
     01185A 1DE7     76     A              FCB     $76        * +V= SMALL V
     01186A 1DE8     77     A              FCB     $77        * +W= SMALL W
     01187A 1DE9     78     A              FCB     $78        * +X= SMALL X
     01188A 1DEA     79     A              FCB     $79        * +Y= SMALL Y
     01189A 1DEB     7A     A              FCB     $7A        * +Z= SMALL Z
     01190                         *
     01191A 1DEC     1B     A              FCB     $1B        * %A= ESC
     01192A 1DED     1C     A              FCB     $1C        * %B= FS
     01193A 1DEE     1D     A              FCB     $1D        * %C= GS
     01194A 1DEF     1E     A              FCB     $1E        * %D= RS
     01195A 1DF0     1F     A              FCB     $1F        * %E= US
     01196A 1DF1     3B     A              FCB     $3B        * %F= ;
     01197A 1DF2     3C     A              FCB     $3C        * %G= <
     01198A 1DF3     3D     A              FCB     $3D        * %H= =
     01199A 1DF4     3E     A              FCB     $3E        * %I= >
     01200A 1DF5     3F     A              FCB     $3F        * %J= ?
     01201A 1DF6     5B     A              FCB     $5B        * %K= LEFT SQUARE BRACKET
     01202A 1DF7     5C     A              FCB     $5C        * %L= REVERSE /
     01203A 1DF8     5D     A              FCB     $5D        * %M= RIGHT SQUARE BRACKET
     01204A 1DF9     5E     A              FCB     $5E        * %N= 0
     01205A 1DFA     5F     A              FCB     $5F        * %O= HAT
     01206A 1DFB     7B     A              FCB     $7B        * %P= LEFT KAGI KAKKO
     01207A 1DFC     7C     A              FCB     $7C        * %Q= VERTICAL DASHU
     01208A 1DFD     7D     A              FCB     $7D        * %R= RIGHT KAGI KAKKO
     01209A 1DFE     7E     A              FCB     $7E        * %S= WAVE
     01210A 1DFF     7F     A              FCB     $7F        * %T= DEL
     01211A 1E00     00     A              FCB     $00        * %U= NUL
     01212A 1E01     40     A              FCB     $40        * %V= @
     01213A 1E02     60     A              FCB     $60        * %W= APOSTROPHY
     01214A 1E03     7F     A              FCB     $7F        * %X= DEL
     01215A 1E04     7F     A              FCB     $7F        * %Y= DEL
     01216A 1E05     7F     A              FCB     $7F        * %Z= DEL
     01217                         *
     01218           0000   A              END
***** TOTAL ERRORS      0
```

CHAPTER 13 MISCELLANEOUS I/O

Page

13.1 Speaker Output 13-1
13.2 Expansion Unit 13-1
13.3 Clock applications 13-2
13.4 Interrupts 13-3
13.5 I/O Initialization and Termination 13-6
13.6 Master MCU Sleep 13-6
13.8 General-purpose Subroutines 13-7
13.9 Subroutine Table 13-8

APPENDIX - Sample Lists 13-13

13.1 Speaker Output

Slave MCU port 15 supplies the output to the speaker. The required square wave frequencies are obtained by dividing this signal and outputting them to the piezoelectric speaker.
To obtain a 1,000Hz output at the piezoelectric speaker, the output at port 15 should be as shown in Fig. 13-1.

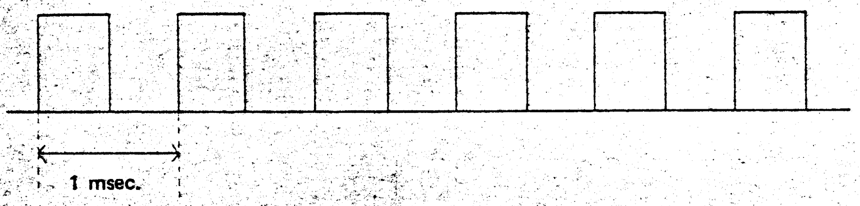


Fig. 13-1 Output to the Piezoelectric Speaker

The SOUND subroutine has been provided to specify the tone and duration of the speaker output.

13.2 Expansion Unit

The expansion unit features a 16K-byte RAM and 16K-byte ROM (only socket is provided). Addresses 0080 to 7FFF can be used as a RAM. A ROM, addresses 8000 to BFFF, may be selected by switching the HX-20 and expansion unit banks.

(1) Memory area

When the expansion unit is connected, addresses 4000 to 7FFF (16K bytes) may be used as a RAM. Data in the RAM is battery-backed up and protected. The ROM (8000-BFFF) is assigned as follows: bank 0 to the HX-20 and bank 1 to the expansion unit. Several memory configurations for the expansion unit are available. For details, refer to the hardware section of this manual.

(2) Switching ROM banks

When the ROM is mounted in the expansion unit, it is selected by switching banks. Banks are switched as follows.

(a) To select the expansion unit ROM (bank 1), access address 0030 (either input or output is fine).

(b) To select the ROM of the HX-20 (bank 0), access address 0032 or 0033 (either input or output is fine).

None of these operations will be possible if the expansion unit is not connected to the HX-20. Also, switching can be performed for ROM area of the expansion unit. The HX-20 ROM (bank 0) is automatically selected when power is turned ON or upon reset.

13.3 Clock applications

The HX-20 clocks may be classified into two types: MCU clocks and IC clocks. The ports and registers related to clocks used in the HX-20 are as follows.

MCU clocks
OCR (Output Compare Register)
(1) Keyboard input sampling
    (Uses OCR interrupt)
(2) RS-232C output timing setting
ICR (Input Capture Register)
(1) Barcode reader timing setting
TOF (Overflow of free running counter)
(1) Built-in microcassette counter sampling
Real-time clock
The real-time clock uses MCU area 4E to 7F as a RAM.

(1) Use of clocks with application software

(a) OCR

An OCF interrupt is generated using OCR when a key on the keyboard is pressed. Sampling (key scanning) is then performed. Therefore, when OCR is used for this purpose, there is a strong chance that input from the keyboard will not be accepted.

A function is also provided whereby, when OCF is set, RS-232C output will be performed by outputting the value of bit 0 of TCSR to P21.

(b) TOF

Counter sampling is executed using the TOF interrupt (at approx. 0.1 sec intervals) during I/O of files by the built-in microcassette.

(c) ICR

This register is used for barcode reader input. ICR measures the interval between pulse edges. However, barcode reader input software is not supported in the basic system of the HX-20.

(d) Real-time clock

The real-time clock is normally employed only to maintain the date and time. It can therefore be used freely in various applications. Sampling may be performed at intervals ranging from 4 to 500msec. Clock registers and RAMs are allocated as shown in Table 13-1.

Table 13-1 Memory Map of Real-time Clock

sec 64

Address	Input/Output	Description
0040	I/O	Seconds
0041	I/O	Alarm (sec.)
0042	I/O	Minutes
0043	I/O	Alarm (min.)
0044	I/O	Hour
0045	I/O	Alarm (hour)
0046	I/O	Day
0047	I/O	Date
0048	I/O	Month
0049	I/O	Year
004A		Control register A
004B		Control register B
004C		Control register C
004D		Control register D
004E~007F		RAM 50 bytes

A 32.768Hz clock pulse is used as the master clock. RAM area 004E to 007F is used as an I/O flag area. Accessing this area can cause an I/O overrun.

13.4 Interrupts

MCU interrupt vectors are assigned as follows in ROM area FFEE~FFFF (Table 13-2).

Table 13-2 Interrupt Vectors

Address	Value	Description
FFEE, FFEF	0106	TRAP
FFF0, FFF1	0109	SCI interrupt

Address	Value	Description
FFF2, FFF3	010C	OCF interrupt
FFF4, FFF5	010F	OCF interrupt
FFF6, FFF7	0112	ICF interrupt
FFF8, FFF9	0115	IRQ1 (Keyboard, power supply switch, clock, voltage down and external interrupts)
FFFA, FFFB	0118	SWI
FFFC, FFFD	011B	NMI
FFFE, FFFF	E000	Reset

Addresses 0106 to 011D are RAM addresses and addresses 0100 to 0105 are used as entry points for interrupts. The initial values for addresses 0110 to 011D are stored in addresses FFB5 to FFCC (Table 13-4). Currently, 5 kinds of IRQ1 interrupts are supported.

Table 13-3: RAM Area Entry Points for Interrupt Processing

Address	Description	Initialize timing
0100~0102	'JMP XXX' command Referenced by IRQ1 interrupt routine (not supported in version 1) when IRQ1 clock interrupt is generated.	Reset (power ON)
0103~0105	'JMP XXX' command Referenced by IRQ1 interrupt routine when external IRQ1 interrupt is generated	Reset (power ON)
0106~0108	'JMP XXX' command (TRAP)	Reset (power ON)
0109~010B	'JNP XXX' command (SCI)	Reset (power ON)
010C~010E	'JNP XXX' command (TOF)	Reset (power ON)
010F~0111	'JNP XXX' command (OCF)	Reset (power ON)
0112~0114	'JNP XXX' command (ICF)	Reset (power ON)
0115~0117	'JNP XXX' command (IRQ1)	Reset (power ON)
0118~011A	Value not set (SW1)	No initialization
011B~011D	Value not set (NMI)	No initialization

Table 13-4 Initial Values for Interrupt Entry Points

Address	Description
FFB5~FFB7	'JMP XXX' command Initial values for addresses 0100 to 0102 (Entry point for clock interrupt)
FFB8~FFBA	'JMP XXX' command Initial values for addresses 0103 to 0105
FFBB~FFBD	'JMP XXX' command Initial values for addresses 0106 to 0108
FFBE~FFC0	'JMP XXX' command Initial values for addresses 0109 to 010B
FFC1~FFC3	'JMP XXX' command Initial values for addresses 010C to 010E
FFC4~FFC6	'JMP XXX' command Initial values for addresses 010F to 0111
FFC7~FFC9	'JMP XXX' command Initial values for addresses 0112 to 0114
FFCA~FFCC	'JMP XXX' command Initial values for addresses 0115 to 0117

Table 13-5 IRQ1 Interrupts

Item	Description	Interrupt confirmation	Interrupt mask
Keyboard	Interrupt is generated while a key is being pressed	P15=0	Set P264 to '0'
Battery Voltage	Interrupt is generated when the battery voltage falls below a specified level	P14=0	None
External interrupt	External bus terminal	P13=0	None
Power switch	Interrupt generated when power switch is turned OFF	P286=0	None
Real-time clock	Real-time clock interrupt is generated	One of the address 004C bits 4, 5, 6 or 7 is set to '1'.	Set address 004B bits 3, 4, 5 and 6 to '0'.

13.5 I/O Initializion and Termination

When the BREAK key is pressed, the interrupt processing routine issues a break command to the slave MCU to terminate the current I/O processing. Then bit 7 of address 007C (variable name SIOSTS) and bit 7 of address 007D (MIOSTS) are turned ON. When the bits 7 of SIOSTS and MIOSTS have been turned ON, the I/O routine assumes that I/O processing has been aborted by BREAK, sets the carry bit to logic '1' and terminates processing.
The following subroutines have been provided to initialize or restart I/O processing.

(1) I/O initialization
Subroutine INITIO initializes I/O operations. Initialization is performed for the keyboard, LCD, microprinter, cassette I/O, ROM cartridge input and RS-232C input. Variables SIOSTS and MIOSTS are cleared. The serial communication driver is not informed. An initialize command is issued to the slave MCU.

(2) I/O restart
Subroutine RSTRIO is used to restart I/O operations. Variables SIOSTS and MIOSTS are cleared. I/O flags for the external cassette, built-in microcassette, ROM cartridge and RS-232C port are also cleared. The microprinter output buffer is also cleared.

(3) Warm start initialization
Subroutine HSTRIO performs warm start initialization. The operation is identical to (1) I/O initialization, above, except that keyboard and LCD initialization are not performed.

(4) Cold start
Subroutine REQINI is provided for cold start processing. The RAM is cleared when the current date and time are entered from the keyboard. The RAM area is checked and the last address of the RAM +1 and stored in addresses 012C, 012D and 0134, 0135. From this point, the processing is the same as that when power is turned ON.

13.6 Master MCU Sleep

The master MCU may be set in the sleep mode to reduce power consumption. The master MCU is reactivated when an interrupt is generated. In the current version, the master MCU enters the sleep mode while awaiting key input. There are restriction on the sleep mode and subroutine SLEEP is called to set the master MCU in the sleep mode.

13.7 Output of Address 26 Port

The value of output port 26 is not actually read. Instead this value is set in address 004F (variable name 'P26') and the value of address 0026 can be obtained by inputting the contents of address 004F. Output of this value is performed by subroutine WRT26.

13.8 General-purpose Subroutines

Entry points have been provided for the following two general-purpose subroutines.

(1) Subroutine HEXBIN converts an ASCII code hexadecimal number into a binary number.
(2) Subroutine BINDEC converts an unsigned 16-bit binary number into an ASCII code decimal number.

13.9 Subroutine Table

Subroutine name	Entry Point	Description
SOUND	FF64	Sounds the speaker Parameters: At Entry (A):Tone:00=pause, 06=440Hz, 0D=880Hz 1,2,3.... 1C:4-octave major scale (from C) 1D, 1E....38:4 octaves a half-tone higher than that for 1,2,3...IC. 39-FF : assumed to be 0 (B):Duration: 1 specifies a duration of 0.1 sec. Duration may be specified in the range 01 to FF. Speaker is not activated when 00 is specified. At Return (C) Abnormal I/O flag Registers retained (A), (B), (X) Subroutines referenced SNSCOM, SNSCOW, CHKRS Variables used None
SLEEP	FFA9	Sets the master MCU in the sleep mode. Control is returned from the SLEEP subroutine when the sleep mode is exited. Parameters: At Entry None At Return None Registers retained (A), (B), (C) Subroutines referenced None Variables used None
CHKPLG	FF2E	Identifies plug-in options currently connected. The value of register (A) is also stored in variable PLGSTS (address 0079). Parameters: At Entry None At Return (C): Abnormal I/O flag (A): Connected device code Bit 2 Bit 1 Bit 0 0 0 0 ROM cartridge 0 0 1 Reserved 0 1 0 Not connected 0 1 1 Reserved 1 x x Microcassette (X: don't care) Registers retained (B), (X) Subroutines referenced CHKRS, SNSCOM Variables used None

PWROFF	FFAC	Turns power supply of the HX-20 OFF. (The power switch is not actually turned OFF.) There is therefore no exit from this subroutine.
		Parameters: At Entry None Subroutines referenced SNSCOM Variables used None
PWRDWN	FF1F	Displays the message "CHARGE BATTERY!" on the LCD. Control is returned from this subroutine when power supply voltage recovers. Otherwise, the power supply is turned OFF after the message has been flashed on the LCD 60 times.
		Parameters: At Entry None At Return None Subroutines referenced DSPLCN, PWROFF Variables used None
REQINI	FF13	Outputs the message "Enter DATE and TIME" at cold start. When the date and time are entered, the extent of the RAM is checked and the memory is cleared. Jumps to the entry point for reset.
		Parameters: At Entry None Subroutines referenced DSPLCN, DSPLCH, KEYIN, HEXBIN
WRTP26	FED4	Port 26 data output. This subroutine is used to output data to port 26. Address 26 data is retained by address 4F.
		Parameters: At Entry (A):Bit positions to be output (for each bit, '1' indicates output and '0', that the bit is not to be output.) (To specify output of bits 0 and 1, set 05 in this register, that is, bits 0 and 1 ON) (B):Output data (Bits not specified in (A) are ignored) Registers retained (A), (B), (X) Subroutines referenced None Variables used ROH (Value is recovered)
BREKIO	FFA3	I/O break. Executes break processing sequence for the slave MCU and turns bits 7 of variables MIOSTS and SIOSTS ON (logic '1').
		Parameters: At Entry None At Return None

Subroutine name	Entry Point	Description
		Registers retained None Subroutines referenced WRTP26, SNSCOM, RSONOF Variables used None
RSTRI0	FFA6	Sets the value of variables to enable restarting of I/O processing after BREAK. Bits 0, 1 and 2 of the following variables are set to '0': MIOSTS, SIOSTS, CSMOD (external cassette status), PRMSTS (ROM cartridge status) and SRSTS. Print buffer is cleared and interrupt is enabled.
		Paramaters: At Entry None At Return None Registers retained (X) Subroutines referenced None Variables used None
CONTIO	FFAF	Clears bits 7 of variables MI0STS and SIOSTS and restarts RS-232C input.
		Parameters: At Entry None At Return None Registers retained None Subroutines referenced CHKRS Variables used None
INITIO	FFCD	Initializes I/O, keyboard and LCD. Sends command 02 to the slave MCU (initialize command). Subroutine RSTRO initialize also performed. Identifies plug-in options and removes interrupt mask. Does not perform initialization for serial communication.
		Paramaters: At Entry None At Return None Registers retained None Subroutines referenced INITKY, INITLC, SNSCOM, HSTRIO, RSTRIO Variables used None

Subroutine name	Entry Point	Description
HSTRIO	FED1	Initializes I/O operation. Does not initialize keyboard and LCD. Parameters: At Entry None At Return None Registers retained None Subroutines referenced SNSCOM Variables used None
HEXBIN	FF2B	Converts an ASCII code hexadecimal number into a binary number. Data is not converted in series but only 1 byte of data can be converted. Parameters: At Entry (A, B):ASCII code 2-digit hexadecimal number At Return (A) Binary number (result of conversion) (B) Return code ØØ: Normal Ø1: Data error ((A, B) not in range Ø to F) (Z) According to the value of (B) Subroutines retained None Variables used Other Reentrant
BINDEC	FF28	Converts unsigned 16-bit binary number into an ASCII code decimal number. Parameters: At Entry (A, B):Unsigned 16-bit binary number (Ø-65535) (X): Address for storing 5-bit result of conversion. Zero are not supressed. At Return None Registers retained (X) Subroutines referenced None Variables used None Øthers Reentrant

Subroutine name	Entry Point	Description
GETCLK	FF31	Inputs the current date and time from the real-time clock (version 2 or better).
		Parameters: At Entry (X): Starting address of the memory area where the input data is to be stored. Data is 6 bytes: Month, Day, Year, Hour, Minutes, Seconds. Each item is in a 2-digit BCD code (one byte). At Return The result is entered in the specified memory address. Registers retained (X) Subroutines referenced None Variables used None
SETCLK	FFF8	Sets the current date in the real-time clock (version 2 or better).
		Parameters: At Entry (X): The starting address of the memory where the specified data is to be stored. The format of the data is the same as for GETCLK. At Return None Registers retained None Subroutines referenced None Variables used None

```
      *** 6301 CROSS MACROASSEMBLER   VER1.0 ***                              10/25/82     11:59:54

ERR   SEQ    LOC  OBJECT         PROGRAM  ALARM       --- ALARM INTERRUPT (BASIC) ---

     00001                                  NAM    ALARM
     00002                                  TTL    --- ALARM INTERRUPT (BASIC) ---
     00003                                  OPT    LOAD
     00004                                  OPT    PAGE=55
     00005                           *
     00006                           * SAMPLE PROGRAM OF 'ALARM INTERPUPT'
     00007                           *  DISPLAY CURRENT TIME.
     00008                           *  THE MOLODY IS PLAYED WHEN MINUTES IS UPDATED (SECOND = 00).  BECAUSE
     00009                           * ALARM INTERRUPT IS CAUSED AND MELODY COMMANDS ARE SENDED TO SLAVE MCU
     00010                           * IN INTERRUPT ROUTINE.
     00011                           *
     00012                           *  FILE NAME   'EX$C'   BY  K.A
     00013                           * BASIC PROGRAM
     00014                           * 10 CLS
     00015                           * 20 FOR I=&HB00 TO &HB06
     00016                           * 30  READ J
     00017                           * 40  POKE I,J
     00018                           * 50 NEXT I
     00019                           * 60 FOR I=&HB10 TO &HB45
     00020                           * 70  READ J
     00021                           * 80  POKE I,J
     00022                           * 90 NEXT I
     00023                           * 100 EXEC &HB00
     00024                           * 105 'WRITE INTERRUPT VECTOR
     00025                           * 110 POKE &H116,&H0B
     00026                           * 120 POKE &H117,&H10
     00027                           * 130 POKE &H7E,&H80
     00028                           * 135 'ENABLE ALARM INTERRUPT
     00029                           * 140 POKE &H4B,&H22
     00030                           * 150 POKE &H41,&H00
     00031                           * 160 POKE &H43,&HFF
     00032                           * 170 POKE &H45,&HFF
     00033                           * 180 LOCATE 5,2
     00034                           * 190 PRINT TIMES
     00035                           * 200 GOTO 180
     00036                           * 1000 DATA &HFC,&HFF,&HCB,&HFD,&H0B,&H07,&H39
     00037                           * 1010 DATA &H96,&H4C,&H23,&H05,&HFE,&H0B,&H07,&H6E,&H00
     00038                           * 1020 DATA &HCE,&H0B,&H33,&H86,&H34,&HBD,&HFF,&H19,&HA6,&H00,&H3.
     00039                           * 1030 DATA &HBD,&HFF,&H19,&H32,&H08,&H81,&HFF,&H26,&HF4
     00040                           * 1040 DATA &H86,&H35,&HBD,&HFF,&H19,&H3B
     00041                           * 1050 DATA 17,06,44,06,17,06,44,06,17,06,14,06,16,06,15,06,13,18,&HFF
     00042                           *
     00043                           *
     00044                           *
     00045A 0B00                            ORG    $B00
     00046                           *
     00047                           * STORE INTERRUPT VECTOR
     00048          FF19   A         SNSCOM EQU    $FF19
     00049          FFCA   A         INTIR1 EQU    $FFCA     * IRQ1 INTERRUPT INITIAL ADDRESS
     00050                           *
     00051A 0B00 FC FFCB   A                LDD    INTIR1+1
     00052A 0B03 FD 0B07   A                STD    SAVADD
     00053A 0B06 39                         RTS
     00054                           *
     00055A 0B07    0002   A         SAVADD RMB    2
     00056                              * IRQ1 INTERRUPT ROUTINE
     00057A 0B10                               ORG    $B10
     00058A 0B10 96 4C     A                   LDA A  $4C       * IS INTERRUPT CAUSED BY CLOCK ?
     00059A 0B12 2B 05 0B19                    BMI    CLKINT
     00060A 0B14 FE 0B07   A                   LDX    SAVADD
     00061A 0B17 6E 00     A                   JMP    0,X
     00062                              *
     00063                              * SEND SLAVE MCU DATA OF MELODY.
     00064A 0B19 CE 0B33   A            CLKINT LDX    #MELTBL   * (X):THE ADDRESS WHERE MELODY DATA ARE STORED.
     00065A 0B1C 86 34     A                   LDA A  #$34      * SEND DATA TO SLAVE MCU.
     00066A 0B1E BD FF19   A                   JSR    SNSCOM    * COMMAND 34: SET MELODY DATA
     00067A 0B21 A6 00     A            SLV10  LDA A  0,X       * SET DATA
     00068A 0B23 36                            PSH A
     00069A 0B24 BD FF19   A                   JSR    SNSCOM
     00070A 0B27 32                            PUL A
     00071A 0B28 08                            INX
     00072A 0B29 81 FF     A                   CMP A  #$FF      * LAST CHARACTER IS $FF
     00073A 0B2B 26 F4 0B21                    BNE    SLV10
     00074                              *
     00075                              * PLAY MELODY.
     00076A 0B2D 86 35     A                   LDA A  #$35
     00077A 0B2F BD FF19   A                   JSR    SNSCOM
     00078A 0B32 3B                            RTI
     00079                              *
     00080                              * MELODY TABLE (FOR ELISE)
     00081A 0B33     11    A            MELTBL FCB    17,06,44,06,17,06,44,06
          A 0B34     06    A
          A 0B35     2C    A
          A 0B36     06    A
          A 0B37     11    A
          A 0B38     06    A
          A 0B39     2C    A
          A 0B3A     06    A
     00082A 0B3B     11    A                   FCB    17,06,14,06,16,06,15,06
          A 0B3C     06    A
          A 0B3D     0E    A
          A 0B3E     06    A
          A 0B3F     10    A
          A 0B40     06    A
          A 0B41     0F    A
          A 0B42     06    A
     00083A 0B43     0D    A                   FCB    13,18
          A 0B44     12    A
     00084A 0B45     FF    A                   FCB    $FF
     00085                              *
     00086           0000  A                   END
***** TOTAL ERRORS      0
```

CHAPTER 14 MEMORY MAP

Page

14.1 Memory Allocation 14-1
14.2 Jump Table 14-2
14.3 ROM Vectors 14-6
14.4 RAM page Ø vectors 14-7
14.5 RAM system variables 14-12
14.6 RAM area used by I/O routines 14-14

14.1 Memory Allocation

The memory of HX-20 is divided into the following areas.

Table 14-1 Memory Map

Address	Without expansion unit	With expansion unit	Applications
0000 to 004D	I/O ports	←	This area is used by I/O routines as work and flag area.
004E to 007F	RAM (Real-time clock)	←	
0080 to 00FF	RAM	←	This area is used as a work area by the BASIC interpreter.
0100 to 04AF	RAM	←	This area is used by I/O routines as work area and and I/O buffer.
04B0 to 0A3F	RAM	←	This area is used as a work area by the BASIC interpreter.
0A40 to 3FFF	RAM	←	
4000 to 5FFF	None	RAM (in expansion unit)	
6000 to 7FFF	ROM (ROM5) (Only socket provided.)	RAM (in expansion unit)	
8000 to 9FFF	ROM (ROM4)	ROM (ROM2) Can be switched to ROM in expansion unit.	ROM in the HX-20 is the BASIC interpreter.
A000 to BFFF	ROM (ROM3)	ROM (ROM1). Can be switched to ROM in expansion unit.	ROM in the HX-20 is the BASIC interpreter.
C000 to DFFF	ROM (ROM2)	←	C000 to CFFF is memory area for the BASIC interpreter. D000 to DFFF contains Menu, Monitor and virtual screen routines.

Address	Without expansion unit	With expansion unit	Applications
E000 to FFFF	ROM (ROM1)	←	This area is used by I/O routines.

14.2 Jump Table

Jump tables show the entry points of various subroutines. Entry points are indicated by a 3-byte address specification. Initial byte specifies 7E (JPM command) followed by high and low bytes of the address.

Table 14-2 Jump Table

Address (from)(to)	Contents	Remarks	For details, refer to this manual, Chapter:
FED1 FED3	JMP HSTRIO	I/O restart, Initialize	13
FED4 FED6	JMP WRTP26	Address 26 port output	13
FED7 FED9	JMP BILOAD	Memory Load: Load, Close after end of processing	9
FEDA FEDC	JMP OPNLOD	Memory Load: Load Open	9
FEOD FEDF	JMP BIDUMP	Memory Dump: Dump and Close after end of processing	9
FEEO FEE2	JMP OPNDMP	Memory Dump: Dump Open	9
FEE3 FEE5	JMP DIRPRM	Read PROM cartridge directory	8
FEE6 FEE8	JMP CLSPRM	Closes PROM cartridge file.	8
FEE9 FEEB	JMP REDPRM	Reads 1 character from PROM cartridge file.	8
FEEC FEEE	JMP OPNPRM	Opens PROM cartridge file.	8
FEEF FEF1	JMP CNTMCS	Read/Write to built-in microcassette counter value.	6
FEF2 FEF4	JMP SECMCS	Advances tape to the specified built-in microcasette counter value.	6
FEF5 FEF7	JMP REWMCS	Rewinds built-in microcasette.	6
FEF8 FEFA	JMP SETCLK	Inputs time and date. (Version 2 or better)	6
FEFB FEFD	JMP CLSMCS	Closes built-in microcassette files.	6
FEFE FEF0	JMP WRTMCS	Outputs one character to built-in microcassette.	6
FF01 FF03	JMP OPNWMS	Opens built-in microcassette file for output.	6
FF04 FF06	JMP READMS	Inputs one character from built-in microcassette.	6
FF07 FF09	JMP SRCRMS	Opens built-in microcassette file for input (initializes file).	6
FF0A FF0C	JMP OPNRMS	Opens built-in microcassette file for input (searches specified file).	6
FF0D FF0F	JMP MCSMAN	Sets built-in microcassette in manual operation mode.	6
FF10 FF12	JMP $DFF7	Jumps to address DFF7.	
FF13 FF15	JMP REQINI	Initializes HX-20 cold start.	13
FF16 FF18	JMP CHKRS	RS-232C recovery after aborting input processing	5
FF19 FF1B	JMP SNSCOM	Sends one command byte to slave CPU.	11
FF1C FF1E	JMP SRINIT	Initializes high-speed serial communication.	4
FF1F FF21	JMP PWRDWN	Battery Low message	13
FF22 FF24	JMP KYSSTK	Stores data in the initial key stack.	2
FF25 FF27	JMP $DFFD	Jumps to address DFFD (MENU).	
FF28 FF2A	JMP BINDEC	Converts binary numbers into ACSII decimal code.	13
FF2B FF2D	JMP HEXBIN	Converts ASCII hexadecimal code into binary code.	13
FF2E FF30	JMP CHKPLG	Identification of plug-in options	13
FF31 FF33	JMP GETCLK	Sets time and date (Version 2 or better)	13
FF34 FF36	JMP CLSCS	Closes external cassette file.	6
FF37 FF39	JMP WRITCS	Outputs one byte to external cassette file.	6

Address		Contents	Remarks	For details, refer to this manual, Chapter:
(from)	(to)			
FF3A	FF3C	JMP OPNWCS	Opens external cassette file for output.	6
FF3D	FF3F	JMP READCS	Inputs 1 byte from external cassette file.	6
FF40	FF42	JMP SRCRCS	Opens external cassette file for input (initializes file).	6
FF43	FF45	JMP OPNRCS	Opens external cassette file for input.	6
FF46	FF48	JMP OPNFCS	External cassette file remote (ON/OFF)	6
FF49	FF4B	JMP DSPLCN	Displays n characters on LCD (Physical screen).	3
FF4C	FF4E	JMP DSPLCH	Displays one character on LCD (Physical screen).	3
FF4F	FF51	JMP $DFFI	Displays one character on virtual screen.	15
FF52	FF54	JMP LCADDR	Link table for LCD routines. Selects LCD driver.	
FF55	FF57	JMP LCDMOO	Link table for LCD routines. Selects LCD driver mode.	
FF58	FF5A	JMP DATMOD	Link table for LCD routines. Outputs data to LCD driver.	
FF5B	FF5D	JMP DISPIT	Displays one character on LCD. (Data is not entered in physical screen buffer.)	3
FF5E	FF60	JMP $DFF4	Calls virtual screen function.	4
FF61	FF63	JMP $DFEE	Displays (recovers) current virtual screen data.	
FF64	FF66	JMP SOUND	Speaker output	13
FF67	FF69	JMP CHRGEN	Generates character font.	3
FF6A	FF6C	JMP KEYSCN	Scans key matrix.	2
FF6D	FF6F	JMP SERIN	High-speed serial data input	4
FF70	FF72	JMP SEROUT	High-speed serial data output	4
FF73	FF75	JMP SERONF	High-speed driver ON/OFF	4
FF76	FF78	JMP RSPUT	Outputs one character to RS-232C.	5
FF79	FF7B	JMP RSGET	Inputs one character to RS-232C.	5
FF7C	FF7E	JMP RSGSTS	Inputs RS-232C status register value.	5
FF7F	FF81	JMP RSCLOS	Closes RS-232C input.	5
FF83	FF84	JMP RSOPEN	Opens RS-232C output.	5
FF85	FF87	JMP RSONOF	Controls RS-232C driver (ON/OFF)	5
FF88	FF8A	JMP RSMST	Sets RS-232C status register mode.	5
FF8B	FF8D	JMP SCRCPY	Screen copy (LCD to micro-printer).	7
FF8E	FF90	JMP NFEED	Performs n dot-lines of line feed on microprinter.	
FF91	FF93	JMP PRTDOT	Prints one dot-line (bit pattern) on the microprinter.	7
FF94	FF96	JMP LNPRNT	Prints one character-line on the microprinter.	7
FF97	FF99	JMP CHPRNT	Prints one character on the microprinter.	7
FF9A	FF9C	JMP KEYIN	Enters one character from keyboard.	2
FF9D	FF9F	JMP KEYSTS	Enters keyboard key status.	2
FFA0	FFA2	JMP INITKY	Initializes keyboard.	2
FFA3	FFA5	JMP BREKIO	I/O break	13
FFA6	FFA8	JMP RSTRIO	Restart after I/O break	13
FFA9	FFAB	JMP SLEEP	Master MCU sleep	13

Address (from)(to)	Contents	Remarks	For details, refer to this manual, Chapter:
FFAC FFAE	JMP PWROFF	Power supply OFF	13
FFAF FFB1	JMP CONTIO	Continuation after I/O break	13
FFB2 FFB4	JMP BRKIN	Entry point after break break key has been pressed.)	
FFB5 FFB7	JMP CLKINT	Initial value for clock interrupt entry point	
FFB8 FFBA	JMP IRQI80	Initial value for IRQ1 external interrupt entry point	
FFBB FFBD	JMP SDFFA	Initial value for TRAP interrupt entry point	
FFBE FFC0	JMP SERINT	Initial value for SCI interrupt entry point	
FFC1 FFC3	JMP TOFINT	Initial value for TOF interrupt entry point	
FFC4 FFC6	JMP OCFINT	Initial value for OCF interrupt entry point	
FFC7 FFC9	JMP IRQI80	Initial value for ICF interrupt entry point	
FFCA FFCC	JMP IRQINT	Initial value for IRQ1 interrupt entry point	
FFCD FFCF	JMP INITIO	I/O initialize	

(1) ROM (ROM2) jump tables (Addresses C000 to DFFF)

Address	Contents	Notes
DFEE to DFF0	JMP LCRECV	Covers the virtual screen and rewrites only the physical screen.
DFF1 to DFF3	JMP SCRCHR	Displays one character on the virtual screen.
DFF4 to DFF6	JMP SCRFNC	Screen functions of the virtual screen.
DFF7 to DFF9	JMP MON	Monitor entry
DFFA to DFFC	JMP MONTRP	Monitor entry on TRAP
DFFD to DFFF	JMP MENU	Menu entry

14.3 ROM Vectors

Address (from) (to)	Variable name	Number of bytes	Description
FFD0 FFD1	NEWKTB	2	Shows the address at which the matrix data is stored after key scanning (NEWKTB).
FFD2 FFD3	COLCNT	2	Shows the address where the amount of data in the built-in microprinter buffer is stored (COLCNT).
FFD4 FFD5	CSBFCM	2	Shows the address where the amount of data in the external cassette buffer is stored (CSBFCM). Used for data read and write.
FFD6 FFD7	MSBFCM	2	Shows the address where the amount of data in built-in microcassette buffer is stored (MSBFCM). Used for data read and write.
FFD8 FFD9	RSDCNT	2	Shows the address where the amount of data in the RS-232C input buffer is stored (RSDCNT).
FFDA FFDB		2	Shows the starting address of the LCD physical screen buffer.
FFDC FFDD	CASBUF	2	Shows the starting address of the 260-byte buffer used by the monitor (CASBUF).
FFDE FFEF		2	Shows the address where the scroll speed data is stored.
FFE0 FFE1	CSHBUF	2	Shows the starting address of the external cassette header buffer (CSHBUF).
FFE2 FFE3	MSHBUF	2	Shows the starting address of the built-in microcassette header buffer (MSHBUF).
FFE4 FFE5	KEYMOD	2	Shows the address where the key input mode data is stored (KEYMOD).

NOTE: Addresses are shown as two bytes in upper- and lower-byte sequence.

FFEE FFEF		2	Shows the address where the TRAP entry point is stored. Set to 0106.
FFF0 FFF1		2	Shows the address where the SCI interrupt entry point is stored. Set to 0109.
FFF2 FFF3		2	Shows the address where the TOF interrupt entry point is stored. Set to 010C.
FFF4 FFF5		2	Shows the address where the OCF interrupt entry point is stored. Set to 010F.
FFF6 FFF7		2	Shows the address where the ICF interrupt entry point is stored. Set to 0112.

Address (from)	(to)	Variable name	Number of bytes	Description
FFF8	FFF9		2	Shows the address where the IRQ1 interrupt entry point is stored. Set to 0115.
FFFA	FFFB		2	Shows the address where the SWI interrupt entry point is stored. Set to 0118.
FFFC	FFFD		2	Shows the address where the NMI interrupt entry point is stored. Set to 011B.
FFFE	FFFF		2	Shows the address where the RESET interrupt entry point is stored. Set to E000.

14.4 RAM page 0 vectors

Address (from)	(to)	Variable name	Number of bytes	Description
4E	4E	PWRFLG	1	Bits 0 to 3: Reserved for selecting processing to be executed when power supply is turned ON. Bits 4 to 7: Indicates the processing to be executed when power supply is turned OFF. Bit 7 Bit 6 Bit 5 Bit 4 0 0 0 0 No operation 0 0 0 1 Executes the 0 0 1 0 subroutine specified in addresses 132~133 (POFADR) prior to turning OFF the power supply. All other bit values No operation.
4F	4F	P26	1	Address 26 port data. Note: Read of address 26 is inhibited.
50	51	R0	2	This area is used as a work area by I/O routine.
52	53	R1	2	Same as R0
54	55	R2	2	Same as R0
56	57	R4	2	Same as R0
58	59	R3	2	Same as R0
5A	5B	R5	2	Same as R0
5C	5D	R6	2	Same as R0
5E	5F	R7	2	Same as R0
60	61	M0	2	This area is used as a work area by Monitor and screen routines.

Address (from)	 (to)	Variable name	Number of bytes	Description
62	63	M1	2	Same as M0
64	65	M2	2	Same as M0
66	67	M3	2	Same as M0
68	69	M4	2	Same as M0
6A	6B	M5	2	Same as M0
6C	6D	M6	2	Same as M0
6E	6F	M7	2	Same as M0
70	71	K0	2	The area is used as a work area by the key input routine.
72	73	K1	2	Same as K0
74	75	S0	2	Same as K0
76	77	S1	2	Same as K0
78	78	INIFL1	1	Indicates application program cold start. For each bit, '0' indicates cold start and '1', warm start Bit 1: Bit 2: Bit 3: Bit 4: Bit 5: Bit 6: BASIC application programs Bit 7: BASIC interpreter
79	79	PLGSTS	1	Bits 0 to 2: Indicate the plug-in option. Bit 2 Bit 1 Bit 0 0 0 0 ROM cassette 0 0 1 Reserved 0 1 0 Not connected 0 1 1 Reserved 1 x x Microcassette (x: don't care) Bit 3: 0 Bits 4 to 6: not used Bit 7: Specifies whether RS-232C driver will be turned OFF when the BREAK key is pressed. 0: Not turned OFF 1: Turned OFF.

Address (from)	 (to)	Variable name	Number of bytes	Description
7A	7A	SRSTS	1	Bits 0 to 2: Indicate current RS-232C status. Bit 2 Bit 1 Bit 0 0 0 0 Input operation is not being performed. 0 0 1 Input operation is being executed. 0 1 x Not used in current version 1 0 0 Undefined. 1 0 1 Input. Operation enters wait state when the slave MCU is busy with other I/O devices such as microprinter. 1 1 x Undefined Bit 3: Indicates RS-232C driver status (ON/OFF) 0: OFF 1: ON Bit 4: Serial I/F driver status The same driver is used as the RS-232C and Serial I/F driver. However, in terms of operation by software, they are treated independently. Bits 5 to 7: SCI (Serial Communication Interface) interrupt mode Bit 7 Bit 6 Bit 5 0 0 0 Input of external cassette data 0 0 1 Input of internal microcassette data 0 1 0 RS-232C data input 0 1 1 Serial I/F data input 1 0 0 Output of external cassette data 1 0 1 Output of built-in microcassette data 1 1 x Undefined
7B	7B	RUNMOD	1	Program execution mode Bits 0 to 3: Reserved for specifying program number, etc. Bits 4 to 5: Undefined Bit 6: Flag indicating whether the virtual screen is being used 0: Virtual screen being used 1: Virtual not being used Bit 7: Indicates the interpreter mode 0: Machine language mode 1: Interpreter mode

Address (from)	 (to)	Variable name	Number of bytes	Description
				NOTE: In machine language mode, the program jumps to the specified address when the BREAK key is poressed, power is turned OFF or the voltage falls. In interpreter mode when one of these interrupts is generated, the appropriate flag is set (MIOSIS) and control is returned. In BASIC the values of bit 7 and bit 6 are 1, Ø and in Monitor Ø, 1.
7C	7D	SIOSTS	1	Flags to indicate the current I/O status of the slave MCU I/O Bit Ø: Microprinter control (1: being executed) Bit 1: External cassette Read/Write (1: being executed) Bit 2: Internal microcassette Read/Write or control (1: being executed) Bit 3: RS-232C Receive (1: being executed) Bit 5: ROM cartridge power supply (1: ON) Bit 6: Bar-code reader power supply (1: ON) Bit 7: BREAK (1: Slave MCU I/O control forcibly terminated by master MCU).
7D	7D	MIOSTS	1	Indicates the I/O status of the master MCU. Bit Ø: Read/Write to LCD (1: being executed) Bit 1: Commant transmit and response with slave MCU (1: being executed) Bit 2: Data communication using the external serial port (Floppy disk unit), (1: being executed) Bit 3: Clock interrupt (alarm, square wave, update), (1: Interrupt) Bit 4: Voltage low (1: Voltage low interrupt) Bit 5: Power OFF (1: Power switch interrupt) Bit 6: PAUSE key ON (1: PAUSE key pressed) Bit 7: BREAK key ON (1: BREAK key pressed)
7E	7E	SDIPS1	1	Software switch Bit Ø, Bit 1: Specify the type of waveform from the external cassette. Bit Ø Bit 1 Ø x Decided automatically 1 Ø Normal waveform 1 1 Reverse waveform

Address		Variable	Number	Description
(from)	(to)	name	of bytes	
				Bit 2, Bit 3 Specify the type of waveform from internal microcassette Bit 3 Bit 2 0 x Decided automatically 1 0 Normal waveform 1 1 Reverse waveform Bit 4, Bit 5 Memory bank selection Bit 6: Indicates the memory bank in which the BASIC in terpreter is located. (Value is set when the menu is initialized.) 0: Bank 0 1: Bank 1 Bit 7: Specifies access of addresses 0000 to 004D 0: Access disabled 1: Access enabled
7E	7F	SDIPS2	1	Software switch Bits 0 to 3: Correspond to DIP switches 1 to 4 0: OFF 1: ON Bit 4: Flag indicating whether DIP switches 1 to 4 will be controlled by software (bits 0 to 3 above) or by the actual setting. 0: Actual DIP switch setting 1: Bits 0 to 3 Bit 5: Flag indicating whether bit 7 will control the printer ON/OFF switch. 0: Actual printer ON/OFF switch setting 1: Bit 7 Bit 6: Undefined Bit 7: Controls the printer ON/OFF switch 0: OFF 1: ON Note: These switches are included in the key matrix. The values of these switches are therefore set in the key matrix (NEWKTB) after key scanning.

14.5 RAM system variables

Address (from) (to)	Variable name	Number of bytes	Description
0100 0102	INTCLK	3	Address of real-time clock interrupt routine (for alarm, etc.) Address 0100 contains 7E (JMP command) and 0101, 0102 the upper and lower bytes of the jump address. Address values are initialized on reset.
0103 0105	INTEXT	3	Address of IRQ1 external port interrupt routine. Contents are identical to INTCLK.
0106 0108		3	Address of TRAP interrupt routine. Contents are identical to INTCLK.
0109 010B		3	Address of the IRQ1 SCI interrupt routine. Contents are identical to INTCLK.
010C 010E	INTOF	3	Address of the IRQ1 TOF interrupt routine. Contents are identical to INTCLK.
010F 0111		3	Address of the IRQ1 OCF routine. Contents are identical to INTCLK.
0112 0114		3	Address of the IRQ1 ICF routine. Contents are identical to INTCLK.
0115 0117		3	Address of the IRQ1 interrupt routine. Contents are identical to INTCLK.
0118 011A	INTSW1	3	Address of the SW1 routine. Three bytes are reserved.
011B 011D		3	Address of the NMI interrupt routine. Three bytes are reserved.
011E 011F	FNTGPN	2	Address of the character fonts for codes E0-FF (Upper- and lower-byte sequence).
0120 0121	BRKADR	2	Address of the subroutine to be executed when the BREAK key is pressed. This specification is valid only when RUNMOD is in machine language mode.
0122 0123	MENADR	2	Address of the subroutine to be executed when the MENU key is pressed. Contents are identical to BRKADR.
0124 0125	PAUADR	2	Address of the subroutine to be executed when the PAUSE key is pressed. Contents are identical to BRKADR.

Address (from) (to)	Variable name	Number of bytes	Description
0126 0127	CT3ADR	2	Address of the subroutine to be executed when CTRL/PS3 key is pressed. Control jumps unconditionally to this address. Address value is initialized at reset.
0128 0129		2	Address of the subroutine to be executed when CTRL/PF4 key is pressed. Contents are identical to CT3ADR.
012A 012B		2	Address of the subroutine to be executed when CTRL/PF5 key is pressed. Contents are identical to CT3ADR.
012C 012D	RMBADR	2	Shows the end of the RAM area. This variable is set when the RAM is checked at initialization (CTRL/@ input from MENU). Last address of the RAM +1 is stored in upper- and lower-byte sequence.
012E 012F	PRMCNT	2	Address where the amount of data remaining in the PROM cartridge file data is stored.
0130 0131	WAKADR	2	Address of the subroutine executed by the clock alarm interrupt at reset (Power ON). Address is in upper- and lower-byte sequence. This address is initialized at reset.
0132 0133	POFADR	2	Address of the last subroutine called prior to turning OFF the power supply. Address is in upper- and lower- byte sequence. This address is initialized at reset.
0134 0135	BSWTAD	2	Starting address of the BASIC application area. Value of RMBADR is set at MENU initialization (CTRL/@). Set to same value as RMBADR.
0136 0137	BSWBAD	2	Starting address of BASIC program area.
0138 0139		2	Address of the BASIC work area save and condense routine.
013A 013A	BITMP0	1	Bank 0 bit map.
013B 013B	BITMP1	1	Bank 1 bit map.
013C 013F	LNKTBL	4	Address of the RAM application program link table.

14.6 RAM area used by I/O routines

Memory range	Description
ØØ4E to ØØ7F	Flag and work area
Ø1ØØ to Ø11Ø	Interrupt entry pointer
Ø11E to Ø139	Vector
Ø13A to Ø13F	Menu and Link tables
Ø14Ø to Ø18F	Keyboard work area
Ø19Ø to Ø1AE	Microprinter work area
Ø1AF to Ø1C3	RS-232C work area
Ø1C4 to Ø1D5	Serial communication work area
Ø1D6 to Ø1EB	External cassette work area
Ø1EC to Ø2Ø7	Built-in microcassette work area
Ø2Ø8 to Ø2ØE	ROM cartridge work area

Memory range	Description
020F to 021A	Binary memory dump, memory load work area.
021B to 021F	Reserved
0220 to 029F	Screen (including LCD routine) routine work area.
02A0 to 02CF	Monitor work area
02D0 to 0323	External cassette header work area.
0324 to 0377	Built-in microcassette header work area.
0380 to 047C	Reserved for system buffer (260 bytes)

CHAPTER 15 VIRTUAL SCREEN

Page

15.1 General 15-1
15.2 Names and Technical Terms 15-1
15.3 Graphic Display 15-4
15.4 Virtual Screen Control 15-5
15.5 Virtual Screen Function Table 15-9
15.6 EPSP Message Format Table for Screen 15-16

15.1 General

Note: This chapter contains descriptions related to the display controller for external display. This hardware is not available in every country. In countries where the display controller is not available, this chapter should be ignoring all references to the display controller and the external display.

The virtual screen is intended to allow the HX-20 to use a larger screen than the physical screen size of its LCD (20 columns by 4 lines). This function is good for both the LCD and the display controller (for external display).
The virtual screen has a maximum size of 255 columns by 255 lines. The display area where characters actually appear is called a "window". (The size of this window become 32 columns by 16 lines with the display controller.) It functions as a viewing window through which any part of the large internal screen can be seen. The virtual screen on the LCD is controlled by the master MCU, whereas that on the external display is controlled by the display controller via a high-speed serial communication interface.

15.2 Names and Technical Terms

(1) Virtual screen and physical screen
Only character (or text) information is handled by the virtual screen. Its maximum size is 255 columns by 255 lines. For the LCD, a screen image is produced on the MCU memory. As opposed to the virtual screen, the screen used for actual display is called a "physical screen".
The size of the physical screen is 20 columns by 4 lines for the LCD display and 32 columns by 16 lines for the display controller. Graphic display (straight line, etc.) is applicable to the physical screen only.

(2) Window
The window is a portion of the virtual screen that is actually displayed for viewing. The contents of the window are the same as those of the physical screen.

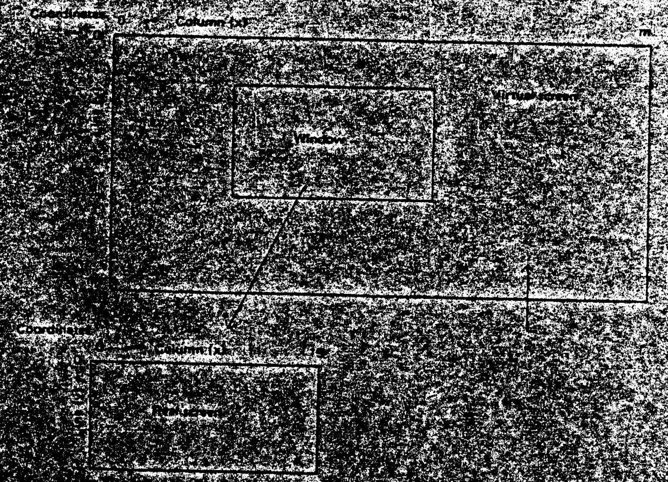


Fig. 15-1 Virtual Screen, Physical Screen and Window

(3) Coordinates on the screen
The screen in a horizontal direction is called "columns", while that in a vertical direction is called "lines". Coordinates are represented by x and y, which correspond to columns and lines, respectively. Column Ø indicates the left end of the screen, while line Ø indicates the top end of the screen. When the screen size is (m,n), the upper left end of the screen is identified by coordinates (Ø,Ø) and the lower right end by coordinates (m-1,n-1).

(4) Scroll
The scroll refers to the movement of the contents of the window up by one line. (Namely, the contents of the 4th line appear in the 3rd line, the contents of the 3rd line in the 2nd line, and the contents of the 2nd line in the 1st line. New data appears in the 4th line. In the HX-2Ø, this function is also applicable to the movement of the screen in the upward, left and right directions.

(5) Scroll step
A character code to specify the number of scroll steps. When this code is accepted, the screen scrolls by the number of columns or lines specified by this code.

(6) Scroll of virtual screen
The scroll of virtual screen refers to the movement of the contents of the virtual screen up or down by one line.

(7) Line status
In some cases, two lines of data to be displayed are desired to be handled as a single line. To support this, a flaq is provided to indicate a continuation line for each line. This flag is called a "line status flag" (see Fig. 9-2). The line status has a value "FF" if the line is a continuation of the preceding line and a value "ØØ" if the line is a new line.

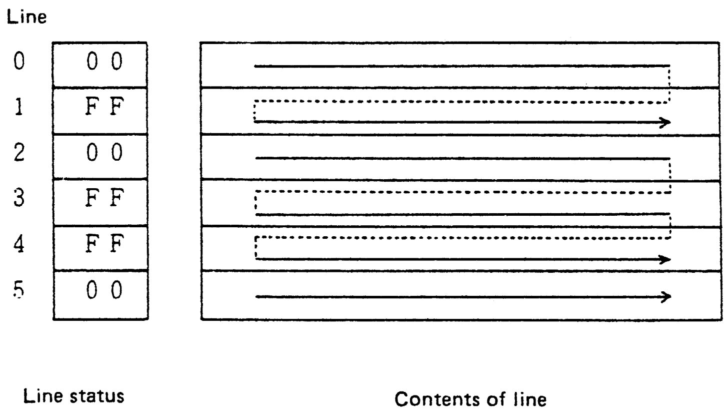


Fig. 15-2 Line Status

In Fig. 9-2, Øth and 1st lines, 2nd through 4th lines, and 5th line are logical single lines, respectively. The conditions for composing a logical single line are detailed in Section 9.4, Virtual Screen Functions.

(8) Cursor and cursor margin

The cursor indicates the position of a character to be displayed. At the same time, it also serves as a reference point for screen control.

The cursor is designed to always stay within the window. If the cursor moves out of the window, the window also moves with the cursor. When the cursor is at either end of the window, the next character cannot be identified. Therefore, a certain width from either end of the window must be predetermined so that the window moves when the cursor reaches this position. This width is called a "cursor margin".

In the following example, the screen size of 40 columns by 8 lines has been defined for LCD display. Assume that the cursor margin is set to a value of 3 and the position of the right margin is "RM", while that of the left margin is "LM". When the cursor is between the positions "LM" and "RM" (i.e., the shaded section in Fig. 9-3), window movement will not take place. When the cursor moves and reaches position "RM" (3rd column from the right), the cursor will not advance; instead the window will move to the right even if an attempt is made to move the cursor. This movement of the window stops when it reaches the right end of the virtual screen. From this point, the cursor moves again.

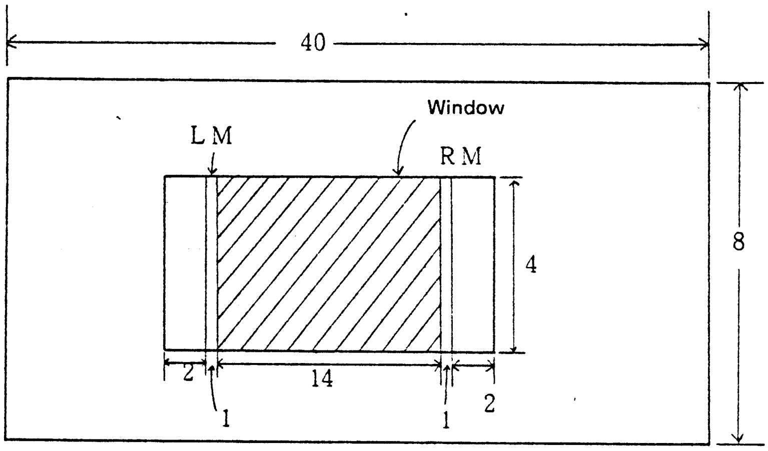


Fig. 15-3 Cursor Margin and Window Movement

The cursor margin may be specified by a value in the range of 1 to 10. If the value is 1, it indicates that no cursor margin is specified.

(9) List flag

If the window moves so that it contains the cursor, the displayed data is difficult to read. In some cases, the window may be desired to be fixed at the left end of the virtual screen (e.g., LIST command in BASIC.) The list flag controls the movement of the window. When the list flag is set, the window moves along the left end of the virtual screen (see the shaded section in Fig. 9-4).

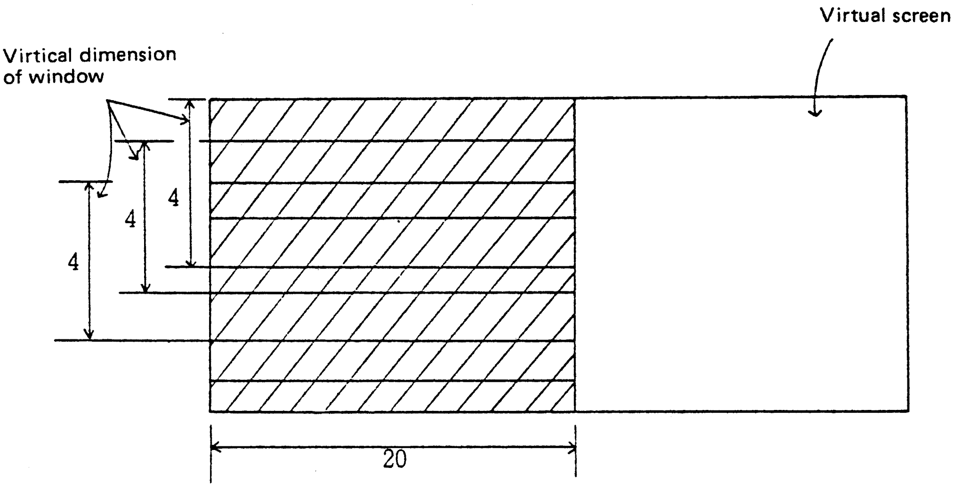


Fig. 15-4 Moving Range of Window when List Flag is ON.

When the list flag is ON, the window cannot move horizontally. However, its vertical movement is not restricted.

(10) Access pointer

When a character is to be input or output to or from the display controller, the location (i.e., coordinates on the virtual screen) where the character may be accessed for input/output must be specified. The access pointer functions to indicate this location.

15.3 Graphic Display

Only character codes may be handled by the virtual screen. It cannot handle graphic data. Graphic data processing is supported by both the LCD and display controller but in a manner different supported by both the LCD and display controller but in a manner different from each other.

(1) LCD

Graphic data is processed only on the physical screen. Display functions such as dot ON/OFF, straight line drawing, etc., are controlled directly against the controller. Therefore, the contents of the virtual screen will not be lost even if the graphic display is activated.

(2) Display controller

On the display controller, both text and graphic data cannot be displayed concurrently. Therefore, either the mode to effect the text display or that to effect the graphic display must be

selected by changing the display mode. Moreover, because of the limited memory size of the display controller, the contents of the virtual screen will be lost when the graphic display is activated.
The display controller is capable of color selection, which is different depending on the display mode. In Text display mode, the background colors are green or orange with the color of all characters fixed. In Graphic display mode, there are two color sets 0 and 1. All the colors in the same color set can be used as background colors. Other colors are available for dots.

Color set 0	Color set 1
Green	White
Yellow	Cyan
Blue	Magenta
Red	Orange

15.4 Virtual Screen Control

The movement of the cursor, deletion of one character, and other controls related to the display contents on the virtual screen are performed by using character codes. Spe~~c~~ial controls such as screen size specification, list control, etc., are provided as the functions of the virtual screen.
The character codes used are 00 through FF. Codes 20 through FF are those to be displayed on the screen as graphic characters. Code 00 through 1F are non-graphic characters which are not displayed on the screen. They are used as control characters for cursor movement, etc.
The description of each character code follows.

(1) Graphic characters
 (a) When not at the right-hand end on the bottom line
  The next line is assumed to be a continuation line. (Line status is FF.) (See Fig. 9-6.)
 (b) When at the right-hand end on the bottom line
  The display contents are scrolled up by one line. The bottom line becomes a continuation line filled with blank codes (20).

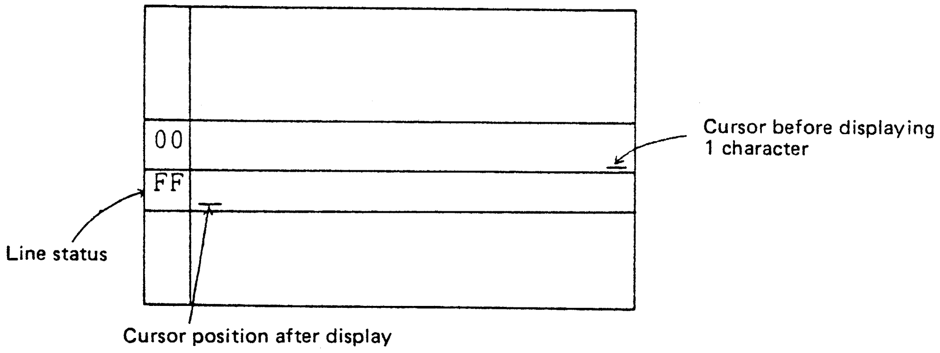

00
FF
Cursor before displaying 1 character
Line status
Cursor position after display

Fig. 15-5 Continuation Line

(2) Control codes

19 character codes can be used as control codes. The functions of the respective control codes are as follows:

(a) Ø1 (Window Left)

Positions the window to the left end of the virtual screen. The cursor moves to the 1Øth column of the window.

(b) Ø4 (Scroll Right)

Moves the window to the right by the number of columns specified by the horizontal scroll steps. However, the window will not move beyond the right end of the virtual screen.

(c) Ø5 (Clear to End of Line)

Changes all the characters from the cursor position to the end of the logical single line to blank codes (2Ø).

(d) Ø6 (Window Right)

Positions the window to the right end of the virtual screen. The cursor moves to the 1Øth column of the window.

(e) Ø8 (Delete one Character)

Moves the cursor position back by one character and deletes the character at the cursor position. All the data following the deleted character on the line are shifted and a blank code (2Ø) is entered at the end of the line. When the cursor is at the beginning of the line and therefore cannot be moved back, the character at the current cursor position is deleted.

(f) Ø9 (TAB)

Moves the cursor to the next Tab position. Tab positions are set at every 8 columns such as Ø, 8, 16 .....

(g) ØA (Line Feed)

Moves the cursor down by one line. When the cursor is at the bottom line of the virtual screen, the virtual screen scrolls one line and the bottom line will be filled with blank codes.

(h) ØB (Home)

Positions the cursor to the upper left corner of the virtual screen. The window moves along with the cursor. (This position is referred to as "Home position").

(i) ØC (Clear)

Changes all the contents of the virtual screen to blank codes (2Ø). The logical single line is set to the virtual screen width and the cursor returns to the home position.

(j) ØD (Carriage Return)

Terminates the logical single line. (The line status of the next line becomes ØØ.) The cursor moves to the left end of the line.

(k) 1Ø (Scroll Up)

Moves the window up by the number of lines specified by the vertical scroll steps. The window will not move beyond the top end of the virtual screen. The cursor moves to the 1Øth column of the virtual screen.

(l) 11 (Scroll Down)

Moves the window down by the number of lines specified by the vertical scroll steps. The window will not move below the bottom end of the virtual screen. The cursor moves to the 1Øth column of the virtual screen.

(m) 12 (Insert)
Inserts a blank code (20) into the cursor position. All the characters following the cursor position are shifted to the right by 1 column. If the last character in the logical single line is a blank code, that character is deleted. If the last character is not a blank code, another line filled with blank codes will be inserted (i.e., scrolling takes place above the cursor position) and the last character is positioned at the beginning of the inserted line.

(n) 13 (Scroll Left)
Moves the window to the left by the number of columns specified by the horizontal scroll steps. However, the window will not move beyond the left end of the virtual screen.

(q) 1A (Clear to End of Screen)
Changes the contents of the virtual screen from the current cursor position to the end of the virtual screen to blank codes. The logical single line is set to the virtual screen width. (Line status is changed to "00".)

(p) 1C (Cursor Right)
Moves the cursor to the right by one column. The cursor at the right end of a line will move to the beginning of the next line. If the cursor is on the bottom line, it will move to the left end of the same line.

(q) 1D (Cursor Left)
Moves the cursor to the left by one column. The cursor at the left end of a line will move to the right end of one immediately above the line. If the cursor is in the top line, it will move to the right end of the same line.

(r) 1E (Cursor Up)
Moves the cursor up by one line. The cursor will not move if it is in the top line.

(s) 1F (Cursor Down)
Moves the cursor down by one line. The cursor will not move if it is in the bottom line.

(3) Subroutine call for virtual screen
The virtual screen is supported by subroutine "SCRFNC". Parameters for this subroutine are given by the parameter packet used on the memory. The packet begins with a 1-byte function code which is followed by a series of several data. The return information is also included in the packet.

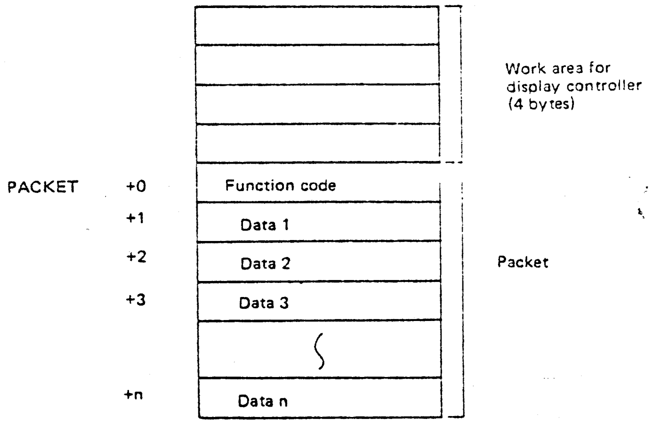


Fig. 15-6 Parameter Packet

A 4-byte work area is required before the packet for the display controller functions. (See Fig. 9-6.)
To call subroutine "SCRFNC" (Entry point FF5E), the top address of the packet must be given to the index register.
Example: To call the function to set the virtual screen. In this example, the screen size of 40 x 8 is defined for the LCD and the buffer address is specified at 5000.

```
SCRFNC    EQU    $FF5E
          LDX    #PACKET
          JSR    SCRFNC
           :
           :
           :
PACKET    FCB    $87        *Function   code(define screen  size)
          FCB    39         *Screen  width
          FCB     7         *Screen  depth
          FDB    $5000      *Buffer  address
           :
           :
           :
          ORG    $5000
          RMB    40× 9 + 1  *Buffer size
```

° Functions for Initialization of Virtual Screen

The following functions must be executed to initialize the virtual screen.

1. Function 84 (Screen device select)
2. Function 87 (Specification of screen size and buffer address)
3. Function C3 (Specification of scroll margin)
4. Function C4 (Specification of scroll steps)
5. Function CB (Specification of scrolling speed)

(Refer to Section 9.5 for detailed description of each function code.)

15.5 Virtual Screen Function Table

Packets for the virtual screen are as listed below. The virtual screen functions are divided into those shared by both the LCD and the display controller and those peculiar to either one. The device to which the particular function is applicable is shown in the "Function (Application)" column as (LCD) for the display and as (Disp) for the display controller. Data in each packet are numbered as 0, 1, 2, ... for each byte and their descriptions are given in order of the data number. These packets are used at the time of both the entry and return of each subroutine. In the following table, "XX" indicates that arbitrary 2-digit values may be used, and unless otherwise specified, all numeric values are hexadecimal.

Function (Application)	Packet data number	Description	
		(At Entry)	(At Return)
84		Screen device select.	
(Disp) (LCD)	00	84 (Function code) Device code 30: Disp 22: LCD	Return code 00: Normal FE: Device is not connected. FF: Device specification is invalid.
85		Initialization of the Display controller. The Display controller is initialized at the default value.	
(Disp)	00 01	85 (Function code) XX	Return code 00: Normal FF: I/O error
87		Specification of the virtual screen. By this function, the screen size and the top address of the buffer are specified. When the screen size is m columns by n lines, the size of the buffer must be m x n + 1 bytes.	
(Disp) (LCD)	00 01	87 (Function code) Screen width (Specify m - 1 for m columns.) Screen length (Specify n - 1 for n lines.) High-order byte of buffer's top address Low-order byte of buffer's top address NOTE: Buffer addressing is not required if the display controller is specified.	Return code 00: Normal FF: Screen oversize

Function (Application)	Packet data number	Description	
		(At Entry)	(At Return)
88		Input of the virtual screen size. By this function, the currently defined size of the virtual screen is obtained.	
(Disp) (LCD)	00	88 (Function code)	
	01	XX	Screen width (m - 1 for m columns)
	02	XX	Screen length (n - 1 for n lines)
89		Input of the window size.	
(Disp) (LCD)	00	89 (Function code)	
	01	XX	Width: 19 (D) for LCD 31 (D) for Display controller
	02	XX	Length: 3 (D) for LCD 15 (D) for Display controller
8A		Input of the window position. By this function, the coordinate values at the upper left corner of the window on the virtual screen are given.	
(Disp) (LCD)	00	8A (Function code)	
	01	XX	Coordinate x
	02	XX	Coordinate y
8C		Input of the cursor position. By this function, the position of the cursor on the virtual screen is obtained.	
(Disp) (LCD)	00	8C (Function code)	
	01	XX	Coordinate x
	02	XX	Coordinate y
8D		Input of the cursor margin value.	
(Disp) (LCD)	00	8D (Function code)	
	01	XX	Margin value
8E		Input of the scroll steps.	
(Disp) (LCD)	00	8E (Function code)	
	01		Number of horizontal scroll steps
	02		Number of vertical scroll steps

Function (Application)	Packet data number	Description	
		(At Entry)	(At Return)
8F		By this function, the dot status at the specified position on the physical screen is obtained.	
(Disp) (LCD)	00	8F (Function code)	XX
	01	High-order byte of coordinate x	(1) LCD FF: ON 00: OFF (2) Display controller Color code
	02	Low-order byte of coordinate x	XX
	03	High-order byte of coordinate y	XX
	04	Low-order byte of coordinate y	XX
91		Input of the range of the logical single line. By this function, the range of the logical single line containing the cursor on the virtual screen is specified.	
(Disp) (LCD)	00	91 (Function code)	XX
	01	XX	First column in the logical single line (Coordinate x with a value 0)
	02	XX	First line in the logical single line (Coordinate y)
	03	XX	Physical screen width (LCD: 19 (D), Disp: 32 (D))
	04	XX	Last line in the logical single line (Coordinate y)
92		Display of one character on the virtual screen.	
(Disp)	00	Character code	Coordinate x of the new cursor position
(LCD)	01	XX	Coordinate y of the new cursor position
93		Specification of a display mode for the display controller.	
(Disp)	00	93 (Character code)	XX
	01	Text mode 00: Graphic mode 01: Text mode	Return code 00: Normal FF: An error has occurred.
	02	Graphic mode 00: Text mode 01: Color graphic mode 02: Monochromatic graphic (high-resolution) mode Note: Text mode and graphic mode must be specified exclusively. In other words, either data 00 or 01 must be 00.	XX

Function (Application)	Packet data number	Description	
		(At Entry)	(At Return)
		Graphic mode is supported on the physical screen. The resolution of the display is 128 x 64 dots in color graphic mode and 128 x 96 dots in monochromatic graphic mode (i.e., high-resolution mode).	
	03	Background color 00: Green 04: White 01: Yellow 05: Cyan 02: Blue 06: Magenta 03: Red 07: Orange Note: Background color selection is effective in Graphic mode only. A color set is defined by the COLOR command in Text mode.	XX
95		Input of one character on the display. By this command, the character at the coordinates specified by the access pointer is input.	
(Disp)	00	95 (Function code)	XX
	01	XX	Character code
	02	XX	Color code (Background color code)
97		Consecutive input of characters from the virtual screen. By this function, characters are input in the number specified from the coordinate positions where data read starts.	
(Disp) (LCD)	00	97 (Function code)	XX
	01	Coordinate x at the read start point	Input character 1
	02	Coordinate y at the read start point	Input character 2
	03	Number of read characters	Input characters 3
	04	XX	
98		Display of one character on the virtual screen. (Note that the packet generation in this case is different from that of function 92.)	
	00	98 (Function code)	XX
(Disp) (LCD)	01	XX	Coordinate x of the new cursor position
	02	XX	Coordinate y of the new cursor position
	03	XX	First line number in the logical single line containing the new cursor (Coordinate y)
	04	XX	Last line number in the logical single line containing the new cursor (Coordinate y)

Function (Application)	Packet data number	Description	
		(At Entry)	(At Return)
CØ		Setting of the window position. By this function, the upper left edge of the window is positioned at the specified address on the virtual screen.	
(Disp) (LCD)	ØØ	CØ (Function code)	XX
	Ø1	Coordinate x on the virtual screen	XX
	Ø2	Coordinate y on the virtual screen Note: If the window position is outside the bounds of the virtual screen, the maximum values are set for both coordinates x and y.	XX
C2 (Disp) (LCD)		Specification of the cursor position. By this function, the cursor is placed at the specified position on the virtual screen, resulting in the movement of the window.	
	ØØ	C2 (Function code)	XX
	Ø1	Coordinate x of the cursor position	XX
	Ø2	Coordinate y of the cursor position Note: The window movement is controlled as follows: (1) The window does not move when the specified cursor position is within the window area. (2) When the specified cursor position is not within the window area, the window moves so that the new cursor is located at the home position of the window. The cursor position cannot be located at the home position of the window, if the bottom edge of the window is in alignment with the bottom edge of the virtual screen. In such a case, the cursor position is set within the window area according to the same rule as that of function code CØ.	XX
C3 (Disp) (LCD)		Setting of the value of the cursor margin.	
	ØØ	C3 (Function code)	XX
	Ø1	Cursor margin value (This value must be in the range from 1 to half the window value.)	XX
C4 (Disp) (LCD)		Setting of the number of scroll steps.	
	ØØ	C4 (Function code)	XX
	Ø1	Number of horizontal scroll steps (Ø to 255 (D))	XX
		Number of vertical scroll steps (Ø to 255 (D))	XX

Function (Application)	Packet data number	Description	
		(At Entry)	(At Return)
C5 (Disp) (LCD)	00	Turning the list flag ON.	
		C5 (Function code)	XX
C6 (Disp)	00	Resetting of the list flag.	
		C6 (Function code)	XX
C7		Setting of a dot at the specified position. This function is effective in Graphic mode.	
(Disp) (LCD)	00	C7 (Function code)	XX
	01	High-order byte of coordinate x	XX
	02	Low-order byte of coordinate x	XX
	03	High-order byte of coordinate y	XX
	04	Low-order byte of coordinate y	XX
	05	Color code With LCD, 00: OFF, FF: ON With Display controller, if color set 0 is specified 00: Green 01: Yellow 02: Blue 03: Red if color set 1 is specified 00: White 01: Cyan 02: Magenta 03: Orange	
C8		Drawing a straight line between any two points on the graphic screen.	
(Disp) (LCD)	00	C8 (Function code)	XX
	01	High-order byte of coordinate x at the start point	XX
	02	Low-order byte of coordinate x at the start point	XX
	03	High-order byte of coordinate y at the start point	XX
	04	Low-order byte of coordinate y at the start point	XX
	05	High-order byte of coordinate x at the end point	XX
	06	Low-order byte of coordinate x at the end point	XX
	07	High-order byte of coordinate y at the end point	XX
	08	Low-order byte of coordinate y at the end point	XX
	09	Color code. Same as function code 07	XX

Function (Application)	Packet data number	Description	
		(At Entry)	(At Return)
C9		Termination of the logical single line. By this function, the line status of the specified line is reset to ØØ.	
(Disp)	ØØ	C9 (Function code)	XX
(LCD)	Ø1	Line number (coordinate y)	XX
CA		Clearing of the screen in Graphic mode. This function is effective for the graphic screen when the Display controller is used, and for the physical screen when the LCD display is used.	
(Disp)	ØØ	CA (Function code)	XX
	Ø1	Background color (Effective only with Display controller)	XX
CB		Setting of the scrolling speed. This function specifies the scrolling speed of the physical screen.	
(LCD)	ØØ	CB (Function code)	XX
	Ø1	Speed A value in the range of ØØ to Ø9 is used to specify the scrolling speed. 9 is the highest scrolling speed.	
CD		Output of one character to the position specified by the access pointer.	
(Disp)	ØØ	CD (Function code)	XX
	Ø1	Character code	XX
CE (Disp)		Specification of the access pointer. By this function, the character position that can read/write on the virtual screen when the Display controller is used is specified.	
	ØØ	CE (Function code)	XX
	Ø1	Coordinate x of the access pointer	XX
	Ø2	Coordinate y of the access pointer	XX
CF		Specification of a color set. Two color sets each consisting of 4 different colors are selectable when the Display controller is used.	
(Disp)	ØØ	CF (Function code)	XX
	Ø1	Color set ØØ: Color set Ø Ø1: Color set 1 If color set Ø is specified, green, yellow, blue and red can be used. If color set 1 is specified, white, cyan, magenta and orange can be used.	XX

15.6 EPSP Message Format Table for Screen

In the following table, SS and MM refer to the slave and master device numbers, respectively. Numeric values are all hexadecimal.
"XX" indicates that arbitrary 2-digit values may be used.

(1) Function Codes for Display Controller

Function code	FMT	DID	SID	FNC	SIZ	Text data number	Description of function and text
84	00	SS	MM	84	00	00	Screen device select Device number (30)
	01	MM	SS	84	00	00	Return code 00: Normal FE: Device is not ready. FF: Device number is invalid.
85	00	SS	MM	85	00	00	Initialization of screen XX
	01	MM	SS	85	00	00	Return code 00: Normal FF: An error has occurred.
87	00	SS	MM	87	03	00	Specification of the screen size Virtual screen width (maximum value of coordinate x)
						01	Virtual screen length (maximum value of coordinate y)
						02	XX
						03	XX
	01	MM	SS	87	00	00	Return code 00: Normal FF: Size specification is invalid.
88	00	SS	MM	88	00	00	Input of the virtual screen size. XX
	01	MM	SS	88	01	00	Virtual screen width (maximum value of coordinate x)
						01	Virtual screen length (maximum value of coordinate y)
89	00	SS	MM	89	00	00	Input of the physical screen
	01	MM	SS	89	01	00	Screen width (maximum value of coordinate x)
						01	Screen length (maximum value of coordinate y)

Function code	FMT	DID	SID	FNC	SIZ	Text data number	Description of function and text
C0							Positioning of the physical screen on the virtual screen. Position values are given with respect to the position (0,0) of the physical screen.
	00	SS	MM	C0	01	00	Coordinate x of the specified position.
						01	Coordinate y of the specified position.
	01	MM	SS	C0	00	00	xx
8A							Input of the physical screen position on the virtual screen. Position values are given with respect to the position (0,0) of the physical screen.
	00	SS	MM	8A	00	00	XX
	01	MM	SS	8A	01	00	Coordinate x of the specified position
						01	Coordinate y of the specified position
C2							Specification of the cursor position on the virtual screen.
	00	SS	MM	C2	01	00	Coordinate x of the specified position
						01	Coordinate y of the specified position
	01	MM	SS	C2	00	00	XX
8C							Input of the cursor position on the virtual screen
	00	SS	MM	8C	00	00	XX
	01	MM	SS	8C	01	00	Coordinate x of the specified position
						01	Coordinate y of the specified position
C3							Setting of the margin value of the cursor
	00	SS	MM	C3	00	00	Margin value
	01	MM	SS	C3	00	00	XX
8D							Input of the cursor margin value
	00	SS	MM	8D	00	00	XX
	01	MM	SS	8D	00	00	Margin value
C4							Setting of the number of scroll steps
	00	SS	MM	C4	01	00	Number of horizontal scroll steps
						01	Number of vertical scroll steps
	01	MM	SS	C4	00	00	XX

Function code	FMT	DID	SID	FNC	SIZ	Text data number	Description of function and text
8E							Input of scroll steps
	00	SS	MM	8E	00	00	XX
	01	MM	SS	8E	01	00	Number of horizontal scroll steps
						01	Number of vertical scroll steps
C5							Setting of the list flag
	00	SS	MM	C5	00	00	XX
	01	MM	SS	C5	00	00	XX
C6							Resetting of the list flag
	00	SS	MM	C6	00	00	XX
	01	MM	SS	C6	00	00	XX
C7							Setting of a dot at the specified position
	00	SS	MM	C7	04	00	High-order byte of coordinate x
						01	Low-order byte of coordinate x
						02	High-order byte of coordinate y
						03	Low-order byte of coordinate y
						04	Color code
	01	MM	SS	C7	00	00	XX
8F							Input of the dot status at the specified position
	00	SS	MM	8F	03	00	High-order byte of coordinate x
						01	Low-order byte of coordinate x
						02	High-order byte of coordinate y
						03	Low-order byte of coordinate y
	01	MM	SS	8F	00	00	Color code
							Drawing of a straight line
C8	00	SS	MM	C8	08	00	High-order byte of coordinate x at the start point
						01	Low-order byte of coordinate x at the start point
						02	High-order byte of coordinate y at the start point
						03	Low-order byte of coordinate y at the start point
						04	High-order byte of coordinate x at the end point
						05	Low-order byte of coordinate x at the end point
						06	High-order byte of coordinate y at the end point
						07	Low-order byte of coordinate y at the end point
						08	Color code
	00	MM	SS	C8	00	00	XX

Function code	FMT	DID	SID	FNC	SIZ	Text data number	Description of function and text
91							Input of the range of the logical single line containing the cursor
	00	SS	MM	91	00	00	XX
	01	MM	SS	91	03	00	00
						01	Coordinate y of the first line in the logical single line
						02	Column size of the physical screen
						03	Coordinate y of the last line in the logical single line
C9							Resetting of the line status of the specified line (i.e., partitioning of the logical single line)
	00	SS	MM	C9	00	00	Line number
	01	MM	SS	C9	00	00	XX
92							Display of one character on the virtual screen
	00	SS	MM	92	00	00	Character code
	01	MM	SS	92	01	00	Coordinate x of the cursor Coordinate y of the cursor
CA							Specification of the background color in Graphic mode
	00	SS	MM	CA	00	00	Color code
	01	MM	SS	CA	00	00	XX
CB							Setting of the scrolling speed
	00	SS	MM	CB	00	00	A value in the range of 0 to 9 is used to specify the scroll.
	01	MM	SS	CB	00	00	XX
93							Specification of display mode
	00	SS	MM	93	02	00	Text mode 00: Mode other than Text mode 01: Text mode
						01	Graphic mode 00: Mode other than Graphic mode 01: Graphic mode 1 02: Graphic mode 2 Note: Either data 00 or data 01 must be "00". Both data must not be "00".
						02	Background color 00: Green 01: Yellow 02: Blue 03: Red 04: White 05: Cyan 06: Magenta 07: Orange
	01	MM	SS	93	00	00	Return code 00: Normal FF: An error has occurred.

Function code	FMT	DID	SID	FNC	SIZ	Text data number	Description of function and text
CD							Writing of one character into the access pointer
	00	SS	MM	CD	01	00	Character code
						01	Color code
	01	MM	SS	CD	00	00	XX
CE							Specification of the access pointer against the virtual screen
	00	SS	MM	CE	01	00	Coordinate x
						01	Coordinate y
	01	MM	SS	CE	00	00	XX
95							Input of one character from the access pointer
	00	SS	MM	95	00	00	XX
95	01	MM	SS	95	01	00	Character code
						01	Color code
CF							Selection of a color set
	00	SS	MM	CF	00	00	00: Color set 0 01: Color set 1
	01	MM	SS	CF	00	00	XX
97							Input of characters at the positions specified consecutively on the virtual screen
	00	SS	MM	97	03	00	Coordinate x of the start point
						01	Coordinate y of the start point
						02	High-order byte of the number of input characters
						03	Low-order byte of the number of input characters
	01	MM	SS	97	mm	00 mm	00~mm denote the character codes of the input characters.
98							Display of one character on the virtual screen followed by the input of the first and last line numbers of the logical single line including the newly set cursor position.
	00	SS	MM	98	00	00	Character code.
	01	MM	SS	98	03	00	Coordinate x of the new cursor position
						01	Coordinate y of the new cursor position
						02	First line number in the logical single line
						03	Last line number in the logical single line.

15 · 21 L

CHAPTER 16 - MENU

Page

16.1 General 16-1
16.2 ID Structure 16-2
16.3 Examples 16-4
16.4 MENU Work Areas 16-5

16.1 General

The title or entry point of a program can be registered or displayed by the MENU function of the HX-20. This chapter first describes the ID structure of the application programs stored in the ROMs of the HX-20, then explains how the ID information is displayed by the MENU with examples.

16.2 ID Structure

The ID (identifying information also called a "header") for both the ROM and user (RAM) application programs is structured as described below.
When the user writes an application program into a ROM or RAM and wishes to display the program on the MENU, he must write the header information to identify the program. Particularly, for an application program stored in a ROM, this header information must be at the top address (low-order address) of the ROM.

16.2.1 Header of ROM/RAM application program

(1) ID 1 (1 byte)

Bit 0 - bit 6 ':' (Code 3A)

Bit 7

0 = The header contains a link address to the next program on the ROM or RAM. The linked program is not displayed on the MENU. This bit can be used when the user writes programs using an EPROM. In other words, if the user wishes to erase a program on the EPROM, bit 7 should be changed from logic "1" to "0" using an EPROM writer. (Bit 7 is "1" with the EPROM in the initialized state.)

1 = Header contains a link address with the next program on the ROM or RAM, the starting address (entry point) of a program and its program name.

(2) ID 2 (1 byte)

Bit 0 - bit 6 = The header information contains one of the following codes:

"A" = Application program (application for general use)

"B" = BASIC interpreter

"E" = End of link (No application program follows this header information.)

[illegible] (RAM application for general use)

Bit 7

0 = Indicates that the linkage with the next program is an absolute value (i.e., absolute address).

1 = Indicates that the linkage with the next program is a relative value (i.e., offset value from the header).

If the ROMs are made available for use on any sockets, programs are relocatable and thus bit 7 must be set to logic "1".

(3) Pointer to next header (2 bytes)

This header information is also called a "link address".
This two-byte data is used as a pointer to the location of the next header. If no next header exists within the same ROM, the value of this data is "FFFF".
If the MENU finds value "FFFF" on a ROM, it scans the next ROM for header information.

(4) Starting address of program (entry point) (2 bytes)
This header information indicates the starting address of a program. The starting address is an absolute value if the bit 7 of ID2 is logic "1" and an offset value from the beginning of this header information if bit 7 is "0".

(5) Filename (program name) (17 bytes max.)
A filename is entered in a maximum of 16 bytes in ASCII code. The last byte of this header information is always "00".

16.2.2 Header of BASIC application program

The header of a BASIC application program (i.e., an application program written in BASIC by the user) is different from that of a ROM/RAM application program.

BASIC application programs have no linkage with ROM/RAM application programs. However ROM/RAM application programs are displayed automatically by the MENU function.

(1) Link offset (2 bytes)
This is a pointer to indicate the starting address of the header of the next BASIC program. For example, program 1 points at program 2, while program 2 points at program 3. When the link offset value is FFFF, it indicates that no next header exists.

(2) Filename (program name) (8 bytes)
The filename of a program is specified by the TITLE command of BASIC. If the program has no filename, blanks must be entered as the filename in the header.

16.2.3 Bit map (2 bytes) and link tables (4 bytes, 013C to 013F)

After the input of "CTRL/@"; the MENU generates a bit map which indicates the presence of the header of a ROM application program, and a link table for linkage with an RAM application program. Bit map addresses are 013A and 013B.

013A	Bit 7	*	ROM at addresses E000 to FFFF of bank 0
	Bit 6	*	ROM at addresses C000 to DFFF of bank 0
	Bit 5	*	ROM at addresses A000 to BFFF of bank 0
	Bit 4	*	ROM at addresses 8000 to 9FFF of bank 0
	Bit 3	*	ROM at addresses 6000 to 7FFF of bank 0
	Bit 2	*	ROM at addresses 4000 to 5FFF of bank 0
	Bit 1	*	ROM at addresses 2000 to 3FFF of bank 0
	Bit 0	*	ROM at addresses 0000 to 1FFF of bank 0
013B	Bit 7	*	ROM at addresses E000 to FFFF of bank 1
	Bit 6	*	ROM at addresses C000 to DFFF of bank 1
	Bit 5	*	ROM at addresses A000 to BFFF of bank 1
	Bit 4	*	ROM at addresses 8000 to 9FFF of bank 1
	Bit 3	*	ROM at addresses 6000 to 7FFF of bank 1
	Bit 2	*	ROM at addresses 4000 to 5FFF of bank 1
	Bit 1	*	ROM at addresses 2000 to 3FFF of bank 1
	Bit 0	*	ROM at addresses 0000 to 1FFF of bank 1

```
* = 0 :   No header exists in the specified ROM socket.
* = 1 :   Header exists in the specified ROM socket.
Bank 0 : Main memory of HX-20
Bank 1 : Memory in the expansion unit for HX-20
```

The link table after the input of "CTRL/@" contains 4-byte data "1:/'E'/FF/FF/". If the user wishes to display any program on the RAM in the MENU, he just needs to link this 4-byte data in the link table to his object program. For example, if the user writes an application program from address 1000, the header of the RAM application program and its link table should be written as follows.

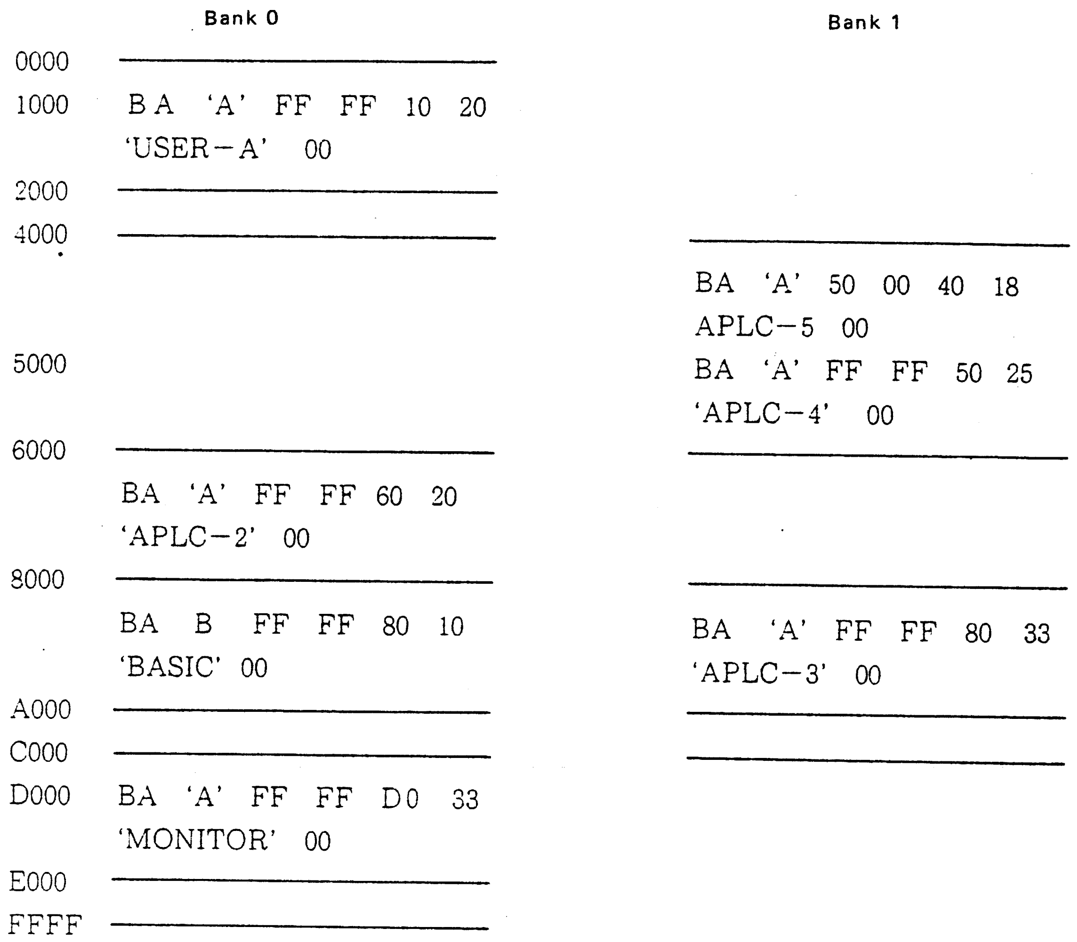
```
1000    /: (bit7=1)/'A'/FF/FF/10/20/U/S/E/R/00/
13C     /: (bit7=0)/'A'/10/00/
```

16.2.4 How bit map and link table are generated

Neither a bit map nor a link table exists before the HX-20 system is initialized (by pressing CTRL and @ keys) (see Section 1.3). Before the system is cold started by "CTRL/@", "CTRL/@ Initialize", "1 MONITOR" and dummy names (19 max.) will appear in the MENU on the LCD. After pressing CTRL and @ keys, the MENU generates a bit map and a link table. When generating a bit map by the MENU, program linking starts from address D000 (MONITOR). Next, scanning of addresses starts from A000 (bank 1 also if an expansion unit is connected) and progresses to addresses 8000, 6000 and 4000 in the order named. The MENU sets the bit map depending on whether or not the header of an application program exists, and writes "/:/'E'/FF/FF/" into the link table.

Subsequently, the MENU displays the filename of a ROM application filename according to the bit map. Next, if there is any linked RAM (user) application program, then the name of the RAM application program is displayed, followed by BASIC application programs.

16.3 Examples



Assume that there are 2 BASIC application programs (APLC-1 and APLC-2) in addition to the above ROM/RAM application program.
The bit map in this case will be as follows:

```
                MSB       LSB
    13A           01011000
    13B           00010100
```

and the link table will be as follows.

```
    13C           /:/'A'/10/00
```

The following information will appear in the MENU on the LCD display.

```
    CTRL/@        Initialize
    1 MONITOR
    2 BASIC
    3 APLC-3
    4 APLC-2
    5 APLC-5
    6 APLC-4
    7 USER-A
    8 APLC-1
    9 APLC-2
```

16.4 MENU Work Areas

Address (from)	(to)	Variable name	Bytes	Description
2D0	48A	SCNBUF	442	Buffer for MENU display
78	78	INTFLG	1	Initialize flag (0: Request; 1: Complete) Bit 0: MENU Bit 7: BASIC Condense (garbage collection) flag (1: Condense request) Bit 6: (BASIC, Application) Condense
7B	7B	RUNMOD	1	Run mode 01: MENU
7E	7E	SFTSWH	1	Software switch 1 Bit 4: Bank switch number currently selected (0: Bank 0; 1: Bank 1) Bit 5: Bank switch number selected before current number Bit 6: Bank number in which BASIC programs are stored Bits 5 and 6 are used to condense application.
80	81	TMPBF1	2	Temporary buffer
82	83	CNTMNU	2	Indicates the top address of ROM (C000, A000, 8000, ...).
84	84	CNTMNU	1	Number of items currently on the MENU display - 1
85	85	MNUNUB	1	MENU number
86	86	BITMP	1	Bit map value of a bank (for temporary use)
87	87	BBTMP0	1	Buffer for BITMP0 (bit map of bank 0)
88	88	BBTMP1	1	Buffer for BITMP1 (bit map of bank 1)
89	89	STKLIN	1	Maximum number of lines on MENU display
8A	8A	MXMNUB	1	Maximum number of MENUs (ASCII code)
8B	8B	BSAPNB	1	BASIC application number
8C	8C	CNTFLG	1	Work area for temporary use
8D	8D	DISFLG	1	Work area for temporary use

Address (from)(to)		Variable name	Bytes	Description
8E	92	PCKT	5	LCD buffer work area for virtual screen packet
13A	13A	BITMP0	1	Bit map for bank 0 Indicates whether the header of a ROM application program exists in one of the ROM chips in bank 0. (0: No header exists; 1: A header exists.) Bit 0: Address 0000 of bank 0 Bit 1: Address 2000 of bank 0 Bit 2: Address 4000 of bank 0 Bit 3: Address 6000 of bank 0 Bit 4: Address 8000 of bank 0 Bit 5: Address A000 of bank 0 Bit 6: Address C000 of bank 0 Bit 7: Address E000 of bank 0
13B	13B	BITMP1	1	Bit map for bank 1 Indicates whether the header of a ROM application program exists in one of the ROM chips in bank 1. (0: Header does not exist; 1: Header exists.) Bit 0: Address 0000 of bank 1 Bit 1: Address 2000 of bank 1 Bit 2: Address 4000 of bank 1 Bit 3: Address 6000 of bank 1 Bit 4: Address 8000 of bank 1 Bit 5: Address A000 of bank 1 Bit 6: Address C000 of bank 1 Bit 7: Address E000 of bank 1
13C	140	LNKTBL	4	Link table for RAM application programs (I) When RAM application program does not exist : E FF FF (II) When the header of a RAM application program exists : A Address of the RAM application program

CHAPTER 17 MONITOR

Page

17.1 General...17-1
17.2 About Trap..17-2

17.1 General

The Monitor is located in the ROM (ROM2) area from C000 to DFFF and has two entry points DFF7-DFF9 and DFFA-DFFC. The former is for entry from the menu display, etc., while the latter is for entry when a trap interrupt is generated. If one of the trap interrupt addresses (0106 through 0108) is specified, the default assumption is "JMP $DFFA". The display of data by the Monitor is always on the physical screen and the virtual screen is never used for the monitor display.
The HX-20 Monitor has 10 types of commands as listed below.

(1) S (Set) command : Displays and changes the contents of the memory.
(2) D (Dump) command : Displays the contents of the memory.
(3) G (Go) command : Executes a program.
(4) X (Examine) command : Displays and changes the contents of each register.
(5) R (Read) command : Loads a program or data into the memory from an external storage.
(6) W (Write) command : Saves the contents of the memory to an external storage.
(7) V (Verify) command: Verifies the data output to an external storage.
(8) K (Key) command : Specifies the data for automatic key input when the power switch is turned ON.
(9) A (Address) command : Specifies the range of the memory space when loading from an external storage or saving data to an external storage.
(10) B (Back) command : Returns control to the procedure by which the Monitor was called.

Refer to the HX-20 OPERATION MANUAL for detailed description of each monitor command.

17.2 About Trap

If an attempt to execute a command not defined for the MCU is made, a trap interrupt is generated. By utilizing this characteristic, a breakpoint is set by the G command. For example, write "00" (undefined code) in the address specified as a breakpoint. Then, try to execute the command at this address, and a trap interrupt will be generated, causing the HX-20 to return to the Monitor mode again.

Address (from)	(to)	Variable name	Bytes	Description
2A0	2A1	BP1	2	Stores the address specified as a breakpoint.
2A2	2A2	BPD1	1	Stores the contents of the breakpoint address.
2A3	2A3	LCDSTS	1	Stores the LCD status ('DISSTS': Address 0280) when the HX-20 enters Monitor mode.
2A4	2BE		27	Work area for packets of binary dump/load routine.
2BF	2C0	PC	2	Stores the program counter value.
2C1	2C2	RTNADD	2	Stores Return address on execution of B command.
2C3	2C4	LINLST	2	Stores the Buffer address corresponding to the end of the first line of the physical screen.
2C5	2C5	SRNMOD	1	Stores the R option of R command.
2C6	2CF		10	Unused.

17 - 3 -

L

CHAPTER 18 Interfacing with BASIC

Page

18.1 Interfacing with Sequential Access Devices 18-1
18.1.1 DCB (Device Control Blocks) 18-1
18.1.2 DCB table 18-2
18.1.3 Error processing 18-3
18.1.4 BREAK key processing 18-3
18.2 Loading from Expansion Devices 18-4
18.3 ABORT Processing 18-5
18.4 RAM Management 18-6
18.4.1 Application files 18-6
18.4.2 RAM map 18-7
18.4.3 Data configuration 18-8
18.4.4 Configuration of BASIC application files 18-10
18.5 Initializing Extended BASIC 18-11
18.5.1 Expansion method 18-11
18.5.2 Expanded ROM format 18-11
18.5.3 Expansion on RAM base 18-12
18.5.3.1 Loading extended BASIC 18-12
18.5.3.2 Program for reserving extended BASIC area 18-12
18.5.3.3 Configuration of extended BASIC object file 18-14
18.5.3.4 Rewriting warm start and initialize hooks 18-15
18.5.4 Extended BASIC work area 18-16
18.6 System Variables and Hook Table 18-17
18.6.1 System variables 18-17
18.6.2 Hook table 18-18
18.6.3 Entry Point Table 18-18

18.1 Interfacing with Sequential Access Devices

18.1.1 DCB (Device Control Blocks)

To perform I/O operations with sequential access devices such as cassette tapes, etc., a DCB is necessary to specify the conditions for interfacing. DCBs are required for each type of sequential access device ("CAS0:", "COM0:", etc.). The contents of the DCBs are shown below.

Item	Data No. (Size)	Description
1	0 - 3 (4 bytes)	Device name (ASCII code). The four-character device name specified in the file descriptor is entered here.
2	4 (1 byte)	I/O mode. Specified as one of the following values. 10_{16}: Sequential input 20_{16}: Sequential output 30_{16}: Sequential input/output
3	5 - 6 (2 bytes)	Entry point for the OPEN routine. The mode of the file (10_{16}: input, 20_{16}: output) is stored in variable FILMOD (address 068A). The OPEN routine references the mode data and opens the file for input or output.
4	7 - 8 (2 bytes)	Entry point for the CLOSE routine. The CLOSE routine also references variable FILMOD and performs close for input or output.
5	9 - 10 (2 bytes)	Entry point for the input routine for one byte. The input routine inputs one byte is then stored in accumulator A. When the end of the file is detected, FF is entered in variable EOFFLG (address 00F8).
6	11 - 12 (2 bytes)	Entry point for the output routine for one byte. This routine outputs the contents of accumulator A.
7	13 - 14 (2 bytes)	Entry point for EOF routine. This routine sets data FF in accumulator B if the EOF is detected during input. Otherwise, 00 is entered in accumulator B.
8	15 - 16 (2 bytes)	Entry point for LOF routine. This routine enters the number of characters in the buffer or the remaining characters in the file in register D (accumulators A, B).
9	17 - 20	Reserved for data unique to each device.
10	21 (1 byte)	Specifies the column position of the next character to be output (leftmost column is taken to be column '0'). This value is returned when the POS function is called. Normally, this value is initialized to 0 and incremented by 1 each time one byte is output by the output routine. Reset to 0 by CR (code 0D) or LF (code 0A). When this value exceeds the range for the length of one line, and the next character is not CR or LF, the output routine for one byte automatically generates CR or LF and resets the column position to 0.

Item	Data No. (Size)	Description
11	22 (1 byte)	Maximum value of characters per line. May be specified in the range 00 to FF. 00 indicates that the number of characters per line is infinite. As a result, BASIC does not automatically output CR/LF. 00 is set by executing WIDTH (device name), 255.
12	23 (1 byte)	Specifies the size of the print zone when items in a PRINT statement are delimited by "," (comma). The default value is 14.
13	24 (1 byte)	Column position of last print zone. This value is according to the maximum number of characters in the line and the size of the print zone. For example, when the maximum number of characters in the line is 80 and the size of the print zone is 14, this value will be 56.
14	25 (1 byte)	If the number of characters per line can be changed by the WIDTH statement, 00 is entered as the value of this item. Otherwise, 80_{16} is entered.

18.1.2 DCB table

This is a 32-byte table which stores the addresses of the DCBs for each device. Addresses are specified in two bytes and up to 16 DCB addresses can be stored in this table. In the current version, seven addresses are stored in the DCB table and space for nine more addresses is reserved. Device numbers ($0-15_{10}$) are assigned to the DCBs in sequence. The variable name for the DCB table is DCBTAB (address 0657).

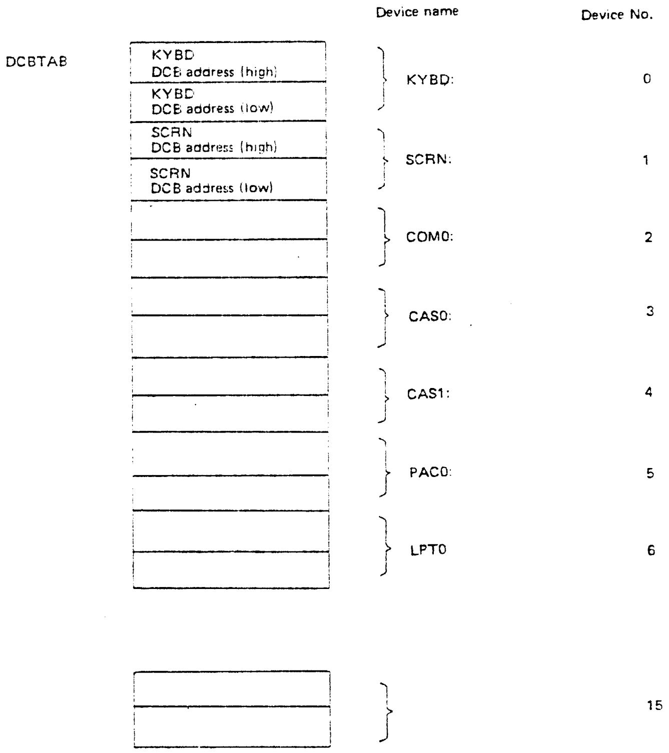


Fig. 18-1 DCB Table

18.1.3 Error processing

When an I/O error occurs during the execution of a routine or when the required device is busy, the corresponding error code is set in accumulator B and the following procedure is executed.

```
ERROR          EQU          $8433
               LDAB         #XX      ; SET THE ERROR CODE.
               JMP          ERROR    ; JUMP TO THE ERROR HANDLER.
```

The following error codes are commonly used.

Error code	Message	Description
53_{10}	IO	Error in communication with a peripheral device.
59_{10}	IU	Specified device is in use (busy).
$6Ø_{10}$	DU	Device is unavailable.

18.1.4 BREAK key processing

The following two procedures are available when BREAK signal is detected during execution of an I/O operation with a peripheral device.

(1) Processing BREAK as an error

In this case, processing is identical to that for an I/O error. Error code 53 (I/O error) is set in accumulator B and control is transferred to the error handler subroutine (label name ERROR).

```
LDAB     #53    ; ERROR CODE FOR 1/Ø ERROR.
JMP      ERROR
```

This procedure does not effect the other open devices or variables. When an ON ERROR GOTO statement has not been executed in the program mode, or when the I/O error occurs in the direct mode, the following error message will be displayed.

```
I/O ERROR (IN XXXX)
```

If an ON ERROR GOTO statement has been executed in the program mode, control is transferred to the specified error trap routine.

(2) Abort processing

Control jumps to label name ABTDO (address $A9Ø8_{16}$). The BASIC interpreter clears all variables, closes all files and initializes all I/O devices. Then, the following message is displayed.

```
ABORT (IN XXXX)
```

18.2 Loading from Expansion Devices

The BASIC interpreter inhibits load from any device other than "CAS0:, "CAS1:", "PAC0:" and "COM0". Loading from any device other than these will result in an FC error. However, load from expansion devices can be enabled by rewriting the hook on the RAM (normally set to jump to the FC error routine). The RAM hook is 3 bytes long and has a format: JMP XXX.

Write the entry point address of the program enabling loading from the expanded device into the address portion of the hook. For load processing, when control is returned from the OPEN routine, variable ASCFLG (one byte, address 068C) is checked, and if ASCFLG is 00, binary format load is performed.

The following two routes are used by the OPEN routine to set the value of variable ASCFLG.

(1) FF is set in variable ASCFLG when the A option is specified in the SAVE statement and 00 is set when the A option is not specified. This data is written to the file header during program save and set in variable ASCFLG by the OPEN routine during load processing.
(2) If the A option is specified in the SAVE statement, a value other than FF is written as the first character of the file. If the A option is not specified, FF is written as the above character. Therefore, the value of ASCFLG can be set by reading of the first character of the file using the OPEN routine.

Hook name
HKLOAD

Address
05E2

Parameters
(A): Device number

Processing sequence

```
  HKLOAD     EQU     $05E2
  FCERR      EQU     $8C70
  LODCNT     EQU     $A6D0
              :
             LDD     #LOADC
             STD     HKLOAD+1
              :
  LOADCK     CMPA    XX         * check the device number
             BEQ     LOADOK
             JMP     FCERR      * GIVE 'FC Error'
  LOADOK     JMP     LODCNT     * CONTINUE LOADING
              :
```

18.3 ABORT Processing

If an I/O operation is aborted by pressing the BREAK key, the BASIC interpreter initializes all devices and closes all files (communications channels). When one of the devices in the DCB table has been expanded, these devices will also have to be initizalized if I/O to another device is aborted. This initialization is also performed using a hook.

Hook name
HKABTD

Address
Ø63C

Note:
Normally, 39_{16} (RTS command) is stored at address Ø63C.

18.4 RAM Management

18.4.1 Application files

Application programs (BASIC interpreter, word processor, etc.) can use the RAM to store the data required by their systems as application files.

Application files are protected against use and accidental destruction by other application programs. Required data can be stored in these files in the same manner as data for BASIC programs can be stored in RAM files.

(1) Before execution of an application program (Fig. 18-2)
All application files are stored in the upper addresses of the RAM.

(2) During execution of an application program (Fig. 18-3)
The application program reserves a work area for itself by moving the application files stored at addresses lower than its own to addresses lower down in the free area. However, the location of this work area varies according to the status of the other application files. Therefore, if a fixed work area is required, the area immediately following the system area is reserved for this purpose. To secure work areas for execution, each application program expands its application files into the fixed and variable work areas.

(3) Upon termination of application program execution
Upon termination of execution of an application program (power switch is turned OFF, RESET switch is pressed or normal completion), control returns to the Menu leaving the RAM allocation as it was during the execution of the application program.
Then, when the same or another application program is selected from the Menu, the menu program calls the file reform routine for the files of the previously executed application program.
The file reform routine selects only the required data from the fixed and variable work areas to create an application file and returns the RAM to the status in (1) above. Control is then transferred to the application program selected from the menu.
For application programs which do not require application files, the free area is used as work area as shown in Fig. 18-2.
In this case, the file reform routine is not called.

18.4.2 RAM map

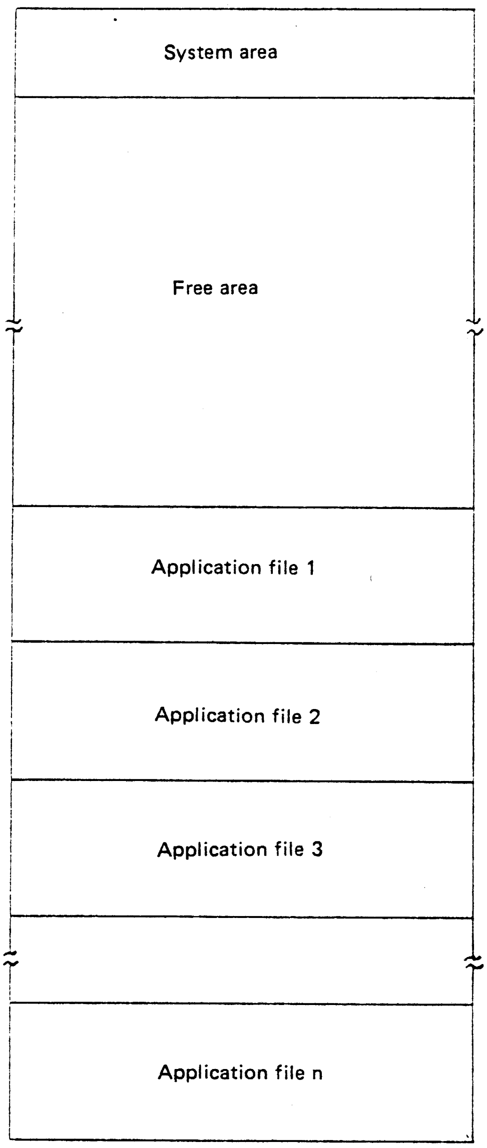
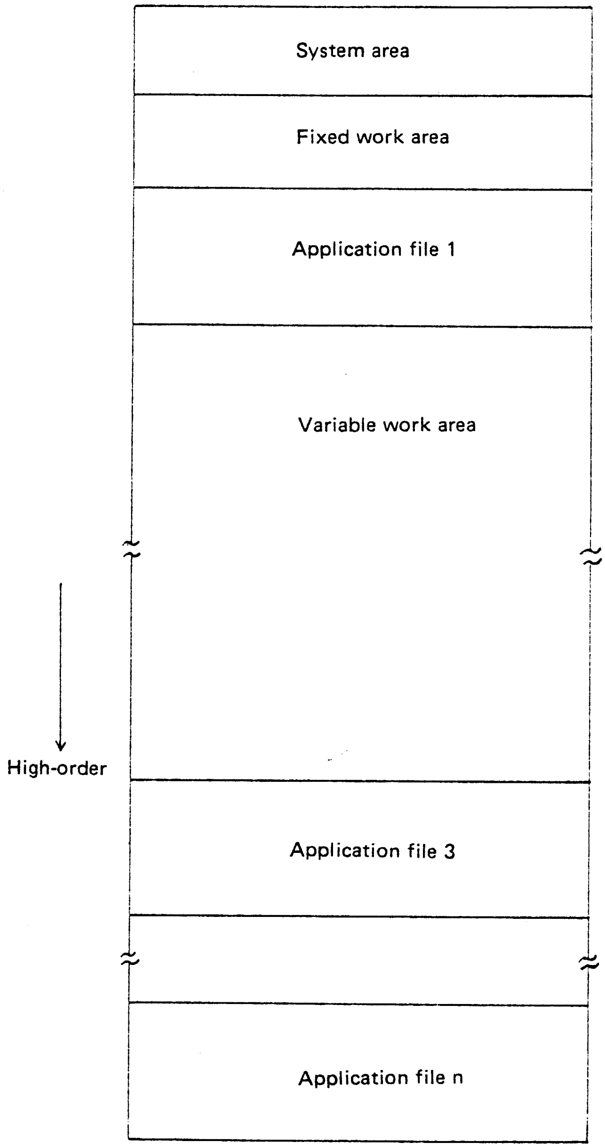


Fig. 18-2 RAM map (1)



When application file 2 is used

Fig. 18-3 RAM map (2)

18.4.3 Data configuration

BASTAB

Indicates the beginning of the application file. When the system is initialized, the address set here is the same as that indicated by RMLTAD.

RMLTAD

Indicates the last address in the RAM +1. The value of RMLTAD is set when the RAM is checked during system initialization.

CNDADR

Indicates the entry point of the file reform routine. The address of the file reform routine for the application program is set in this variable when the application program is executed and the application files are expanded.

INITAB

INITAB bit 6 is set (logic '1') to indicate that the files must be reformed before the next application program can be executed. This flag is set when the value of CNDADR is set.
When this flag is set, the Menu program calls the subroutine whose address is stored in CNDADR (file reform routine for the previously executed application program) before transferring control to the application program selected from the menu. This flag is reset within the subroutine after the application files are reformed. When application files are not used, this flag must not be set.

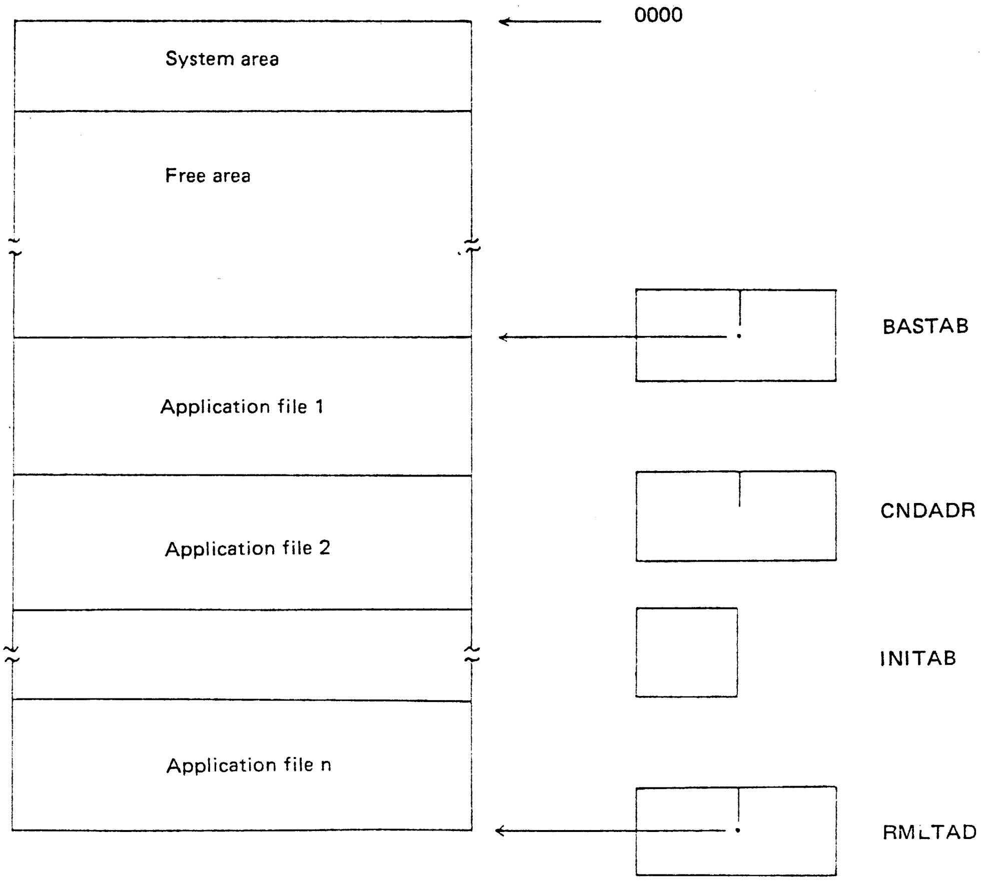


Fig. 18-4 Pointers Used for Application Files

Fig. 18-5 shows an example of when two application files exist simultaneously. The beginning of application file 1 is indicated by BASTAB while the end of application file 2 is indicated by RMLTAD.

(1) File size

The file size is shown by the first two bytes of the file in higher- and lower-order byte sequence. The starting address of the next application file can be obtained by adding the file size to the beginning address of the current file.

(2) Application ID

Application programs are assigned unique one-byte values which are used as IDs. These application IDs are used by application programs when searching for their files.

(3) Data

The data length is the file size -3 bytes. Data format is optional. Unique formats may be used for individual application programs.

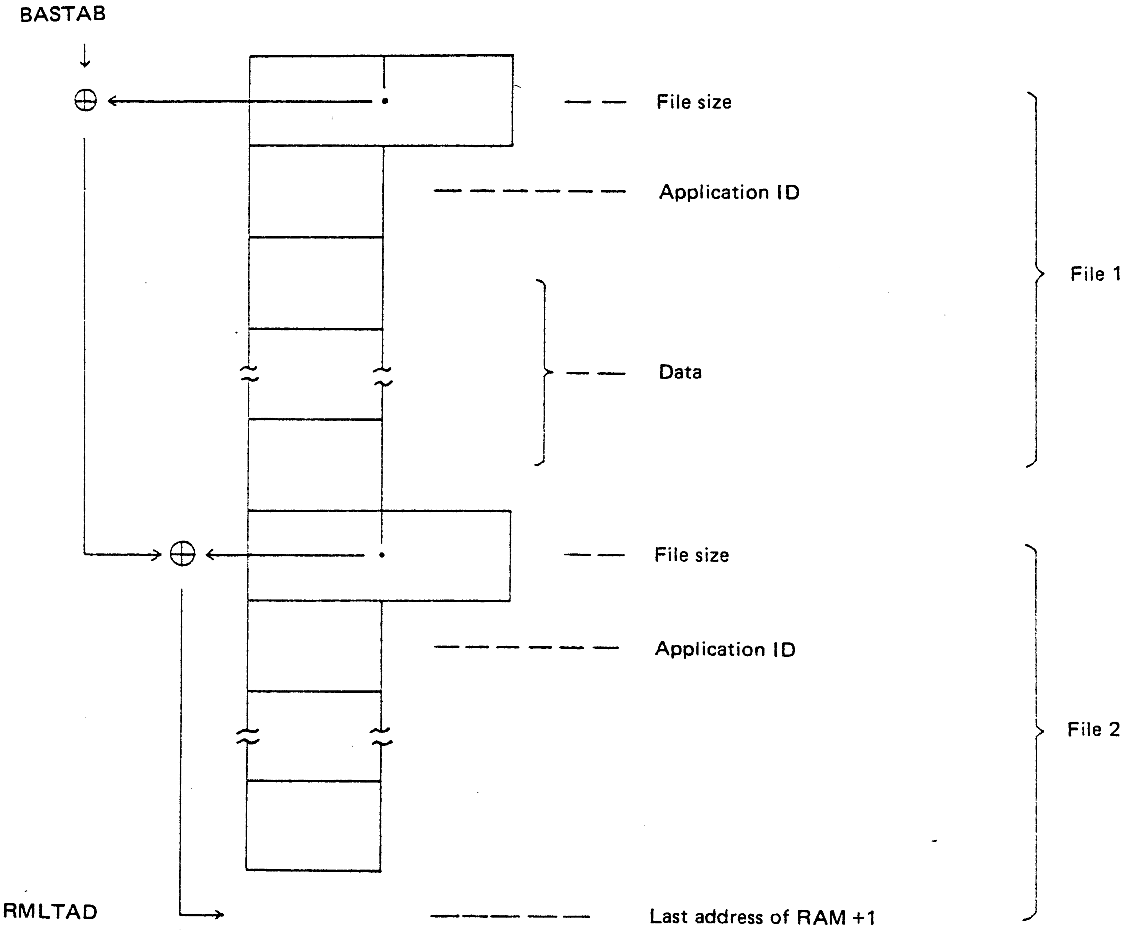


Fig. 18-5 Use of Pointers for Two Application Files

18.4.4 Configuration of BASIC application files

BASIC application files must be stored at the end of the application file area.

(1) Application ID

BASIC : 80_{16}

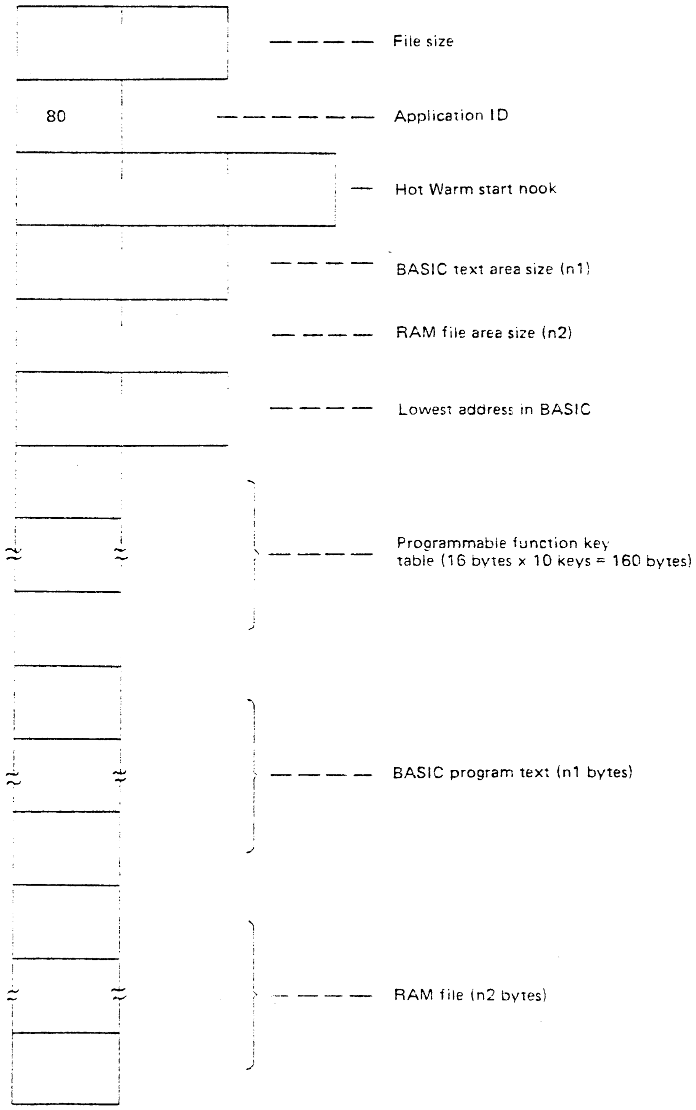
(2) Warm start hook

The one- to three-byte machine language command stored in this hook is executed to execute BASIC warm start. 39_{16} (RTS command) is set here when the system is initialized.

When the expanded BASIC code is stored in the RAM, a JMP command (C3XXXX) is set in this hook to transfer control to the initialize routine for expanded BASIC.

(3) Lowest address used by BASIC

The address specified in the MEMSET statement is set.



File size
80
Application ID
Hot Warm start nook
BASIC text area size (n1)
RAM file area size (n2)
Lowest address in BASIC
Programmable function key table (16 bytes x 10 keys = 160 bytes)
BASIC program text (n1 bytes)
RAM file (n2 bytes)

Fig. 18-6 Application File

18.5 Initializing Extended BASIC

18.5.1 Expansion method

When executing warm start, the BASIC interpreter copies the DCBs and the DCB tables from the ROM and initialize the hooks and pointers. To expand BASIC, these hooks and DCBs must be changed after warm start has been executed. Three methods of expanding BASIC (ROM base, RAM base and Disk base) are available.

After initialization has been completed, the BASIC interpreter executes BASIC expansion in the following sequence. The DCBs and hooks are rewritten by the initialize routines in ROM or RAM or by the DISK boot program.

(1) Check executed for whether the expansion ROM has been set in the memory bank in which the BASIC interpreter is currently located. Control is transferred to (3) below, if the expansion ROM is not stored in this memory bank.

(2) The initialize routine for the expansion ROM is executed.

(3) Check executed for whether the floppy disk unit is available for serial communications. If the disk unit is not connected, control is transferred to (5) below.

(4) The boot program is loaded from the floppy disk unit and then executed.

(5) Warm start hook is executed. (If RAM-base expansion is to be executed, a JMP command to transfer control to the initialize routine is set in this hook.)

18.5.2 Expanded ROM format

Format for expanding BASIC on a ROM base is shown below.

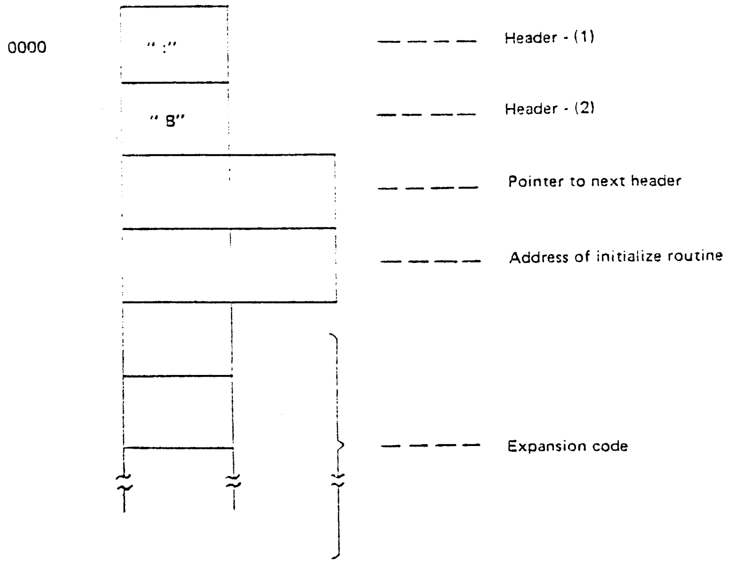


Fig. 18-7 Expanded ROM Format

Notes:

(1) The expanded ROM for extended BASIC must be located in address 6000_{16}.

(2) Other application programs may be stored in the same ROM with extended BASIC. However, the header of extended BASIC must be located at the starting address of the ROM.

18.5.3 Expansion on RAM base

18.5.3.1 Loading extended BASIC

The memory area for extended BASIC is reserved by creating a special application file at the end of the other application files. The procedure for loading extended BASIC is described below.

(1) The BASIC interpreter is executed after initialization (CTRL/@).

(2) Load extended BASIC and the program to reserve the necessary memory area into the machine language area (LOADM command).

(3) Execute the program for reserving the memory area.
This program renews BASTAB and RMLTAD and reserves a RAM area sufficient to store extended BASIC. It then moves extended BASIC from the machine language area to these files. Also, the warm start hooks, etc., in the BASIC application file are rewritten and the initialize routine for extended BASIC is attached at the end of the initialize routine chain which starts from the warm start hook.

(4) Transfer control to the BASIC interpreter warm start routine.

The above sequence makes extended BASIC resident in the RAM. Thereafter, when warm start is executed, the initialize routine in extended BASIC rewrites the DCBs and hooks to expand BASIC.

As the area reserved for extended BASIC is at the end of the application files area, it remains unaffected even if the application files are used by other application programs.

The extended BASIC codes must be assembled to enable their use at the destination addresses. However, these addresses of course vary with the current RAM capacity. In order to enable use of the codes irrespective of the RAM capacity, extended BASIC must be relocated after it is moved to the RAM.

18.5.3.2 Program for reserving extended BASIC area

The procedure for reserving the necessary memory area for extended BASIC is described below.

(1) When control is transferred to the program for reserving memory area, the BASIC interpreter is already running and the BASIC application files are already extended. The file reform routine is therefore called to store only the necessary data in the application files. (Fig. 18-8)

```
LDX     CNDADR
JSR     , X
AIM     #$BF, INITAB
```

(2) Next, the BASIC application files are moved forward (BASTAB → RMLTAD-1) to reserve the area for extended BASIC. (BASTAB is also updated).
Simultaneously, (RMLTAD) is also updated and set at the head of the extended BASIC area to protect extended BASIC. (Fig. 18-9)

(3) Extended BASIC, loaded simultaneously with the memory reserve program, is then moved to the newly reserved application files.

(4) A jump command to transfer control to the initialize routine for extended BASIC is set in the warm start hook in the BASIC application file (currently, RTS command) or in the initialize hook for extended BASIC already existing in the RAM.

(5) Control jumps to the BASIC interpreter warm start entry point.

```
        LDX     $8004
        JMP     0, X
```

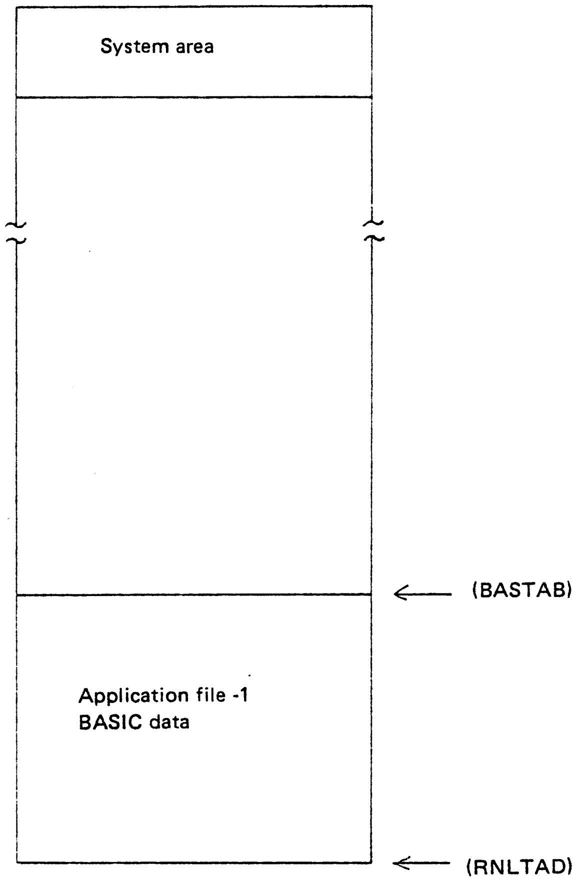


Fig. 18-8 Status before Reserving Memory Area

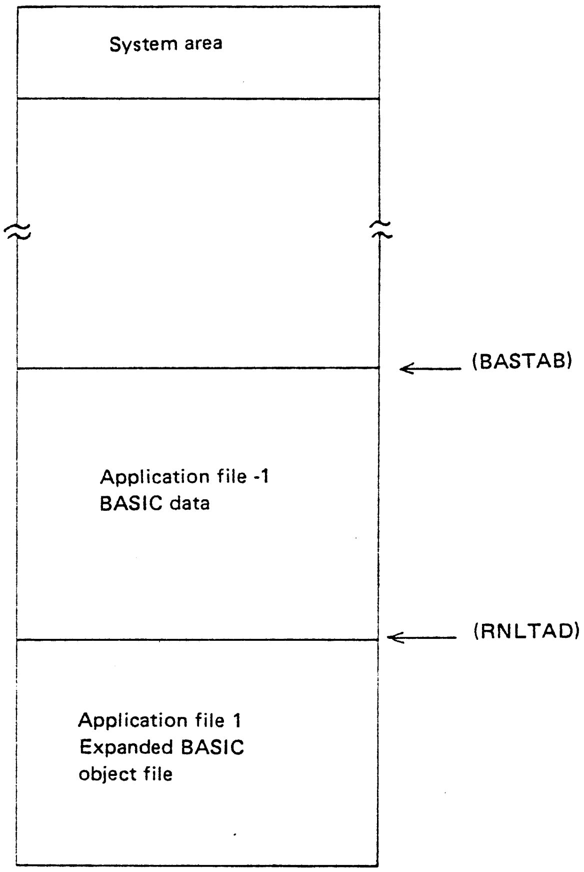
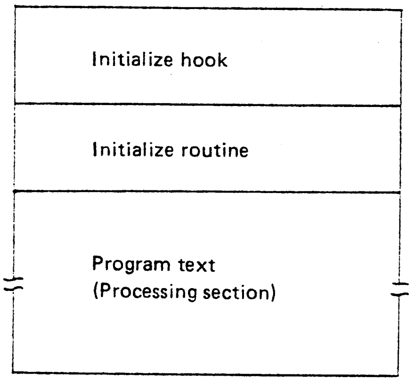


Fig. 18-9 Status after Reserving Memory Area

18.5.3.3 Configuration of extended BASIC object file

The configuration of the extended BASIC object file is shown below.



(1) Initialize hooks

The initialize hook consists of the 3 bytes shown below. When multiple extended BASICs reside in the RAM, this hook is used to link the different initialize routines.
The initial value of the hook is RTS (39_{16})

39	00 00

(2) Initialize routine

The initialize routine starts from the next address following the initialize hook. Each time BASIC is warm started, this routine rewrites the hooks, adds DCBs, etc.
When the initialize routine is entered, the pointer to the sign-on message is stored in register (X). This is either the current BASIC sign-on message or else the sign-on message set by the previous initialize routine for extended BASIC. The pointer to the sign-on message must be set in register (X) when the initialize routine is existed. To display a sign-on message for extended BASIC, set the pointer for the sign-on message in register (X) on exit from the initialize routine for extended BASIC. The sign-on message will then be output when control is returned to the BASIC interpreter or when control is transferred to the next initialize routine for extended BASIC. If the set message is to be output when the initialize routine is entered, STROUT should be called on entry.
If above sign-on message is not to be output, the value of register (X) should be retained so that this register can be returned to its initial value on exit from the initialize routine. In this case, the normal message or the message set by the previous initialize routine will be output.

(3) Chaining initialize routines

When multiple extended BASICs are to be expanded on the RAM, the initialize routines for all of these BASICs must be executed at warm start. First, as the warm start hook has been rewritten to transfer control to the first BASIC initialize routine, this routine is executed.

Upon completion of execution of the initialize routine, control jumps to the initialize hook. At this stage, if the initialize hook is still set to its initial value, the RTS command will be executed and control returned to the BASIC interpreter. If the initialize hook has been rewritten to jump to another BASIC initialize routine, that routine will be executed next. Initialize routines can in this way be chained and executed in succession until the RTS command is encountered.

18.5.3.4 Rewriting warm start and initialize hooks

The procedure for adding the initialize routine for an extended BASIC, newly loaded in the RAM, to the end of the execution chain starting from the warm start hook is described below.

(1) The warm start hook in the BASIC application file is checked. If the value of the warm start hook has not been rewritten (that is, if it is still RTS), it is rewritten to jump to the initialize routine for extended BASIC.
If the warm start hook has already been rewritten (if a jump command has been set), operation proceeds to (2) below.

(2) The extended BASIC initialize hook at the jump destination of the warm start hook is checked. If it has not been rewritten, it is rewritten to jump to the initialize routine for the newly loaded extended BASIC.
If the initialize hook has already been rewritten (if a jump command has been set), control is returned to (2). This operation is repeated until an initialize hook in which RTS has not been rewritten is encountered.

18.5.4 Extended BASIC work area

The following RAM area is used as the work area for extended BASIC irrespective of whether BASIC has been expanded on the ROM or on the RAM.

```
    0A38 -- 0A3D      6 bytes
```

For RAM base expansion, if the work area is insufficient, a work area in the application files is reserved along with the area required for loading extended BASIC. For ROM base expansion, a RAM area is reserved with the application files as in RAM base. This area is then used as the work area. (Subroutines are set in the ROM and executed manually (EXEC command) after system initialize.)
The same procedure is followed to retain data in extended BASIC.

18.6 System Variables and Hook Table

18.6.1 System variables

(1) INITAB (address 0078_{16}, 1 byte)

Bits 0 to 5 and bit 7 are initialize request flags. One bit is assigned for each application. The flag is set (logic "1") to indicate that initialization has been executed. It is reset at system initialize.

The bit of this variable corresponding to the application program to be executed is checked prior to execution if the program requires initializaiton for its files, etc. If the flag is reset (logic "0"), initialize processing is performed to reserve the necessary work areas, etc., for the files and execution of the program is performed only after the INITAB flag becomes "1". If the flag is set (logic "1"), this means that the application program has already been initialized. It can therefore be executed immediately. INITAB flags are not assigned to application programs which do not require initialization.

Bits currently used are as follows.

Bit 0 ----- Menu program
Bit 7 ----- BASIC interpreter

Bit 6 is a file reform request flag. For application programs which require their files be expanded, the pointer to the file reform routine must be set in variable CNDADR and bit 6 of INITAB must also be set after file expansion has completed.

The file reform routine is called by the menu program and resets bit 6 of INITAB after reforming the files.

(2) RMLTAD (address $012C_{16}$, 2 bytes)

This is the pointer for the last address in the RAM +1. This variable is set at system initialize. Also functions as the pointer for the last address of the application files +1.

(3) BASTAB (address 0134_{16}, 2 bytes)

Pointer to the starting address of the application files. Set to the same address as RMLTAD at system initialize.

(4) CNDADR (address 0136_{16}, 2 bytes)

Pointer to the file reform routine. Set by the application program. Valid only if INITAB bit 6 is also set.

(5) DCTAB (address 0657_{16})

DCB table

(6) DEVNUM (address $063E_{16}$)

Enables LOAD from expansion devices.

(7) ASCFLG (address 068C, 2 bytes)

Specifies mode (ASCII or binary) for load. Set by the device OPEN routine.

The BASIC interpreter interprets the flag status as follows:
FF: ASCII load
0: Binary load

(8) OPTBUF (address 068F)
The character string in the file descriptor used to specify options is set in this buffer. The option routine uses this data. The file descriptor option statement is set in this buffer in its original form. It is not placed in brackets. (00) is used as the end mark. If (00) is entered as the first character, option is assumed not to have been specified.

18.6.2 Hook table

(1) HKLOAD (address $05E2_{16}$)
Enables LOAD from expansion devices.

(2) HKABTD (address $063C_{16}$)
Used to initialize expansion devices in case of ABORT.

18.6.3 Entry Point Table

	Label name	Address
(1)	ERROR	8433
(2)	ABTDO	A9D8
(3)	FCERR	8C70
(4)	LODCNT	A6D0

18 - 19

L

APPENDIXES

1. LCD CONTROLLER/DRIVER μPD7227C

2. HD146818RTC (Real Time Clock)

3. HD6301V MCU (Microcomputer Unit)

4. MNEMONIC CODES

5. INSTRUCTION CODES

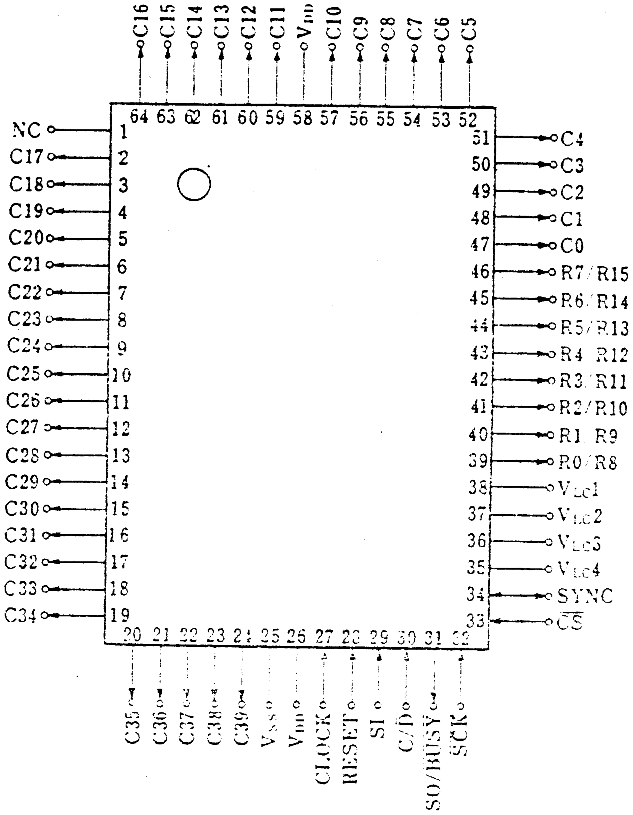
1. LCD CONTROLLER/DRIVER µPD7227G

The µPD7227G, an LCD (Liquid Crystal Display) controller/driver programmable by software, interfaces with the CPU in the microcomputer application system and can directly drive the 8 or 16 time-sharing dot matrix LCD. It also incorporates a character generator that generates specific patterns.

FEATURES

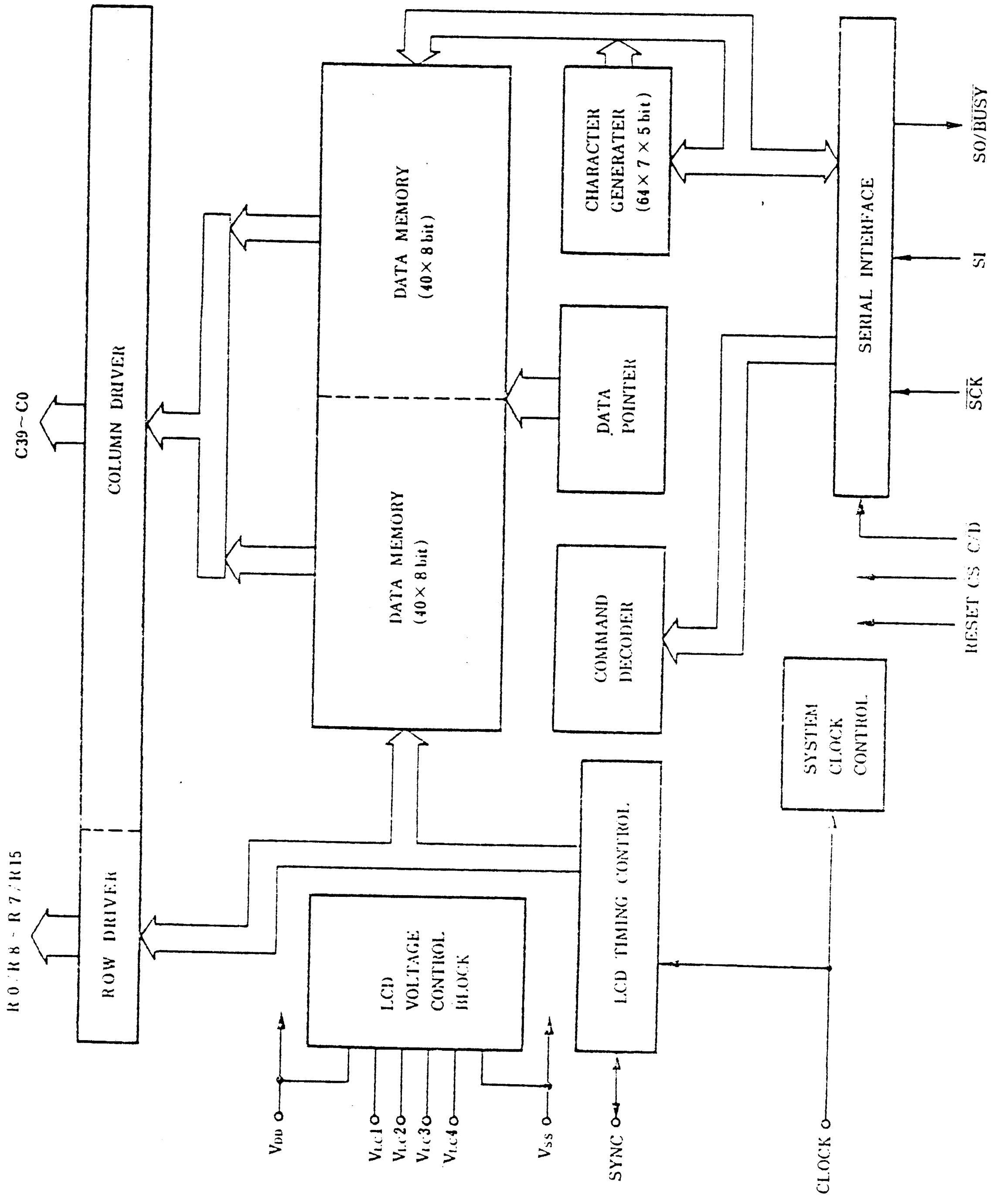
- ° Direct LCD drive
- ° Time-sharing for drive (Note 1)
  8 time-sharing (single/multi-chip structure)
  16 time-sharing (2-chip structure or above)
- ° Expandable multi-chip display digits
- ° Expandable multi-chip display digits
- ° 5 x 7 character generator which can generate 36 alphanumeric characters and 28 symbols
- ° 8-bit serial I/O compatible with µPD7500 series, µCOM-43n series, and µCOM-87 serial interface
- ° 12 types of controller commands
- ° CMOS
- ° Single power supply
- ° 64-pin plastic flat package

Terminal assignment diagram
(Top view)



Note 1: The HX-20 is a 16 time-sharing type.

BLOCK DIAGRAM



FUNCTIONS OF TERMINALS

(1) SI (Serial Input) terminal: Input
This terminal inputs serial commands and data from the μPD7227.

(2) SO/$\overline{\text{BUSY}}$ terminal: Oatput
This terminal functions not only as the serial data output terminal (SO function) but also as the $\overline{\text{BUSY}}$ signal that permits and inhibits the serial data transfer.
The SO and $\overline{\text{BUSY}}$ functions are switched according to the state of the C/$\overline{\text{D}}$ terminal only in the read mode set by an SRM command.

The $\overline{\text{BUSY}}$ function is always used in modes other than Read mode.
The SO/$\overline{\text{BUSY}}$ terminal enters a high-impedance state when the $\overline{\text{CS}}$ terminal becomes High level.

The SO function is selected in read mode when the C/$\overline{\text{D}}$ terminal is Low level and is used to output the contents of data memory.

The $\overline{\text{BUSY}}$ function is used when the CPU checks whether or not the μPD7227 can input and output serial data.

(3) $\overline{\text{SCK}}$ terminal: Input
This terminal inputs the serial clock that shifts the contents of the serial register and latches a signal from the SI terminal, and inputs and outputs serial data in synchronization with the input serial clock.

The serial clock input when the $\overline{\text{CS}}$ terminal is High level is ignored.

(4) C/$\overline{\text{D}}$ terminal: Input
This terminal specifies the serial command and data input from the SI terminal. When the C/$\overline{\text{D}}$ terminal become High level, the $\overline{\text{BUSY}}$ function of the SO/$\overline{\text{BUSY}}$ terminal is selected. When the terminal becomes Low level, the SO function is selected.

Operating mode	C/$\overline{\text{D}}$	SO/$\overline{\text{BUSY}}$	SI
Write, AND, OR, and Character modes	0	$\overline{\text{BUSY}}$	Data input
	1	$\overline{\text{BUSY}}$	Command input
Read mode	0	SO	Invalid
	1	$\overline{\text{BUSY}}$	Command input

(5) SYNC (Synchronous) terminal: I/O
This terminal performs the wire-OR operation to increase the number of multi-chip display digits.
When multi-chips are configured (a row/drive signal is used together), one of the chips is selected as a master chip and the SYNC terminal of that selected master chip is used as output mode.
The SYNC terminal of the slave chip is used as input mode. An SMM (Set Multiplexing Mode) command is used to specify I/O.
The master chip outputs a SYNC signal per frame. Other slave chips fetch the output SYNC signals to the LCD timing controller to tune the alternating drive signal and the frame cycle to the master chip.

(6) $\overline{\text{CS}}$ (Chip Select) terminal: Input
This is a low active chip select input terminal. When this terminal becomes Low level, the serial data transmission/reception between the μPD7227 and a microcomputer, etc., is enabled.

(7) C0~C39 column terminals: Output
These output terminals function as the column drive signals.

(8) R0/R7-R8/R15 row terminals: Output
These output terminals function as the ROW0-ROW7 or the ROW8-ROW15 row drive signals. The SMM command is used to select these signals.

(9) V_{LC1}-V_{LC4} (LCD Drive Voltage) terminals: Input
These are the LCD drive voltage input terminals that generate the LCD drive signals (row/column drive signals).

(10) RESET terminal: Input
This is a high active reset input terminal.

(11) CLOCK terminal: Input
This terminal directly inputs an external clock. When multi-chips are configured, a clock with the same frequency and phase must be supplied to the CLOCK terminal of each μPD7227.

(12) V_{DD}
This is a positive power supply terminal for the system.

(13) V_{SS}
This is a ground (GND) potential terminal for the system.

μPD7227 Commands

Both the C/$\overline{D}$ terminal and the $\overline{BUSY}$ signal must become High level before μPD7227 commands are input. The μPD7227 commands are stored in the serial register via the SI terminal in synchronization with the serial clock to be input from the $\overline{SCK}$ terminal. The command decoder decodes the commands in the serial register and then performs processing according to the decoded commands.

(1) LCD display mode commands

i) SFF (Set Frame Frequency)

0	0	0	1	0	F_2	F_1	F_0

Sets the frame frequencies. The frame frequencies are obtained by demultiplying the clocks input from the CLOCK terminal by the demultiply rates specified by bits F2 through FØ of the SFF command.

F2	F1	FØ	Frame frequency	
Ø	Ø	Ø	$fcl/2^{14}$	← HX-2Ø
Ø	Ø	1	$fcl/2^{13}$	
Ø	1	Ø	$fcl/2^{12}$	
Ø	1	1	$fcl/2^{11}$	
1	Ø	Ø	$fcl/2^{1Ø}$	

NOTE: fcl indicates the clock frequency.

ii) SMM (Set Multiplexing Mode)

0	0	0	1	1	M_2	M_1	M_0

Specifies the time-sharing count, the I/O operation of the SYNC terminal, and data memory banks (effective only for the 8 time-sharing specification). This command is also used to select the row driver functions.

M_2	M_1	$M_Ø$	Time-sharing count	Selection of driver functions	SYNC terminal	Data memory bank spec.	
Ø	Ø	Ø	8	ROWØ-ROW7	Input	Bank Ø	
Ø	Ø	1		ROWØ-ROW7	Input	Bank 1	
Ø	1	Ø		ROWØ-ROW7	Output	Bank Ø	
Ø	1	1		ROWØ-ROW7	Output	Bank 1	
1	Ø	Ø	16	ROW8-ROW15	Input	Banks Ø and 1	
1	Ø	1		ROW8-ROW15	Input	Banks Ø and 1	
1	1	Ø		ROWØ-ROW7	Output	Banks Ø and 1	← HX-2Ø
1	1	1		ROWØ-ROW7	Output	Banks Ø and 1	

iii) DISP OFF (Display Off)

0	0	0	0	1	0	0	0

Erases the LCD display indicating the relationship between the row and column signals regardless of the displayed data.

iv) DISP ON (Display On)

0	0	0	0	1	0	0	1

Displays data according to the displayed data.

(2) Mode setting commands for data processing
Five mode setting commands are available: SRM (Set Read Mode), SWM (Set Write Mode), SANDM (Set AND Mode), SORM (Set OR Mode), and SCM (Set Character Mode). These commands are used to set the modes to control the I/O operations and specify the update operation of a data pointer according to the states of bits I_1 and I_0 of each command. Note that the update operation is performed only when data is input/output.

I_1	I_0	Data pointer operation
0	0	The pointer is incremented by 1 each time data is input/output.
0	1	The pointer is decremented by 1 each time data is input/output.
1	0	Inhibited
1	1	The same address is retained even after data I/O operation.

For example, if I_1, I_0=00 is specified in the SWM command, the data pointer is incremented by 1 each time data is written. After execution of the SWM command, data is sequentially written into the data memory indicated by the data pointer.
In this case, the set mode is retained until other I/O command is executed.

i) SRM (Set Read Mode)

0	1	1	0	0	0	I_1	I_0

Sets read mode, specifies to update the data pointer according to the state of bits I_1 and I_0 of this command each time data is read, and loads the contents of the memory indicated by the current data pointer to the serial register.

ii) SWM (Set Write Mode)

0	1	1	0	0	1	I_1	I_0

Sets write mode and specifies to update the data pointer according to the specification of I_1 and I_0 of this command each time data is written. In this mode, the serial data in the serial register is directly written into the data memory by the μPP7227.

iii) SANDM (Set AND mode)

0	1	1	0	1	1	I_1	I_0

Sets AND mode and specifies to update the data pointer according to the specification of I_1 and I_0 of this command each time data is written. In this mode, the serial data in the serial register is ANDed with the contents of the data memory addressed by the data pointer and then the result is stored in the data memory by the µPD7227.

iv) SORM (Set OR Mode)

0	1	1	0	1	0	I_1	I_0

Sets OR mode and specifies to update the date pointer according to the specification of I_1 and I_0 of this command each time data is written.
In this mode, the serial data in the serial register is ORed with the contents of the data memory addressed by the data pointer and then the result is stored in the data memory by the µPD7227.

v) SCM (Set Character Mode)

0	1	1	1	0	0	1	0

Sets character mode. In this mode, the serial data in the serial register is written into the data memory via the character generator. In this mode, the data pointer is incremented by 5 each time one character (5 x 7 bits) is written into the data memory.

(3) Memory bit manipulation commands
i) BSET (Bit Set)

0	1	0	B_2	B_1	B_0	J_1	J_0

Sets the bits specified by B_0 through B_2 of the data memory addressed by the data pointer and updates the data pointer according to the specification of codes J_1 and J_0 of this command.
The data pointer can be updated according to the specification of J_1 and J_0 only when the BSET or the BRESET command is executed. Control returns to one of the SRM, SWM, SANDM, and SORM commands after the data pointer is updated by the BSET or the BRESET command.
For example, suppose that data is read from address 0 to address 15 after I_1, I_0=00 (auto increment) is specified in the SRM command. In this case, the data pointer indicates address 16 because it is incremented by 1 after data is read from address 15. If the BSET command where J_1, J_0=01 (auto decrement) is specified is executed, the specified bit of address 16 is set and the data pointer indicates address 15 because it is decremented by 1. If data is read from address 15 again, the data pointer indicates address 16 because it is incremented by 1.

J_1	J_0	Data pointer operation
0	0	The data pointer is incremented by 1.
0	1	The data pointer is decremented by 1.
1	0	Inhibited
1	1	The same address is retained.

ii) BRESET (Bit Reset)

0	0	1	B_2	B_1	B_0	J_1	J_0

Resets the bits specified by B_0 through B_2 of the data memory addressed by the data pointer and updates the data pointer according to the specification of codes J_1 and J_0 of this command after the reset.
J_1 and J_0 of the BRESET command have the same meaning as those of the BSET command.

(4) Data pointer specification command

LDPI (Load Data Pointer with Immediate Data)

1	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Sets immediate data D_0 through D_6 in the data pointer.

DATA I/O

(1) Data input

i) Write mode

Write mode is set by the SWM command. Serial data must be input after checking that the $\overline{\text{BUSY}}$ signal output from the SO/$\overline{\text{BUSY}}$ terminal becomes High level. 8-bit serial data beginning with an MSB (Most Significant Bit) is first supplied to the SI terminal, concurrently with the serial clock supplied to the $\overline{\text{SCK}}$ terminal. The µPD7227 then stores the serial data input from the SI terminal in the serial register in synchronization with the serial clock and directly writes it in the data memory addressed by the data pointer when the eighth serial clock rises. The µPD7227 updates the data pointer according to the specification of codes I_1 and I_0. Once 8-bit serial data is stored in the serial register, the µPD7227 enters the busy state and outputs the $\overline{\text{BUSY}}$ signal (SO/$\overline{\text{BUSY}}$=Low) from the SO/$\overline{\text{BUSY}}$ terminal. $\overline{\text{BUSY}}$ remains effective while the serial data in the serial register is internally being processed. No new serial data can be input during this period. In other words, new serial data can be input after $\overline{\text{BUSY}}$ is released (SO/$\overline{\text{BUSY}}$=High).

ii) AND mode

AND mode means the same as Write mode except that the serial data in the serial register is ANDed with the contents of the data memory addressed by the data pointer and then the result is stored in the data memory by the µPD7227.

iii) OR mode

OR mode means the same as Write mode except that the serial data in the serial register is ORed with the contents of the data memory addressed by the data pointer and then the result is stored in the data memory by the µPD7227.

iv) Character mode

In Character mode, the serial data (5 x 7 bits) in the serial register is decoded by the character generator and is stored in the data memory. In this mode, the data pointer is incremented by 5 each time one decoded character is written into the data memory. The I/O operations of serial data in this mode are the same as those in Write mode.

(2) Data output

i) Read mode

The C/$\overline{\text{D}}$ terminal and the $\overline{\text{BUSY}}$ output from the SO/$\overline{\text{BUSY}}$ terminal must be High level before serial data is read from the serial register. If $\overline{\text{BUSY}}$ is already High level, the C/$\overline{\text{D}}$ is set to Low level and the SO/$\overline{\text{BUSY}}$ to the SO function to initiate the serial data output. 8-bit serial data beginning with an MSB is first supplied to the SO/$\overline{\text{BUSY}}$ (SO function) in synchronization with the 8-clock serial clock. The µPD7227 then loads the contents of the data memory addressed by the data pointer to the serial register when the eighth serial clock rises. The µPD7227 enters the $\overline{\text{BUSY}}$ state after the load. To read serial data from the serial register, the C/$\overline{\text{D}}$ and the $\overline{\text{BUSY}}$ of the SO/BUSY must be set to High level again.

ii) Command input during Read mode

When an SWM, SANDM, SORM, or SCM command is input, the current mode changes to that specified by the input command. If another command is input, the contents of the data memory addressed by the data pointer are stored in the serial register by the µPD7227 after that command is processed. In this case, the µPD7227 updates the data pointer according to the specification of I_1 and I_0 of the SRM command. If the BSET or the BRESET command is input, the contents of the serial register addressed by the data pointer updated according to the specification of J_1 and J_0 after execution of the input command are stored in the serial register by the µPD7227. Always note the data pointer operation because the µPD7227 updates the data pointer according to the specification of I_1 and I_0 or J_1 and J_0 after serial data is written or read. For example, if the contents of address n are stored in the serial register by the µPD7227 after I_1, I_0=0,0 is specified in the SRM command, the data pointer indicates address n+1. If the DISP OFF command is input, the current mode changes to read mode and the contents of address n+1 are stored in the serial register by the uPD7227. Note that the contents of address n is not output from the SO/BUSY. (Since DISP OFF command is input with C/D=High level, the SO/BUSY becomes BUSY.)

Internal Clock Function

(1) Serial interface

The serial interface, which consists of an 8-bit serial register and a 3-bit SCK counter, inputs μPD7227 commands, and inputs and outputs data in units of 8 bits.

i) Serial data input

Serial data is stored in the serial interface in synchronization with the serial clock to be output to the $\overline{\text{SCK}}$ terminal.

Serial data is input from the SI terminal per bit each time a serial clock rises (the first bit to be input is an MSB) and then stored in the μPD7227 serial register starting from the LSB (bit Ø).

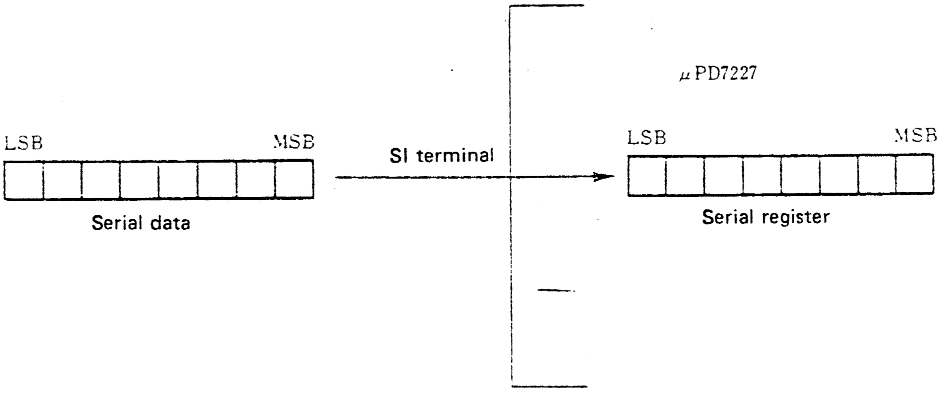


Fig. 3 Structure of Serial Data

The serial data stored in the serial register shifts when the next serial clock falls. The SCK counter counts up when the serial clock rises and issues an overflow indicating that 1-byte (8 bits) serial data has been input when the eighth serial clock rises. In this case, the serial interface stores the contents of the C/$\overline{\text{D}}$ terminal in the serial register in synchronization with the eighth serial clock and checks whether the stored contents are data or commands. If the contents are those of a command (C/$\overline{\text{D}}$=High), the serial interface stores them in the command decoder. If they are data (C/$\overline{\text{D}}$=Low), the interface processes them in Write, AND, OR, or Character mode. When 8-bit serial data (8 serial clocks) is sent, $\overline{\text{BUSY}}$ (SO/$\overline{\text{BUSY}}$=Low) is also output. This $\overline{\text{BUSY}}$ remains effective while the μPD7227 is in process of the previously input serial data. No new serial data can be input during this period. In other words, new serial data can be input any after $\overline{\text{BUSY}}$ is released (SO/$\overline{\text{BUSY}}$=High).

ii) Serial data output

Serial data is output from the serial register when a serial clock is supplied to the $\overline{\text{SCK}}$ terminal as in the case of the data input. Serial data is output from the serial register in turn via the SO/$\overline{\text{BUSY}}$ terminal if serial register shifts when serial clock falls. In this case, the first bit to be output is an MSB.

The SCK counter counts up when a serial clock rises and issues an overflow indicating that one-byte data (8-bit data) has been output when the eighth serial clock rises.

The contents of the data memory addressed by the data pointer are stored in the serial register when the eighth bit rises. These contents are output via the SO/$\overline{\text{BUSY}}$ terminal (C/$\overline{\text{D}}$=Low) when a serial clock is supplied. The input of the $\overline{\text{SCK}}$ is inhibited until next serial data is stored in the serial register after output of 8-bit serial data but $\overline{\text{BUSY}}$ (SO/$\overline{\text{BUSY}}$=Low) is output. Note that the C/$\overline{\text{D}}$ terminal must be set to High and the $\overline{\text{BUSY}}$ of the SO/$\overline{\text{BUSY}}$ must be released (SO/$\overline{\text{BUSY}}$=High) before serial data is output from the serial register.

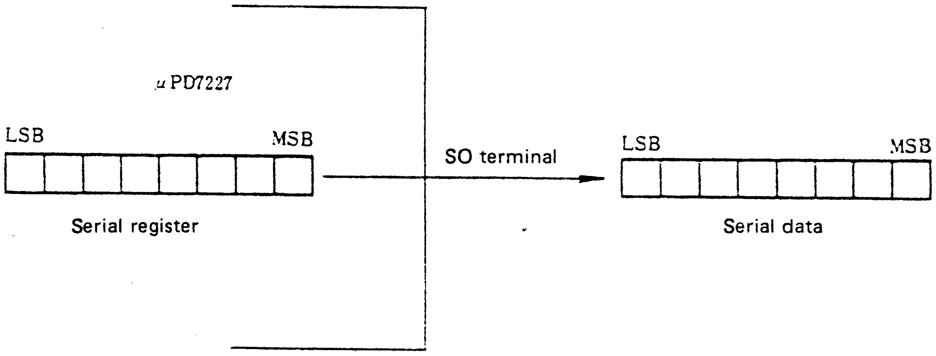


Fig. 4 Structure of Serial Data

(2) Command decoder

The command decoder decodes the commands fetched from the serial register and controls the µPD7227 according to the decoded commands.

(3) Character generator

The character generator, which can be used in Character mode only after execution of the SCM command, generates 36 alphanumeric characters and 28 symbols from the input data (7 x 5 dot matrix). In Character mode, the character generator decodes the serial data sent from the serial register and stores in it in the corresponding data memory (C/$\overline{\text{D}}$=Low). Since decoded data is written into the data memory in 7 x 5 bit format, bit 7 of each data memory address can hold the previous data because it is not affected at all by the write operation.

The data pointer is incremented by 5 each time one character is written into the data memory.

Fig. 5 shows the structure of the 8 time-sharing LCD when the character generator is used. Fig. 6 shows the structure of the 16 timeμsharing LCD.

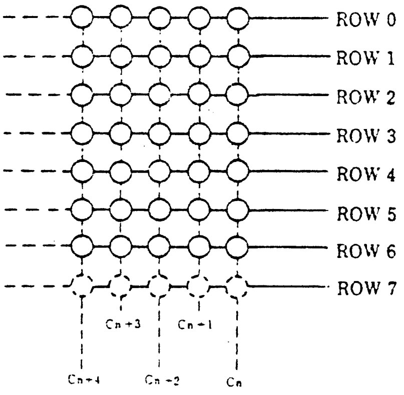


Fig. 5 Structure of 8 time-sharing LCD

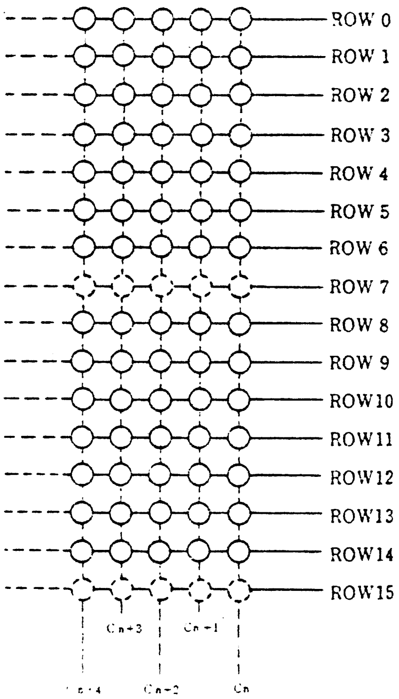
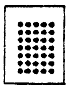


Fig. 6 Structure of 16 time-sharing LCD

NOTE: The portions indicated by can be used as a cursor and an ndicator because they display the contents of bit 7 in each data memory address regardless of the displayed characters.

Fig. 7 shows the correspondence between serial data and display patterns in character mode.
When D6 through D0 are 1111111 or 0000000, all the dots light up.



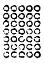
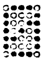
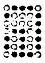
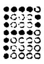
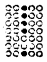
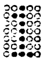
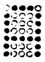
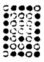
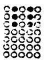
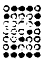
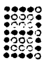
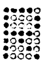
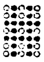
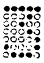
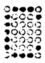
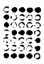
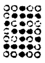
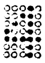
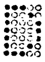
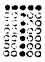
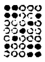
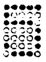
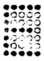
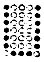
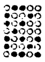
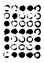
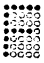
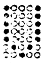
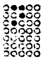
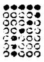
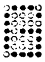
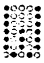
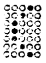
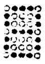
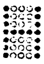
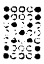
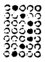
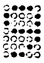
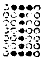
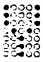
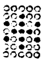
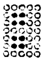
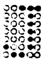
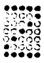
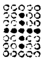
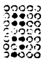
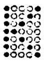
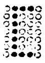
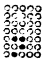
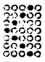
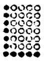
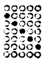
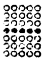
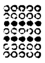
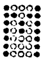
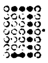
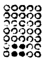
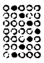
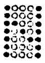
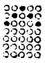
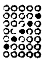
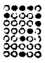
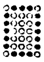
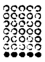
				0	0	1	1	D6
				1	1	0	0	D5
D3	D2	D1	D0	0	1	0	1	D4
0	0	0	0					
0	0	0	1					
0	0	1	0					
0	0	1	1					
0	1	0	0					
0	1	0	1					
0	1	1	0					
0	1	1	1					
1	0	0	0					
1	0	0	1					
1	0	1	0					
1	0	1	1					
1	1	0	0					
1	1	0	1					
1	1	1	0					
1	1	1	1					

Fig. 7 Correspondence between Serial Data and Display Patterns in Character Mode

(4) Data pointer

The data pointer, which consists of a 6-bit binary counter and a 1-bit bank flag, specifies the memory bank and its addresses.

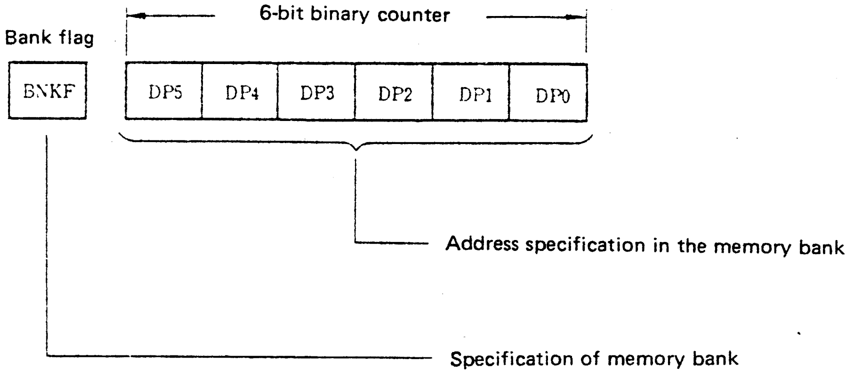

6-bit binary counter
Bank flag
BNKF
DP5
DP4
DP3
DP2
DP1
DP0
Address specification in the memory bank
Specification of memory bank

Fig. 8 Structure of the Data Pointer

The bank flag specifies banks 0 and 1 (40 x 8 bits) and is set/reset by the D6 of an LDPI command. D_6=0 specifies bank 0 and D_6=1 specifies bank 1.
The 6-bit binary counter specifies the bank address and is set by the D_5 through D_0 of the LDPI command. The address specification area of each bank is from address 0 through address 39. Although the 6-bit binary counter can take 40 through 63 (for example, if decrement is performed from 0, 63 is assumed) by setting immediate data or using the automatic increment or decrement function, nothing is performed because there is no data memories corresponding to these values.
The data pointer is updated as follows when data is input and output:

i) Data I/O

In Read, Write, AND, and OR modes, the µPD7227 updates the data pointer according to the specifications of I_1 and I_0 of the commands that set the above modes each time data is input and output.

In Character mode, the data pointer is incremented by 5 each time 8-bit data is input.

ii) Command input or execution

When the BSET or the BRESET command is executed, the µPD7227 updates the data pointer according to the specifications of J_1 and J_0 of the command. After the update, the data pointer returns to the state of I_1 and I_0 before execution of the BSET or BRESET command.

When Read mode is set by the SRM command, the µPD7227 loads the contents of the data memory to the serial register according to the data pointer and updates the data pointer according to the specification of I_1 and I_0 of this command. The data pointer is not updated if commands other than the BSET, BRESET, SRM, and LDPI commands are executed.

(5) Data memory

The µPD7227 incorporates two 4∅ x 8∅ bit data memories each of which consists of banks ∅ and 1. Display data from the character generator (Character mode) and the serial register (Write mode) are written into these memories. They can be used not only to execute the AND (AND mode) and OR (OR mode) operations for the serial register but also to set/reset the specific bits. The specific bits are set when the BSET command is executed and are reset when the BRESET command is executed. In Read mode, the contents of the data memory addressed by the data pointer are loaded to the serial register. As explained in (4), the data pointer specifies banks ∅ and 1.

Since the µPD7227 automatically sends the contents of the data memory to the column driver in bits independently of the I/O operations of the serial register, converts them into appropriate column driver signals, and outputs the signals from the column driver signal output terminal, the data write operation, etc., is not affected at all by the serial data transmission. The relationship between the column drive signals and the row drive signals is determined according to the bit contents read from the data memory. If the bit contents are 1, the corresponding LCD element goes to the selection level and then lights up. If they are ∅, the corresponding LCD element goes to the nonselection element and then goes out.

i) Writing data into a data memory

Serial data is written into the data memory in the following four modes: The bit set/reset operations can also be used when serial data is written into the data memory.

° Write mode

Write mode is set by the SWM command. In this mode, serial register data is directly written into the memory bank addressed by the data pointer.

° Character mode

Character mode is set by the SCM command. In Character mode, serial register data is decoded in 7 x 5 bit format (one character) by the character generator and is written into the data memory addressed by the data pointer. This means that each decoded data is written into the data memory in 5 bytes. Since the MSB (Most Significant Bit) of each address is not used in this mode, the previous data still remains effective even after decoded data is written into the data memory.

° AND mode

AND mode is set by the SANDM command. In this mode, serial register data is ANDed with the contents of the data memory addressed by the data pointer and then the result is stored in the data memory.

° OR mode

OR mode is set by the SORM command. In this mode, serial register data is ORed with the contents of the data memory addressed by the data pointer and then the result is stored in the data memory.

° Bit set operation
Three bits of the data memory addressed by the data pointer are set by specifying B_2 through B_0 of the BSET command. The set bits are released when the command execution terminates and the current mode returns to the mode (Write, Character, AND, OR, or Read mode) before execution of the BSET command.

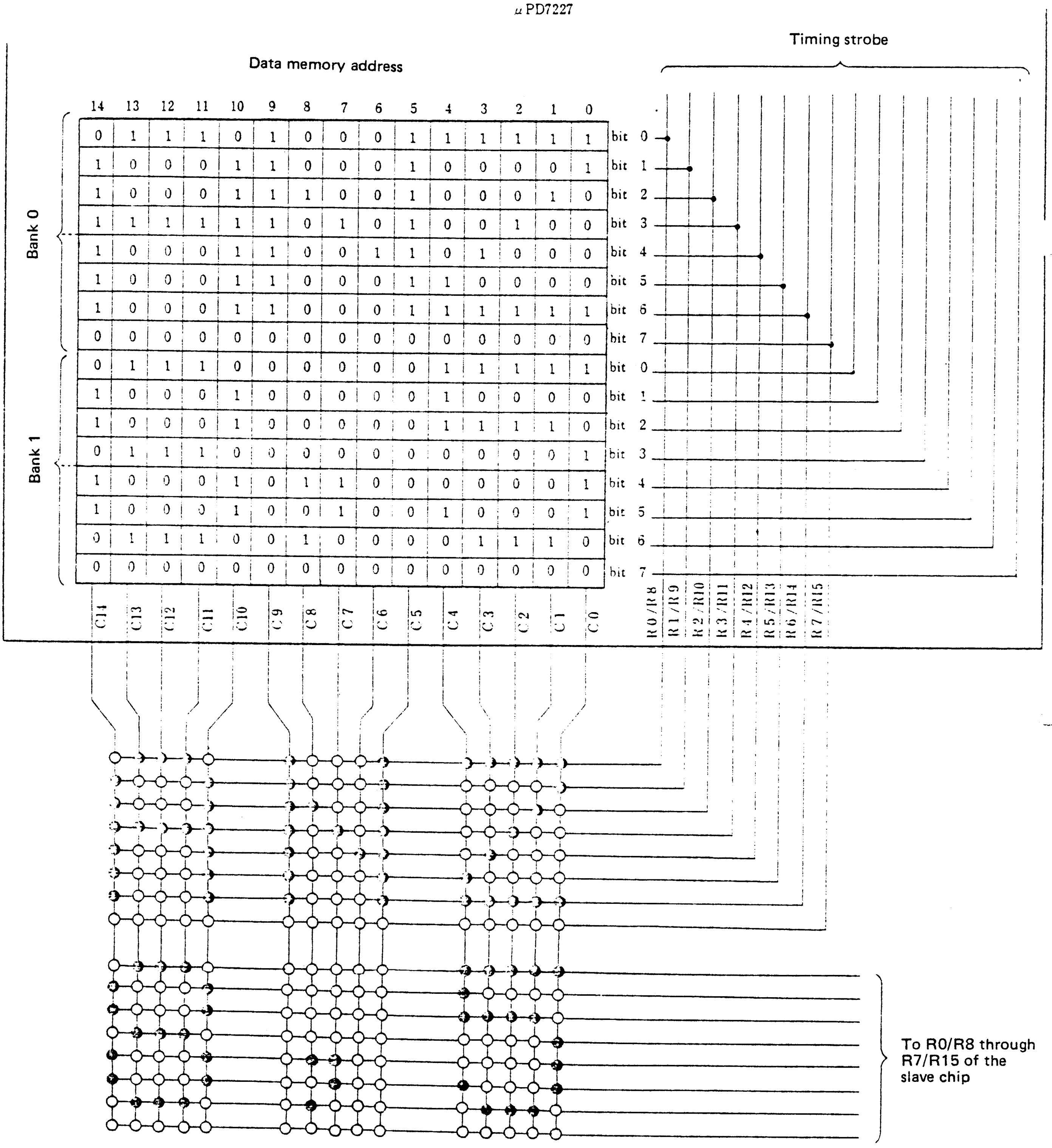
° Bit reset operation
Three bits of the data memory addressed by the data memory are reset by specifying B_2 through B_0 of the BRESET command. The reset command bits are released when the command execution terminates and the current mode returns to the mode (Write, Character, AND, OR, or Read mode) before execution of the BRESET command.

ii) Loading the contents of the data memory to the serial register
The contents of the data memory addressed by the data pointer are loaded to the serial register by specifying the SRM command.

iii) Reading display data from the data memory
Display data to drive the LCD is read from the data memory in bits. The column driver converts the read display data into an appropriate column driver signal and outputs it from the column driver signal output terminal. The read operation of display data depends on the time-sharing count.

° 16 time-sharing

In 16 time-sharing drive, banks 0 and 1 function as 16 x 40 bit data memory. Display data is read in turn from bit 0 of each address in bank 0 in synchronization with the row drive signals. Fig. 10 shows how row drive signals ROW0 through ROW7 and ROW8 through ROW15 are supplied.



	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Bank 0	0	1	1	1	0	1	0	0	0	1	1	1	1	1	1	bit 0
	1	0	0	0	1	1	0	0	0	1	0	0	0	0	1	bit 1
	1	0	0	0	1	1	1	0	0	1	0	0	0	1	0	bit 2
	1	1	1	1	1	1	0	1	0	1	0	0	1	0	0	bit 3
	1	0	0	0	1	1	0	0	1	1	0	1	0	0	0	bit 4
	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	bit 5
	1	0	0	0	1	1	0	0	0	1	1	1	1	1	1	bit 6
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	bit 7
Bank 1	0	1	1	1	0	0	0	0	0	0	1	1	1	1	1	bit 0
	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	bit 1
	1	0	0	0	1	0	0	0	0	0	1	1	1	1	0	bit 2
	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	bit 3
	1	0	0	0	1	0	1	1	0	0	0	0	0	0	1	bit 4
	1	0	0	0	1	0	0	1	0	0	1	0	0	0	1	bit 5
	0	1	1	1	0	0	1	0	0	0	0	1	1	1	0	bit 6
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	bit 7

Fig. 10 16 Time-sharing

(6) Column driver signals CØ through C39

The column driver converts the display data read from address Ø through address 39 of the data memory into appropriate column drive signals and outputs them from the column drive signal output terminal. CØ through C39 are selected by the LCD timing controller and display data based on the LCD drive standard voltage input from the LCD voltage control block. The 1 alternating drive signal of each column drive signal contains one bit display information.

(7) Row driver signals RØ/R7 through R8/R15

The row driver generates row drive signals ROWØ through ROW7 and ROW8 through ROW15. When M2-MØ=Ø-3 is executed by the SMM command, the row driver generates ROWØ through ROW7 at the timing of 8 time-sharing drive input from the LCD timing controller and outputs them from the row drive signal output terminal.
When M2-MØ=6,7 is executed by the SMM command, the row driver generates ROWØ through ROW7 at the timing of 16 time sharing drive input from the LCD timing controller and outputs them from the row drive signal output terminal. When M2-MØ-4,5 is executed by the SMM command, the row driver generators ROW8 through ROW15 at the timing of the 16 time-sharing drive input from the LCD timing controller and outputs them from the row drive signal output terminal. The row driver can only generate either ROWØ through ROW7 or ROW8 through ROW15 at 16 time-sharing. Therefore, to display the LCD at 16 time-sharing, the following must be satisfied:

° Two or more µPD7227s are provided.
  A set of ROWØ through ROW15 is created by outputting ROWØ through ROW7 from the master chip and ROW8 through ROW15 from the slave chip.

(8) LCD timing controller

The LCD timing controller generates an LCD drive timing using the time-sharing count and the demultiply rate, supplies it to the column and row drivers, and at the same time outputs the SYNC signal from the SYNC terminal that synchronizes with the display timing of each µPD7227 when multi-chips are configured. When the SYNC terminal of each µPD7227 is wired OR, the ROW signal can be used, enabling the display digit area to be expanded.

(9) LCD voltage control block

The LCD voltage control block inputs LCD drive standard voltages from $V_{LC}1$ through $V_{LC}4$ and supplies them to the column and row drivers.
The column and row drivers generate the drive voltages for the column and row drive signals from these supplied voltages.

Control Function

(1) Chip select function

When the $\overline{\text{CS}}$ terminal becomes Low level, the μPD7227 is activated, enabling serial data to be input/output.

i) $\overline{\text{CS}}$=1

When the $\overline{\text{CS}}$ terminal is 1 (high), the SI, $\overline{\text{SCK}}$, and C/$\overline{\text{D}}$ terminals becomes High level in the μPD7227 regardless of the input level and the SO/$\overline{\text{BUSY}}$ terminal enters the high-impedance state.

ii) $\overline{\text{CS}}$ rise

When the $\overline{\text{CS}}$ becomes High level (rises) during I/O operation, the SCK counter is cleared, invalidating the transmission/reception of 8-bit data.
When the CS becomes Low level, the transmission/reception of 8-bit data is enabled.

(2) RESET function

When a high 3-clock cycle is input from the RESET terminal, the μPD7227 is reset and then enters the states in i) and ii) below.

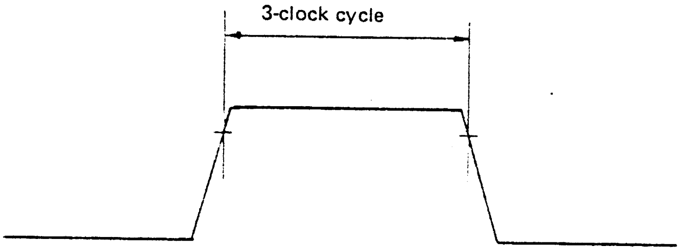


Fig. 13 RESET Waveform

i) Reset state

° When the $\overline{\text{CS}}$ is Low level, the SO/$\overline{\text{BUSY}}$ also becomes Low level. When the $\overline{\text{CS}}$ is High level, the SO/$\overline{\text{BUSY}}$ enters the high impedance state.

° The VLC3 signal is output from both the column drive signal output terminal and the row drive signal output terminal.

ii) State after reset

When the μPD7227 is reset, the SCK counter is cleared and then enters the state when one of the following commands is executed:

SWM(I_1,I_0=00): The data pointer is incremented by 1 in Write mode each time data is input.

LDPI(D_6-D_0=0000000): The data pointer is cleared.

DISP OFF: Both the row drive and column signals go to the nonselection level.

SMM(M_2-M_0-000): The LCD uses an 8 time-sharing drive; the R0/R7 through R8/R15 terminals output ROW0 through ROW7; the SYNC terminal is used for input; the display data memory specifies bank 0.

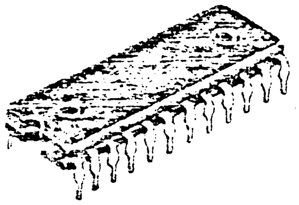
SFF(F_2-F0=000): The frame frequency is set to $f_{cl}/2^{14}$.

2. HD146818RTC (Real Time Clock)

The HD146818RTC is an HMCS6800 microcomputer peripheral LSI with the clock/calendar function that can count year, month, day, day of the week, hour, etc.

It incorporates a 50-byte user RAM and has the timer function that generates square waves, periodic interrupts, etc. The HD146818RTC can be directly connected to MCUs such as HD6801 and HD6301 through the multiplexed bus interface. It can also be directly connected to the 8085-system microprocessor because it incorporates the 8085-system interface circuit. The HD146818RTC is also designed to save power consumption.

HD146818P

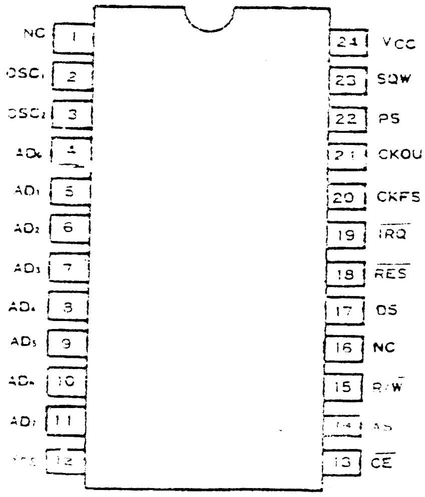


(DP—24)

FEATURES

- Clock/calendar function that can count second, minute, day, day of the week, year, and month.
- 64-byte address spaces
  10 bytes for clock, alarm, and calendar RAMs
  4 bytes for a control register
  50 bytes for a general user RAM
- Three types of interrupts
  Periodic interrupt
  Alarm interrupt
  Update ended interrupt
- Square wave output
- Three types of standard clocks
  4.194304 MHz
  1.048576 MHz
  32.768 KHz
- Binary/BCD display switching of the clock and calendar
- 12- and 24-hour mode switching for hour display
- Built-in automatic leap-year compensation circuit
- Direct connection of the HD146818RTC to the HD6801 and HD6301 via the multiplexed bus

Pin Arrangement



Top View

° Built-in 80-system interface circuit
° Lower CMOS power consumption
° Compatible with the MC146818
  (Motorola Corporation)

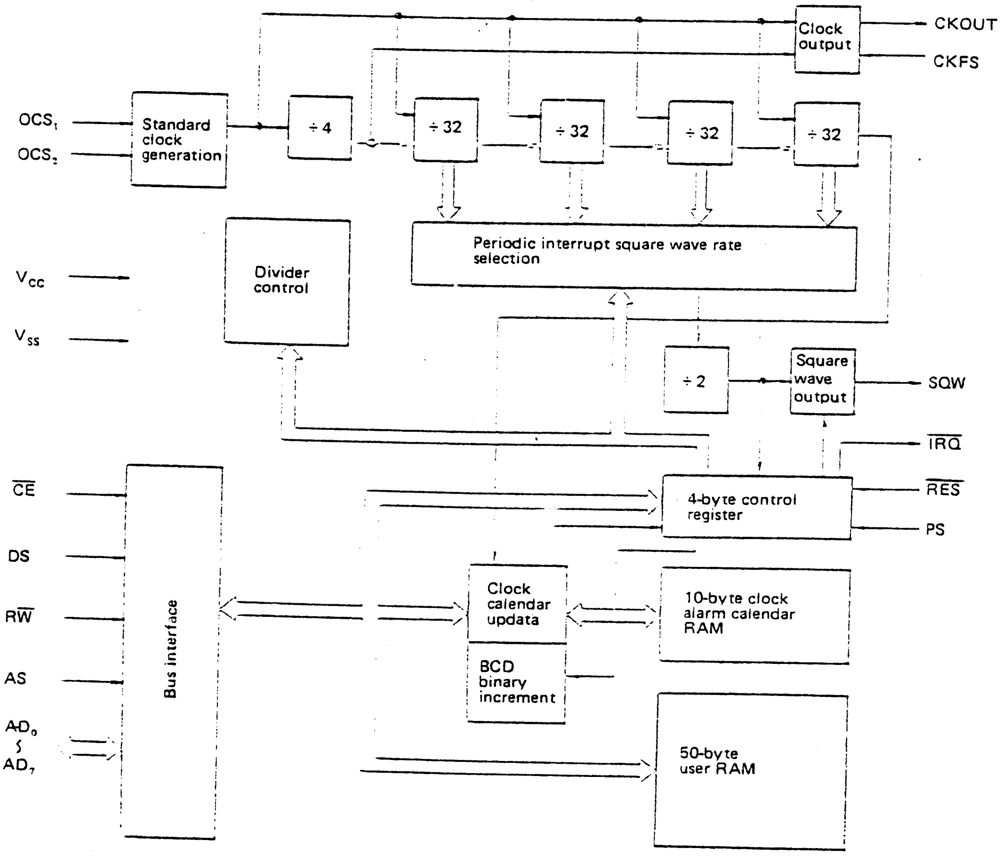


Fig. 1 Internal Block Diagram

LSI FUNCTIONS

Clock Function
The RTC updates the contents of the RAM areas for hour, calendar, and display using the update cycle function once per second and displays the updated year, month, day, day of the week, and hour on the internal RAM.
The processor reads the contents of the internal RAM. Table 2 lists the data displayed on the internal RAM.

Static CMOS RAM (general-purpose 50-byte user RAM)
Necessary data maintained by the system can be stored in this RAM because the RTC is used after battery backup.

Square Wave Generation
The demultiply circuit in the RTC generates a square wave to be supplied to the SQW terminal. The frequency of the generated square wave can be selected by the program. The SQW signal is used as a standard clock signal in the system.

Square Wave Generation

The demultiply circuit in the RTC generates a square wave to be supplied to the SQW terminal. The frequency of the generated square wave can be selected by the program. The SQW signal is used as a standard clock signal in the system.

Three Independent Interrupts

Periodic interrupt

This interrupt can occur once per 30 μs to per 500 ms.

Alarm interrupt

This interrupt can occur only when the time set for this matches that (hour, minute, second) set for an alarm. The interrupt is usually specified so that it can occur at the specified time of a day. When the "11xxxxxx" codes (don't care codes) are written for an alarm, an alarm interrupt occurs per the specified hour, minute, and second.

Example: If the "don't care" codes are written in the RAM for an hour
alarm when 13:25:30 is specified, an alarm interrupt occurs.

Update ended interrupt

This interrupt occurs each time the time update terminates.

HD146818

Update Cycle Function

The HD146818 updates the calendar once per second when the SET bit of register B is 0. The basic update cycle function is that it reads the second, minute, hour, day of the week, day, month, and year displayed on the internal RAM, updates them if necessary, and writes them into the corresponding areas. The HD146818 compares each alarm specification byte and each time during the update cycle for the alarm judgment.

The update time required for 4.19340- and 1.048576-MHz standard clocks is 248 μs while that required for the 32.768 MHz standard clock is 1984 μs. The time, calendar, and alarm bytes cannot be accessed by the program during the update cycle. If the RAM is read before update processing completes, "FF" indicating that update processing is in progress is output. The program may read this "FF" value because it accesses the RAM asynchronously. The RAM can be accessed in the following three methods without the update cycle function:

In the first method, an update ended interrupt is used to access the RAM. Since the date to be used for approx. 999 ms after this interrupt occurs is already decided in this method, no update cycle occurs.

In the second method, the UIP (Update In Progress) bit of register B is used to check whether or not the update cycle is in progress. The UIP bit is set once per second. The update cycle starts 244 μs after the UIP bit is set. Since the contents of the time/calendar do not change for at least 244 μs (Buc) after the UIP bit is cleared if register A is read, the RAM can be accessed during this period. When the UIP bit is 1, the contents of the time/calendar becomes "FF".

In the third method, the periodic interrupt is used to check whether or not the update cycle is in progress.

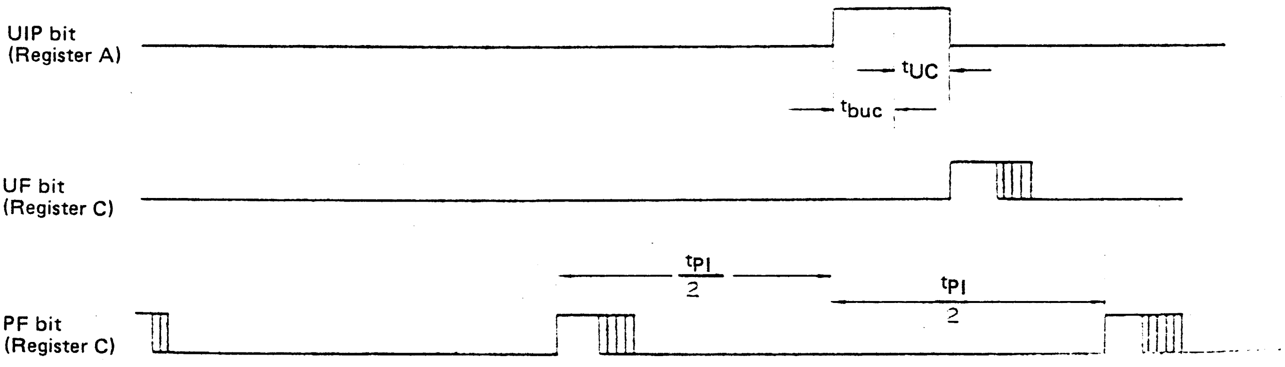
This interrupt is designed to avoid the update cycle when it is issued. In other words, since the update cycle is not performed during (tpI/2)+tBUC if the periodic interrupt cycle is greater than the value obtained by adding tuc to Buc, the contents of the time/calendar do not change.

Table 1 lists the update cycle time.

Table 1 Updata Cycle Time

UIP Bit	Time Base (OSC1)	Update Cycle Time (t_{UC})	Minimum Time Before Update Cycle (t_{BUC})
1	4.194304MHz	248μs	—
1	1.048576MHz	248μs	—
1	32.768kHz	1984μs	—
0	4.194304MHz	—	244μs
0	1.048576MHz	—	244μs
0	32.768kHz	—	244μs

* The update cycle time for the HX-20 is 32.768 kHz.



t_{PI}: Periodic Time Interval (500ms, 250ms, 62.5ms, etc.)

Table 2 Data Format of Time, Calendar and Alarm

Address	Function		Decimal data range	Hexadecimal data range	
				Binary data mode	CD data mode
0	SECONDS		0 to 59	00 to 3B	0 to 59
1	SECONDS ALARM		0 to 59	00 to 3B	0 to 59
2	MINUTES		0 to 59	00 to 3B	0 to 59
3	MINUTES ALARM		0 to 59	00 to 3B	00 to 59
4 (Note 1)	HOURS	12-hour mode	1 to 12	01 to 0C/31 to 8C*	01 to 12/81 to 92C*
		24-hour mode	0 to 23	00 to 17	00 to 23
5 (Note 2)	HOURS ALARM	12-hour mode	1 to 12	01 to 0C/81 to 8C*	01 to 12/81 to 92*
		24-hour	0 to 23	00 to 17	00 to 23
6	DAY OF THE WEEK		1 to 7**	01 to 07	01 to 07
7	DATE OF THE MONTH		1 to 31	01 to 1F	01 to 31
8	MONTH		1 to 12	01 to 0C	01 to 12
9	YEAR		0 to 99***	00 to 63	00 to 99

Note 2

*: The highest-order bit (MSB) is used to distinguish a.m. from p.m.
0: a.m. (AM)
1: p.m. (PM)

**: 1: Sunday
2: Monday
3: Tuesday
4: Wednesday
5; Thursday
6: Friday
7: Saturday

Note 1: The HX-20 uses 24-hour mode.
Note 2: The HX-20 uses BCD data mode.

***:The lower-order 2 digits are set in A.D.

LSI STRUCTURE

Address Map

The HD146818 incorporates the 64-byte memory address consisting of the general-purpose 5-byte RAM, 10-byte RAM for time, day, and alarm data, and four 1-byte registers. The processor can directly read the contents of this memory address.

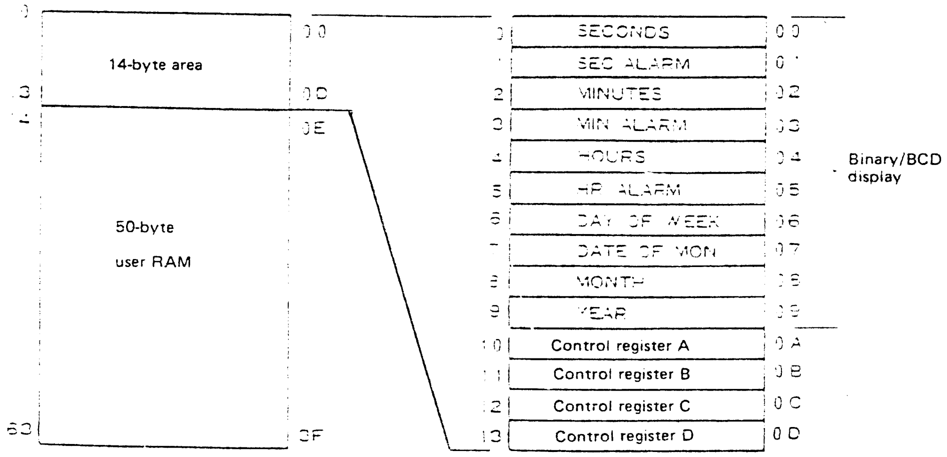


Fig. 2 Address Map

EXPLANATION OF TERMINALS

V_{CC} and V_{SS}
The power supply is connected between these two terminals. The V_{CC} is connected to the plus side of the power supply. The V_{SS} is connected to the ground side.

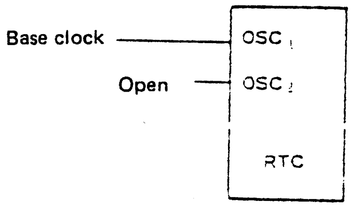
OSC1 and OSC2 (standard clocks): Input
These terminals are used to connect a crystal oscillator or an extererrnal base clock. 4.19430 MHz is used for the crystal oscillator as its frequency. Either 4.19430MHz or 32.768 kHz is used for the external base clock. The bits of control register A are used to specify these frequencies.

Table 3 Base clocks

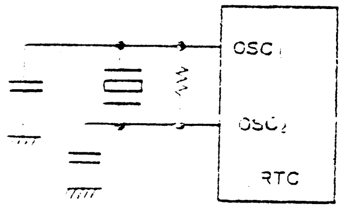
Base clock frequency	CKFS input signal	CKOUT output signal frequency
4.194304 MHz	"High"	4.194304 MHz
	"Low"	1.048578 MHz
1.048576 MHz	"High"	1.04857 MHz
	"Low"	262.144 kHz
32.768 kHz	"High"	32.768 kHz
	"Low"	8.192 kHz

*: These frequencies apply only to the external clocks.

HX-20 clock



(External base clock connection)



(Crystal oscillator connection)

CKOUT (Clock Out): Output
This terminal is a clock output signal created by demultiplying the base clock signal by ÷1 or ÷4. The level of the CKFS input signal is used to specify the demultiply rate. This terminal can also be used as a system base clock (Table 3).

CKFS (Clock Out Frequency Select): Input
This terminal is the input signal that specifies the demultiply rate for the CKOUT output signal. When this terminal is connected to the V_{CC}, the frequency same as that of the OSC1 pin is specified for the CKOUT output.
When this terminal is connected to the V_{SS}, the 4 demultiplied frequency (÷4) is specified (Table 3).

AD_0 through AD_7 (Multiplexed Address/Data Bus)

AD_0 through AD_7 are the bi directional buses that not only send address information used by the processor to access the RTC but also inputs and outputs data. The first half of the cycle is used to send address information while the second half of the cycle is used to send data. The address information to be used must be determined before the AS signal falls. The RTC uses AD0 through AD7 as address information in the first half of the cycle. Data to be used must be determined before the second half of the cycle.
The data bus driver used as the three-state output buffer is always in the high-impedance state unless the RTC outputs data.

AS (Multiplexed Address Strobe): Input

This strobe signal fetches address information from the multiplexed address bus. Address information to be fetched is stored in the RTC when this signal falls.

HD146818

Table 4 Divider Selection Bits

Base clock frequency	Divider selection			CKFS Input	CKOUT Output	Norm. mode	Test mode	Div. reset	No. of dividers bypassed
	DV2	DV1	DV0						
4.194304 MHz	0	0	0	High Low	4.194304 MHz 1.048576 MHz	○	-	-	N=0
1.048576 MHz	0	0	0	High Low	1.048576 MHz 262.144 kHz	○	○	-	N=2
32.768 kHz	0	1	0	High Low	32.768 kHz 8.192 kHz	○	○	-	N=7
--	0	1	1	-	-	○	○	-	N=12
--	1	0	0	-	-	-	○	-	N=17
--	1	0	1	-	-	-	○	-	N=22
--	1	1	0/1	-	-	-	-	○	-

Bits RS3 through RS0 (Rate Selection)

Bits RS3 through RS0 are used to tap the divider circuits to generate square waves and periodic interrupts. These bits can be set only by the program but are not affected by $\overline{\text{RES}}$ at all. They can be read and written.

Table 5 Relationship between the rate selection bits,

periodic interrupts, and SQW

Rate selection bit				4.194304 MHz or 1.048576 MHz		32.768 kHz	
RS3	RS2	RS1	RS0	Periodic interrupt cycle	SWQ output frequency	Periodic interrupt cycle	SWQ output frequency
0	0	0	0	-	-	-	-
0	0	0	1	80.517 μs	32.768 kHz	23.90625 ms	256 Hz
0	0	1	0	61.035 μs	16.384 kHz	7.8125 ms	128 Hz
0	0	1	1	122.070 μs	8.192 kHz	122.070 μs	8.192 kHz
0	1	0	0	244.141 μs	4.096 kHz	244.141 μs	4.096 kHz
0	1	0	1	488.281 μs	2.048 kHz	488.281 μs	2.048 kHz
0	1	1	0	976.562 μs	1.042 kHz	976.562 μs	1.024 kHz
0	1	1	1	1.953125 ms	512 Hz	1.953125 ms	512 Hz
1	0	0	0	3.960625 ms	256 Hz	3.960625 ms	256 Hz
1	0	0	1	7.8125 ms	128 Hz	7.8125 ms	128 Hz
1	0	1	0	15.625 ms	64 Hz	15.625 ms	64 Hz
1	0	1	1	31.25 ms	32 Hz	31.25 ms	32 Hz
1	1	0	0	62.5 ms	16 Hz	62.5 ms	16 Hz
1	1	0	1	125 ms	6 Hz	125 ms	8 Hz
1	1	1	0	250 ms	4 Hz	250 ms	4 Hz
1	1	1	1	500 ms	2 Hz	500 ms	2 Hz

HX-20

DS (Data Strobe): Input

This terminal inputs the Ø2 clock (E: Enable) signal of the HMCS6800 system. This signal becomes High level when the processor reads data from the RTC and becomes Low level when the processor writes data into the RTC. The RD signal is input when the 8085 system is used.

R/$\overline{\text{W}}$ (Read/Write): Input
This terminal inputs the R/$\overline{\text{W}}$ signal of the HMCS6800 system. This signal becomes High level when the processor reads data from the RTC and becomes Low level when the processor writes data into the RTC. The $\overline{\text{WR}}$ signal is input when the 8085 system is used.

$\overline{\text{CE}}$ (Chip Enable): Input
This input terminal (signal) becomes Low level when the LSI is accessed. The level of this signal must be determined during one cycle. The RTC cannot be accessed when the signal is High level.

$\overline{\text{IRO}}$ (Interrupt Request): Open drain output
This active low level output signal requests the processor to issue an interrupt. If the interrupt factor is set in the start bit when the interrupt enable bit is set, this signal becomes Low level.

This signal is released when the processor reads control register C. In other words, this signal is released because the pending interrupt factor is cleared when the $\overline{RES}$ signal becomes Low level. If no interrupt request is issued, the signal enters the high-impedance state. It can be connected to other $\overline{IRQ}$ signal using a pull-up resistor.

$\overline{RES}$ (Reset): Input
This input signal resets the RTC. When this signal becomes Low level, the write/read operations of the internal register and the RAM are inhibited and the following operations are performed:
(1) The Periodic Interrupt Enable (PIE) bit is cleared.
(2) The Alarm Interrupt Enable (AIE) bit is cleared.
(3) The Update Ended Interrupt Enable (UIE) bit is cleared.
(4) The interrupt request status flag is cleared.
(5) The Periodic Interrupt Flag (PF) is cleared.
(6) The Alarm Interrupt Flag (AF) is cleared.
(7) The Update Ended Interrupt Flag (UF) is cleared.
(8) The $\overline{IRQ}$ terminal enters the high impedance state.
(9) The Square Wave Output Enable (SQWE) bit is cleared.
$\overline{RES}$ does not affect the functions of the clock calendar and the RAM. When the power is turned on, the RLH of this signal listed in the Electric Characteristic Table must be Low level.

PS (Power Sense): Input
This input signal is used along with the VRT (Valid RAM and Times) bit of control register 4. When PS becomes Low level, the VRT bit is cleared.

Control Registers
The RTC incorporates four control registers accessible by the program. These registers can be accessed even during update cycle.

Control register A

b7	b6	b5	b4	b3	b2	b1	b0
UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0

Bit UIP (Update in Progress)
Bit UIP is a read-only status flag that can be monitored by the program. When this bit is 1, the update cycle is in progress or is initiated immediately. When the bit is Ø, the update cycle is not in progress but the program can use information on the RAM time, a calendar, and an alarm. This bit is not affected by $\overline{RES}$ at all.

Bits DV2, DV1, and DVØ (Divider Selection)
These bits are used:
(1) To specify the basic frequencies: 4.19 MHz, 1 MHz, and 32 kHz.
(2) To reset and restart the internal 22-tier (row) divider circuit.
(3) To specify the LSI test mode.

The update cycle starts one second after the reset status is released. These bits can be read/written but only the program can set them. They are not affected by RES at all.

Control register B

b7	b6	b5	b4	b3	b2	b1	b0
SET	PIE	AIE	UIE	SQWE	DM	24/12	DSE

Bit SET (Set)
Bit SET is used to interrupt the update cycle when the time and date are set. When this bit is 0, the update cycle is performed once per second. When this bit is 1, the update cycle is interrupted, causing the UIE bit to be cleared. This bit can be set only by the program but is not affected by RES at all.

Bit PIE (Periodic Interrupt Enable)
Bit PIE is used to specify whether or not to output the IRQ signal according to the status of the periodic interrupt flag (PF). When this bit is 1, the periodic interrupt request is output from IRQ. When this bit is 0, no periodic interrupt request is output from IRQ but the PF bit is periodically set. The program can set/reset this bit but RES can only reset it.

Bit AIE (Alarm Interrupt Enable)
Bit AIE is used to specify whether or not to output IRQ according to the status of the alarm interrupt flag (AF). An alarm interrupt occurs when the three bytes of second, minute, and hour correspond to the three alarm bytes. When this bit is 1, an update ended interrupt request is output from IRQ. When this bit is 0, no update ended interrupt request is output but the AF bit is set if the condition is established. The program can set/reset this bit but RES can only reset it.

Bit UIE (Update Ended Interrupt Enable)
Bit UIE is used to specify whether or not to output IRQ according to the status of the update ended interrupt flag (UF). When this bit is 1, an update ended interrupt request is output from IRQ. When this bit is 0, no update ended interrupt request is output from IRQ but the UF bit is set even if the update cycle completes. The program can set/reset this bit but RES can only reset it.

Bit SQWE (Square Wave Enable)
When bit SQWE is set, the square wave with the frequency specified by one of rate selection bits RE3 through RE0 is output from the SQW terminal. When this bit is 0, SQW remains at Low level. This bit is specified only by the program but is not affected by RES at all.

Bit DM (Data Mode)
Bit DM is used to specify the binary or BCD mode in whose mode the time and calendar are displayed.

DM=1: Binary mode
DM=0: BCD mode

Only the program can specify this bit.

Bit 24/12 (24/12 Control)
Bit 24/12 is used to specify the 24- or 12-hour mode.

24/12=1: 24-hour mode
24/12=0: 12-hour mode

Only the program can specify this bit.

Bit DSE (Daylight Saving Enable)
When bit DSE is 1, the following special update operations are performed:

(1) Last Sunday of April	From 1:59:59AM to 3:00:00AM
(2) Last Sunday of October	From 1:59:00AM to 1:00:00AM

When this bit is 0, these operations are not performed. Only the program can specify this bit.

Control register C

b7	b6	b5	b4	b3	b2	b1	b0
IRQF	PF	AF	UF	0	0	0	0

(Read only)

Not used (0 is set when this bit is read.)

Bit IRQF (Interrupt Request Flag)
Bit IRQF is set when the following conditions are satisfied:

PF=PIE=1
AF=AIE=1
UF=UIE=1
IRQF=PF.PIE+AF.AIE+UF.UIE

When this bit is set, $\overline{\text{IRQ}}$ always becomes Low level. This bit is cleared when control register 3 is read or RES becomes Low level.

Bit PF (Periodic Interrupt Flag)
Bit PF is set according to the cycle specified by one of RS3 through RS0. This bit is set independently of the PIPF bit but the $\overline{\text{IRQ}}$ signal and the IRQF bit become Low level or are set only when the PIE bit is 1. This bit is cleared when control register 3 is read.

Bit AF (Alarm Interrupt Flag)
Bit AF is set when the specified time matches the set alarm time. This bit is set independently of the AIE bit but the $\overline{\text{IRQ}}$ signal and the IRQF bit become Low level or are set only when the AIE bit is 1. This bit is cleared when control register 3 is read or $\overline{\text{RES}}$ becomes Low level.

Bit UF (Update Ended Interrupt Flag)
Bit UF is set each time each update cycle completes. This bit is set independently of the UIE bit but the $\overline{\text{IRQ}}$ signal and the IRQF bit become Low level or are set only when the UIE bit is 1. This bit is cleared when control register 3 is read or $\overline{\text{RES}}$ becomes Low level.

Control register D

b7	b6	b5	b4	b3	b2	b1	b0
VRT	0	0	0	0	0	0	0

(Read only)

Not used (0 is set when this bit is read.)

Bit VRT (Valid RAM and Time)
Bit VRT indicates the status of the RAM for general purpose and time display. When this bit is 0, the RAM contents and the time display are not guaranteed because V_{CC} is too low. The processor must be set to provide against the power cut and error after initializing the contents of the RTC. This bit is set when a control register is read.

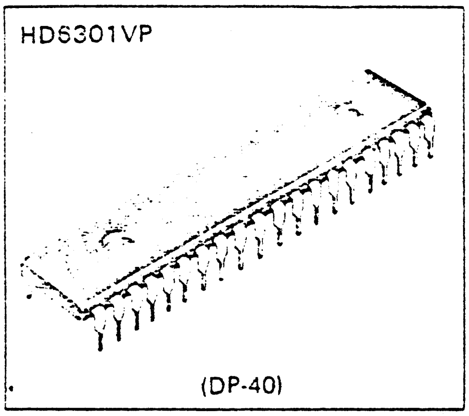
Table 6 RTC Control Register

Address								Reg.	Rd	Wt	Control register bit							
A7	A6	A5	A4	A3	A2	A1	A0	name			b7	b6	b5	b4	b3	b2	b1	b0
x	x	0	0	1	0	1	0	Ctrl reg.A	0	0	UIP*	DV2	DV1	DV0	RS3	RS2	RS1	RS0
x	x	0	0	1	0	1	1	Ctrl reg.B	0	0	SET	PIE	AIE	UIE	SQWE	DM	24/12	DSE
x	x	0	0	1	1	0	0	Ctrl reg.C	0	x	IRQF	PF	AF	UF	0	0	0	0
x	x	0	0	1	1	0	1	Ctrl reg.D	0	x	VRT**	0	0	0	0	0	0	0

* Read only bit
** Set to '1' by read out of control register 4.

3. HD6301V MCU (Microcomputer Unit)

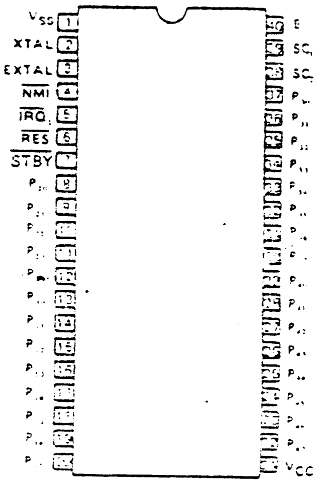
The HD6301V, a single-chip microcomputer unit, provides command object codes compatible with the HD6801, 4K-byte ROM, 128-byte RAM, serial communication interface (SCI), parallel I/O terminals as well as 3 functions of timers on same chips. It is bus compatible with the HMCS6800 and has some additional functions such as an improved execution time plus several new instructions of operation to increase system throughput. The HD6301V can also access the address space of up to 65K words. Like the HMCS6800 family, the I/O level of the HD6301V is compatible with TTL and can operate on +5 V single power supply.



FEATURES

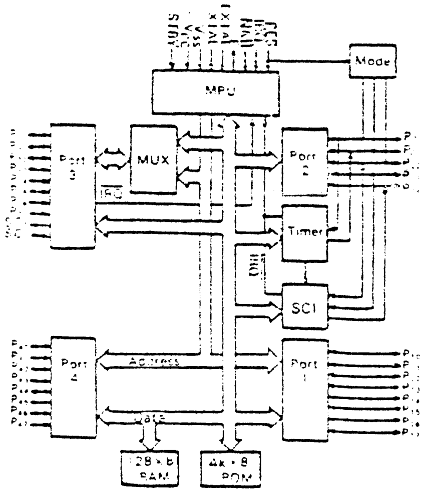
- Extended instruction sets of the HD6801 family
- Abundant incorporated functions compatible with the HD6801VO: 4K-byte ROM, 128-byte RAM, 29 parallel I/O terminals, 2 terminals of data strobe, 16-bit timer, and SCI
- Low power consumption modes: Sleep and standby modes
- Minimum instruction execution time 1 μs (f=1MHz), 0.67 μs (f=1.5MHz), 0.5 μs (2MHz)
- Bit manipulation and bit test instruction
- Error detect function Address trap and op-code trap
- Up to 65K-word address space
- Wide operation range VCC=3 through 7 V (f=0.5 MHz) f=0.1 through 2 MHz (Vcc=5 V ±10%)

Pin Arrangement



Top View

BLOCK DIAGRAM



AC Characteristics

Bus timing (Vcc=5.0 V ± 10%, Vss=0 V and Ta=0 ~ +70°C unless otherwise noted)

Item	Symbol	Test condition	1MHz version			1.5MHz version			2MH version			Unit
			min	typ	max	min	typ	max	min	typ	max	
Cycle time	t_{cyc}	Figs.1,2	1	-	10	0.666	-	10	0.5	-	10	μ
Address strobe pulse width ("High" level)	PW_{ASH}		200	-	-	150	-	-	120	-	-	ns
Address strobe rise time	T_{ASr}		-	-	20	-	-	20	-	-	20	ns
Address strobe fall time	T_{ASFf}		-	-	20	-	-	20	-	-	20	ns
Address strobe delay time	T_{ASD}		60	-	-	40	-	-	20	-	-	ns
Enable rise time	t_{Er}		-	-	20	-	-	20	-	-	20	ns
Enable fall time	t_{Ef}		-	-	20	-	-	20	-	-	20	ns
Enable pulse width ("High" level)	PW_{EH}		450	-	-	330	-	-	220	-	-	ns
Enable pulse width ("Low" level)	PW_{EL}		450	-	-	330	-	-	220	-	-	ns
Delay time address strobe to enable	t_{ASED}		60	-	-	40	-	-	20	-	-	ns
Address delay time	t_{AD}		-	-	250	-	-	170	-	-	120	ns
Address delay time for latch	t_{ADL}		-	-	250	-	-	170	-	-	120	ns
Data set-up time for write	t_{DSW}		225	-	-	150	-	-	120	-	-	ns
Data set-up time for read	t_{DSR}		80	-	-	60	-	-	40	-	-	ns
Data hold time for read	t_{HR}		10	-	-	10	-	-	10	-	-	ns
Data hold time for write	t_{HW}		20	-	-	20	-	-	20	-	-	ns
Address set-up time for latch	t_{ASL}		20	-	-	20	-	-	20	-	-	ns
Address hold time for latch	t_{AHL}		20	-	-	20	-	-	20	-	-	ns
Address hold time	t_{AH}		10	-	-	10	-	-	10	-	-	ns
A0-A7 set-up time for enable rise	t_{ASM}		200	-	-	180	-	-	120	-	-	ns
Peripheral read access time for nonmultiplexed bus	(t_{ACCN})		-	-	600	-	-	400	-	-	300	ns
Peripheral read access time for multiplexed buses	(t_{ACCM})		-	-	600	-	-	400	-	-	300	ns
Oscillator stabilization time (power ON reset time)	(t_{RC})	Figs. 10, 11	20	-	-	20	-	-	20	-	-	ms
Processor control set-up time	(t_{PCS})		200	-	-	200	-	-	200	-	-	ns

Peripheral port timing (Vcc=5.0 V ± 10%, Vss=0 V and Ta=0 to +70°C unless otherwise specified)

Item		Symbol	Test condition	1MHz version			1.5MHz version			2MH version			Unit
				min	typ	max	min	typ	max	min	typ	max	
Peripheral data set-up time	Ports 1, 2, 3 and 4	t_{PDSU}	Fig. 3	200	-	-	200	-	-	200	-	-	ns
Peripheral data hold time	Ports 1, 2, 3 and 4	t_{PDH}	Fig. 3	200	-	-	200	-	-	200	-	-	ns
Delay time (Enable rise to $\overline{OS}$ 3)		t_{OSD1}	Fig. 5	-	-	300	-	-	300	-	-	300	ns
Delay time (Enable rise to $\overline{OS}$ 3)		t_{OSD2}	Fig. 5	-	-	300	-	-	300	-	-	300	ns
Delay time (Enable fall to peripheral data output)	Ports 1, 2*, 3 and 4	t_{PWD}	Fig. 4	-	-	300	-	-	300	-	-	300	ns
Input strobe pulse width		t_{PWIS}	Fig. 6	200	-	-	200	-	-	200	-	-	ns
Input data hold time	Port 3	t_{IH}	Fig. 6	50	-	-	50	-	-	50	-	-	ns
Input data set-up time	Port 3	t_{IS}	Fig. 6	20	-	-	20	-	-	20	-	-	ns

* Except P21

Timer and SCI timing (usual case: Vcc=5.0 V ± 10%, Vss=0 V Usual Case: 0 ~ +70°C)

Item	Symbol	Test condition	1MHz version			1.5MHz version			2MH version			Unit
			min	typ	max	min	typ	max	min	typ	max	
Timer input pulse width	t_{pwt}		2.0	-	-	2.0	-	-	2.0	-	-	t_{cyc}
Delay time (enable rise to timer output)	t_{TOd}	Fig.7	-	-	400	-	-	400	-	-	400	ns
SCI input clock cycle	t_{scyc}		1.0	-	-	1.0	-	-	1.0	-	-	t_{cyc}
SCI input clock pulse width	t_{pwsck}		0.4	-	0.6	0.4	-	0.6	0.4	-	0.6	t_{cyc}

Mode programming (Vcc=5.0 V ± 10%, VSS=0 V Usual case: Ta=0 ~ +70°C)

Item	Symbol	Test condition	1MHz version			1.5MHz version			2MH version			Unit
			min	typ	max	min	typ	max	min	typ	max	
Low level voltage for mode programming input	V_{MPL}	Fig.8	-	-	0.8	-	-	0.8	-	-	0.8	V
High level voltage for mode programming input	V_{MPH}		2.0	-	-	2.0	-	-	2.0	-	-	V
RES pulse width (low)	PW_{RSTL}		3	-	-	3	-	-	3	-	-	t_{cyc}
Mode programming set-up time	t_{MPS}		2	-	-	2	-	-	2	-	-	t_{cyc}
Mode programming hold time	t_{HMP}		150	-	-	150	-	-	150	-	-	ns

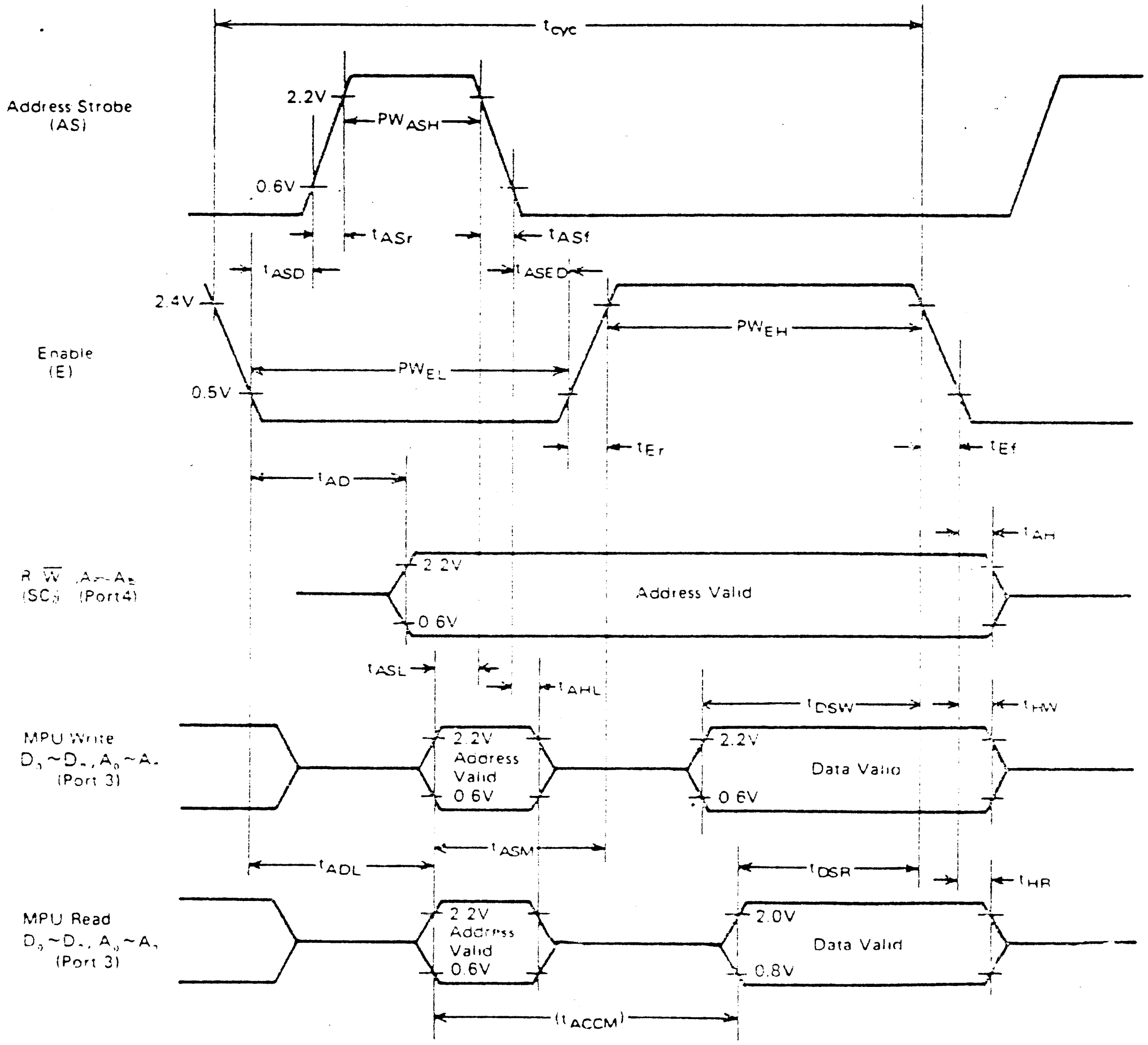


Fig. 1 Expanded Multiplexed Bus Timing

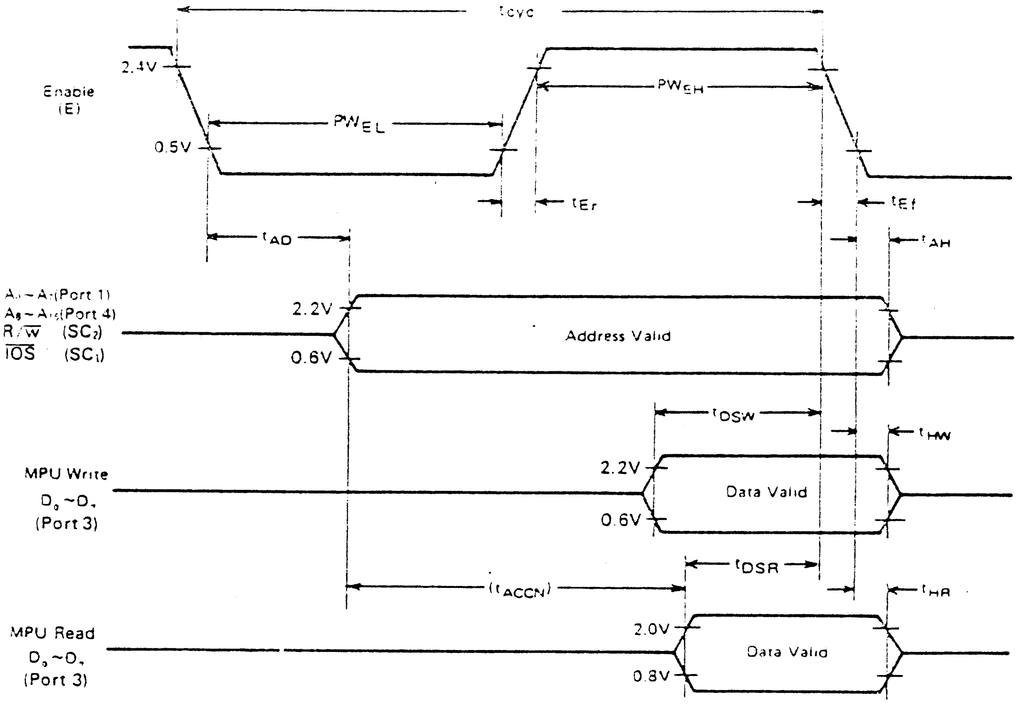
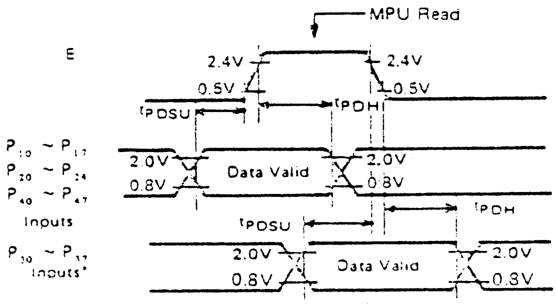
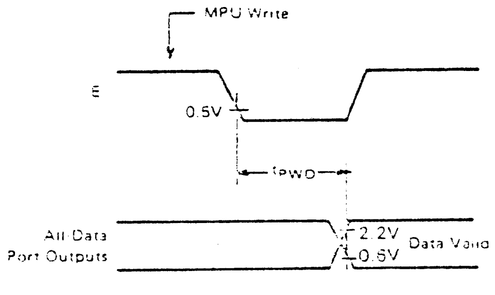


Fig. 2 Expanded Non-Multiplexed Bus Timing



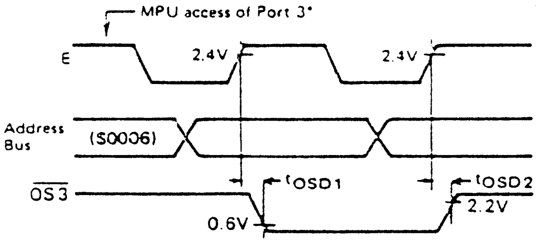
* Port 3 Non-Latched operation

Fig. 3 Port Data Set-up and Hold Times (MPU Read)



Note: Port 2, Except P_{21}

Fig. 4 Port Data Delay Times (MPU Write)



* Access matches Output Strobe Select (OSS=0, a read; OSS=1, a write)

Fig. 5 Port 3 Output Strobe Timing (Single Chip Mode)

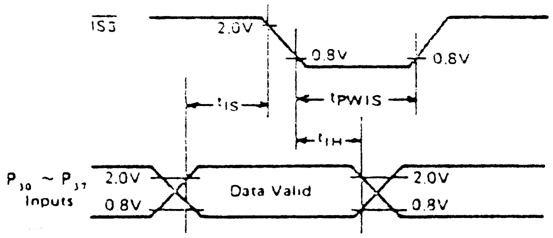


Fig. 6 Port 3 Latch Timing (Single Chip Mode)

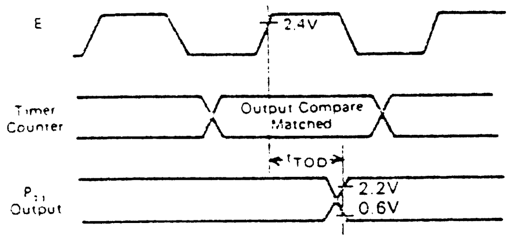


Fig. 7 Timer Output Timing

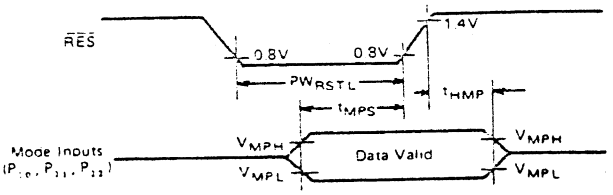
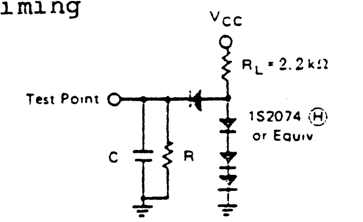


Fig. 8 Mode Programming Timing



C = 90 pF for P_{30}~P_{37}, P_{40}~P_{47}, E, SC_1, SC_2
= 30 pF for P_{10}~P_{17}, P_{20}~P_{24}
R = 12 kΩ for P_{30}~P_{37}, P_{40}~P_{47}, E, SC_1, SC_2
= 24 kΩ for P_{10}~P_{17}, P_{20}~P_{24}

Fig. 9 Bus Timing Test Loads (TTL Load)

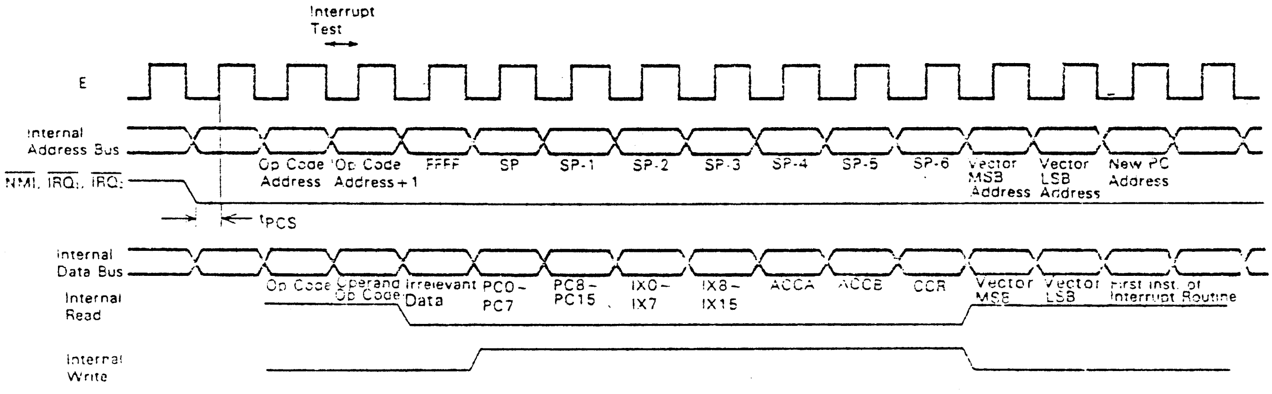


Fig. 10 Interrupt Sequence

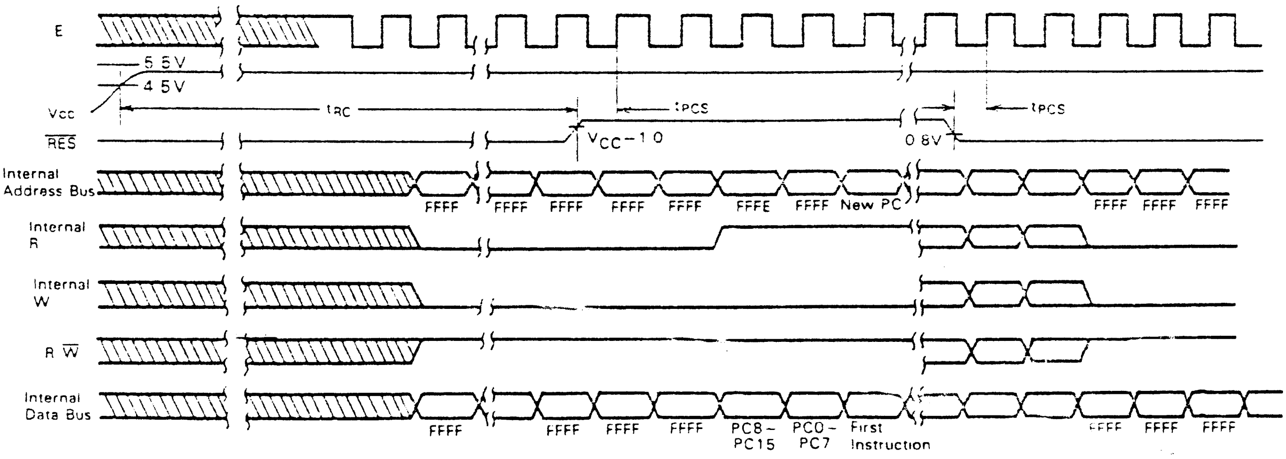


Fig. 11 Reset Timing

DESCRIPTION ON FUNCTIONS OF PIN

-VCC and VSS

These two pins are used for the power supply and the ground (GND). Recommended power supply voltage is +5 V ±10%. 3 through 7 V are used for low speed operation.

- XTAL and EXTAL

An AT-cut parallel resonant fundamental crystal is connected to these two pins. For instance, to obtain the 1-MHz system clock, a 4-MHz resonant fundamental crystal is used because the 4-demultiply circuit is incorporated.

The EXTAL receives an external clock input of duty 50% (±10%) to drive. In this case, the LSI internally obtains the system clock which is a quarter the frequency of an external clock. The external driving frequency less than 4 times the maximum operating frequency (maximum internal clock) can be used. For external driving, no XTRAL should be connected. Fig. 12 is an example of the connected crystal circuit.

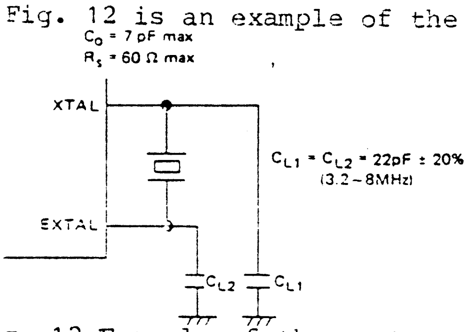


Fig. 12 Example of the connected crystal circuit

- $\overline{\text{STBY}}$

This pin is used to place the MCU in standby mode. When this pin is set to LOW, the internal condition is reset because the oscillation is inactivated and the internal clock is fixed. To retain information in RAM during standby, 0 must be written into RAM enable bit (RAME). The RAM is bit 6 of the RAM control register at address $0014.

Since this operation disables the RAM, the contents of that RAM is guaranteed in standby mode. For details on the standby mode, see the STANDBY section.

- Reset (RES)

This pin is used to reset and start the MCU from the power off state. The RES must be held at Low level for at least 20 ms when the power is on. To reset the MCU during system operation, it must be held at Low level for at least 3 system clock cycles. From the third cycle on, address buses go High with the RES at Low level.

When the "High" level is detected, the MCU performs the following:

1) The MCU latches bits 0, 1, and 2 of I/O port 2 into bits PC0, PC1, and PC2 of the program control register, respectively.
2) The MCU stores the contents of two start addresses $FFFE and $FFFF in the program counter from which the program starts (Table 1).
3) The MCU sets the interrupt mask bit. To allow the MCU to recognize maskable interrupts an IRQ1 and IRQ2, these interrupts must be cleared beforehand.

- Enable (E)
This pin is used to supply a system clock to the rest of the system when the internal oscillator is being used. The output is a single-phase and compatible with the TTL. The clock frequency is quarter the crystal oscillation frequency. This pin can drive one TTL load and 90 pF capacity.

- Nonmaskable interrupt (NMI)
If the fall of the input signal of this pin is detected, the NMI sequence starts in the MCU. As in the case of the interrupt request IRQ1, the instruction being executed when the NMI is detected continues to be executed to the last. The interrupt mask bit in the condition code register does not affect the NIM at all.
When a response is made to the NMI, the contents of the program counter, index register, accumulators, and condition code register are saved to the stack area. When this sequence terminates, vector addresses $FFFC and FFFD occur to read the contents of these addresses into the program counter and branch to the nonmaskable interrupt service routine. IRQ1 and NMI are hardware interrupt pins sampled by an internal clock. After execution of the instruction, the interrupt routine must be generated in synchrozization with the E.

- Interrupt request (IRQ1)
This level detection pin is used to generate an interrupt sequence in the MPU. If there is any instruction being executed when an interrupt request is issued, the MPU holds the request until it completes the execution of that instruction. If the interrupt mask bit in the condition code register is not set, the MPU initiates the interrupt sequence; otherwise, the interrupt request is ignored.
Once the sequence has started, the MPU saves the contents of the program counter, index register, accumulators, and condition code register to the stack area, and sets the interrupt mask bit, inhibiting further maskable interrupts to occur. At the end of the cycle, the MPU generates a 16-bit vector address indicating memory addresses $FFF8 and $FFF9, stores the contents of these addresses into the program counter, and branches to the interrupt service routine.

Table 1 Interrupt vectoring memory map

	Vector		Interrupt
	MSB	LSB	
Highest Priority	FFFE	FFFF	$\overline{RES}$
	FFEE	FFEF	TRAP
	FFFC	FFFD	$\overline{NMI}$
	FFFA	FFFB	Software Interrupt (SWI)
	FFF8	FFF9	$\overline{IRQ}_1$ (or $\overline{IS3}$)
	FFF6	FFF7	ICF (Timer Input Capture)
	FFF4	FFF5	OCF (Timer Output Compare)
	FFF2	FFF3	TOF (Timer Overflow)
Lowest Priority	FFF0	FFF1	SCI (RDRF + ORFE + TDRE)

The internal interrupt generates internal interrupt signal IRQ2, which is the same as the IRQ1, except that it uses vector addresses $FFF0 through $FFF7. If IRQ1 and IRQ2 are generated at the same time, the former precedes the latter. The interrupt mask bit in the condition code register disables these interrupts if it is set.
If an address error or an op-code error occurs, the TRAP interrupt, the second highest priority interrupt next to the RES, will also occur. In this case, the MPU starts the interrupt sequence independently of the mask bit condition. The vector for this interrupt is FFEE, FFEF.

The following 3 pins apply only to port 3 in single chip mode.

- Input strobe (IS3) (SC1)
This signal controls the IS3 interrupt and the latch of port 3. When the fall of this signal is detected, the IS3 flag of the port 3 control status register is set. For details on the bits of the port 3 control status register, see the I/O PORT 3 CONTROL STATUS REGISTER section.

- Output strobe (OS3) SC2)
This signal is used when the processor informs the external device that effective data is on the I/O pins. Fig. 5 shows the timing of the port 3 output strobe.
The following pins apply to expanded mode.

-Read/write (R/W) (SC2)
This TTL compatible output signal is used to inform the peripheral device and memory whether the MCU is in the read (High) or write (Low) state. This signal is usually in the standby state. The output drives one TTL load and 90-pF capacitor.

- I/O strobe (IOS) (SC1)
In expanded nonmultiplexed mode 5 of operation, this signal internally decodes A9 through A15 as "0" and A8 as "1", allowing the access of up to 256 addresses from $0100 to $01FF in external memory.
Fig. 2 shows the timing of expanded nonmultiplexed bus.

-Address strobe (AS) (SC1)
In expanded multiplexed mode, an address strobe signal appears at this pin. This signal is used to latch the lower 8-bit addressed multiplexed with data on port 3 and to control the 8-bit latch using the address strobe as shown in Fig. 18.
This allows the user to use I/O port 3 as a data bus during E pulse.
Fig. 19 shows the timing of the expanded multiplexed bus.

PORTS
The HD6301V MCU provides 4 I/O ports (three 8-bit ports and one 5-bit port). 2 control pins are connected to one the of the 8-bit ports.
Each port has an independent write-only data direction register to program individual I/O pins for input and output.

When the bit of the corresponding data direction register is 1, the I/O pin is programmed for output. When the bit is "0", The I/O pin is programmed for input. There are 4 ports: port 1, port 2, port 3, and port 4. Table 2 lists the correspondence between the 4 I/O port addresses and data direction register addresses.

: Only one exception is bit 1 of port 2. Since this bit (terminal) is used as a data input or a timer output, it cannot be used as an output port.

Table 2 Port and data direction register addresses

Ports	Port Address	Data Direction Register Address
I/O Port 1	$0002	$0000
I/O Port 2	$0003	$0001
I/O Port 3	$0006	$0004
I/O Port 4	$0007	$0005

- I/O port 1

This is an 8-bit port, each bit being defined as input or output according to the contents of the corresponding data direction register.

The 8-bit, three-state output buffer changes to the high-impedance state when it is used for input. For accurate data read, the voltage on the input line (pin) must be 2.0 V or above when the logic is 1 and must be 0.8 V or below when the logic is 0.

These input lines (pins) compatible with the TTL can be used not only as the 1-mA or less drive source with 1.5 V but also to directly drive the darlington transistor base.

After the MCU is reset, all the I/O lines (pins) function as input.

In modes other than expanded nonmultiplexed mode 1, port 1 always functions as the parallel I/O line (pin). In expanded nonmultipelexed mode 1, port 1 functions as the output line (pin) for lower address lines A0 through A7.

- I/O port 2

This port has 5 lines (pins), whose I/O directions depend on the data direction register.

The 5-bit, three-state output buffer changes to the high-impedance state when it is used for input. For accurate data read, the voltage on the input line (pin) must be 2.0 V or above when the logic is 1 and must be 0.8 V or below when the logic is 0.

After the MCU is reset, all I/O lines (pins) function as input.

Pins 8 through 10 of port 2 are used for mode programming during reset. The values of these 3 pins during reset are latched into the upper 3 bits (bits 5, 6, and 7) of port 2. For details, see the MODE SELECTION section.

In all modes, port 2 not only functions as the I/O line (pin) but also provides access to the SCI and timer. Note that bit 1 (P21) can be used only for the data input port and the timer output.

- I/O port 3

This 8-bit port can be used as an I/O terminal, a data bus, or an address but multiplexed with a data bus. The function depends on hardware operation mode programmed by the user using 3 bits of port 2 during reset. Port 3 used as a data bus is bidirectional.
For an input from a peripheral device, a standard TTL level must be supplied. In other words, the level must be 2.0 V or above when the logic is 1 and must be 0.8 V or below when it is 0. This TTL compatible, 3-state buffer can drive one TTL load and the 90 pF capacitor.
In expanded mode, the use of the data direction register is inhibited and the data flow depends on the status of the R/W signal (line).
Port 3 has the following functions in each mode:
Single chip mode (mode 7)
In this mode, port 3 is used for parallel I/O and defined as input or output using the corresponding data direction register. In this mode, two control lines (pins) input strobe (IS3) and output strobe (OS3), are used for handshaking and controlled by the I/O port 3 control/status register.
Three additional characteristics of this register are summarized as follows:

(1) Input data of port 3 can be latched using IS3 (SC1) as a control signal.
(2) OS3 (SC2) is generated when the port 3 data register is read or written.
(3) Interrupt IRQ1 is generated when the IS3 falls. Figs. 5 and 6 show the timings of port 3 strobe and latch, respectively.

I/O port 3 control/status register

	7	6	5	4	3	2	1	0
\$000F	$\overline{\text{IS3}}$ FLAG	$\overline{\text{IS3}}$ $\overline{\text{IRQ}_1}$ ENABLE	X	OSS	LATCH ENABLE	X	X	X

Bit 0: Not used.
Bit 1: Not used.
Bit 2: Not used.

Bit 3: LATCH ENABLE
This bit is used to control the input latch of I/O port 3. If the bit is set at 1, the input data on port 3 is latched by the falling edge of the IS3. This latch is cleared and then enters the latch possible state again when a signal is read from I/O port 3. This bit is cleared by a reset.

Bit 4: OSS (Output Strobe Selection)
This bit is used to identify the cause of the output strobe generation. The output strobe is generated when a signal is read and written to and from I/O port 3. In other words, the strobe is generated in the following cases:
- A signal is read from I/O port 3 and bit 4 is cleared.
- A signal is written into I/O port 3 and bit 4 is set.

This bit is cleared by a reset.

Bit 5: Not used.

Bit 6: IS3 ENABLE

If both bit 6 and bit 7 (IS3 flag) are set, the interrupt is enabled. If the flag is cleared, the interrupt is disabled. This bit is cleared by a reset.

Bit 7: IS3 FLAG

This read-only bit is set by the falling edge of the IS3 (SC1). The bit is cleared when a signal is read from or written into I/O port 3 after the control/status register is read. It is cleared by a reset.

Expanded nonmultiplexed mode (modes 1 and 5)

In this mode, port 3 is used as data buses D0 through D7.

Expanded multiplexed mode (modes 0, 4, and 6)

In this mode, port 3 is used as data buses D0 through D7 and multiplexed lower 8 bits of addresses buses A0 through A7.

- I/O port 4

This 8-bit port functions as either the I/O line (pin) or the address output line (pin) depending on the operation mode selected.

The voltage on the input line (pin) must be 2.0 V or above when the logic is 1 and must be 0.8 V or below when it is 0. The line (pin) compatible with the TTL can drive one TTL load and the 90 pF.

Since this port is used for input after reset, it must be programmed as output when the lines (pins) are used as addresses.

Port 4 has the following characteristics in each mode:

Single chip mode (mode 7)

In this mode, port 4 is used for parallel I/O and defined (programmed) as input or output using the corresponding data direction register.

Expanded nonmultiplexed mode (mode 5)

In this mode, port 4 function as lower address lines (pins) A0 through A7 when 1 is written into the data direction register. If some of the 8 bits are not required as addresses, the remaining lines (pins) can be used as the I/O lines (pins).

Expanded nonmultiplexed mode (mode 1)

In this mode, port 4 functions as upper address output lines A8 through A15 when 1 is written into the data direction register. If some of the 8 bits are not required as addresses, the remaining lines (pins) can be used as the I/O lines (pins) starting with the MSB.

Expanded multiplexed mode (modes 0 and 4)

In this mode, port 4 functions as upper address output lines A8 through A15 regardless of the value of the data direction register.

Table 3 lists the relationship between the modes and I/O ports 1 through 4.

MODE SELECTION

The user must determine the operation mode used after the HD6301V is reset by wiring pins 8, 9, and 10 externally. These 3 pins, which correspond to lower 3 bits (I/O 0, I/O 1, and I/O 3) of port 2, respectively, are latched into program control bits PCO, PS1, and PC2 of the I/O port 2 register when the reset goes High.

The format of the I/O port 2 register is as follows:

Port 2 data register

	7	6	5	4	3	2	1	0
$0003	PC2	PC1	PC0	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0

Fig. 13 shows an example of the external hardware used for mode selection. The HD14053B, which separates the peripheral device from the MCU during reset, is necessary when a data contention occurs between the peripheral device and the mode generation circuit.
The mode cannot be changed by software because bits 5, 6, and 7 of port 2 are for read only. Table 3 lists the modes selected by the HD6301V.
The HD6301V operates in the following 3 basic modes:

(1) Single chip
(2) Expanded multiplexed (compatible with the HMCS6800 peripheral LSI)
(3) Expanded nonmultiplexed (compatible with the HMCS6800 peripheral LSI)

- Single chip mode

In this mode, all ports are used for the I/O functions. Fig. 15 shows the MCU single chip mode.
In this mode, pins SC1 and SC2 function as control lines (pins) of port 3 and can also be used as input strobe signal IS3 and output strobe signal OS3.

- Expanded multiplexed mode

In this mode, port 4 functions as I/O or address lines (pins). The data bus and the lower address buses multiplexed in port 3 can be separated by the address strobe output. Port 2 functions as 5 parallel I/O lines (pins), a serial I/O, or a timer. Any combination of these functions is also possible. Port 1 functions as 8 parallel I/O lines (pins) and can be expanded to the address space of up to 65K words (Fig. 16).

- Expanded nonmultiplexed mode

In this mode, the HD6301V can directly address the HMCS6800 peripheral devices with no external logics. Port 1 can be used only as the parallel I/O lines (pins). Port 2 functions as the parallel I/O lines (pins) for the serial I/O, or a timer. Any combination of these functions is also possible. Port 3 functions as a data bus. Port 4 functions as address buses A0 through A7, a part of an address bus, or I/O (input only). In this mode, the HD6301V can address up to 256 addresses. In the application system with fewer addresses, the remaining pins of port 4 can be used as I/O lines (input only) (Fig. 17).

In mode 1, port 1 functions as address buses A0 through A7; port 2 functions as parallel I/O lines, serial I/O lines, or a timer (any combination of these functicns is also possible) port 3 functions as a data bus; port 4 funcions as address buses A8 through A15 or a part of an address bus, or I/O (input only). In this mode, the HD6301V can be expanded to the address space of up to 65K words without external logics. In the application system with fewer addresses, the remaining pins of port 4 can be used as the I/O lines (input only) (Fig. 17).

- Lower address bus latch

Since the data bus is multiplexed with the lower address bus of port 3 in expanded multiplexed mode, the address bits must be externally latched. The lower address bytes can be latched by using the 74LS373 with 8 D-byte flip-flops. Fig. 18 shows how the HD6301V and the 74LS373 are connected.

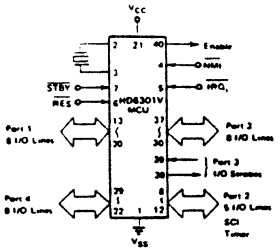
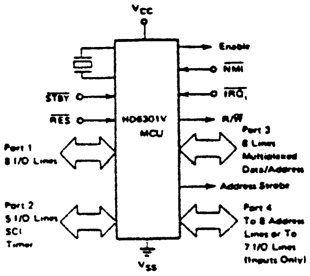
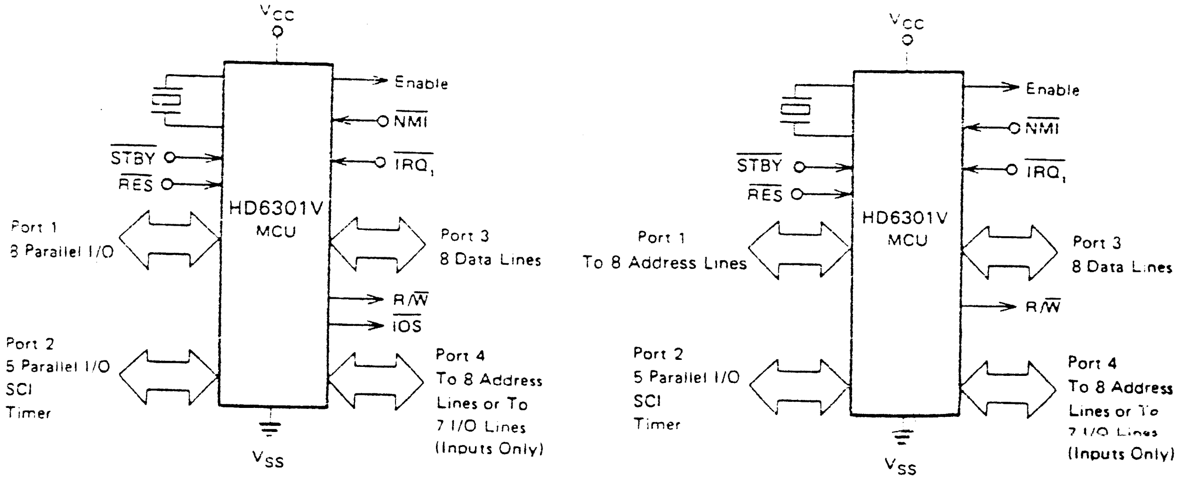


Fig. 15 HD6301V MCU single chip mode



Vcc
Enable
NMI
IRQ1
STBY
RES
R/W
HD6301V MCU
Port 1 8 I/O Lines
Port 3 8 Lines Multiplexed Data/Address
Address Strobe
Port 2 5 I/O Lines SCI Timer
Port 4 To 8 Address Lines or To 7 I/O Lines (Inputs Only)
Vss

Fig. 16 HD6301V MCU expanded multiple ed mode

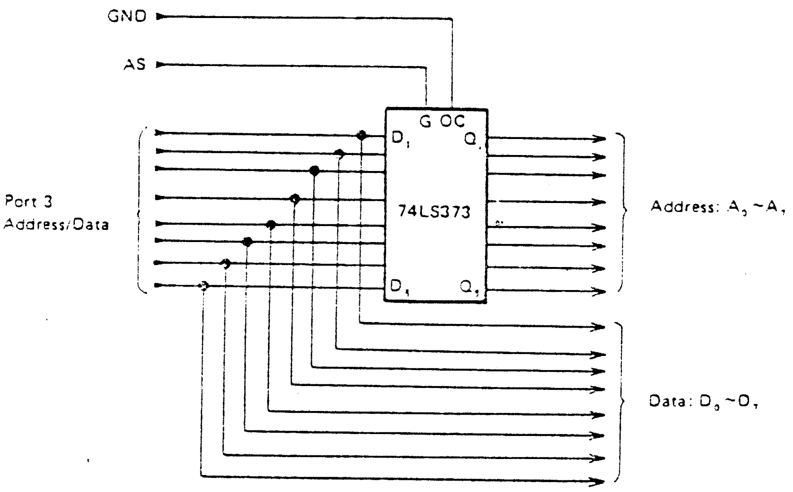


Vcc
Enable
NMI
IRQ1
STBY
RES
HD6301V MCU
Port 1 8 Parallel I/O
Port 3 8 Data Lines
R/W
IOS
Port 2 5 Parallel I/O SCI Timer
Port 4 To 8 Address Lines or To 7 I/O Lines (Inputs Only)
Vss
Vcc
Enable
NMI
IRQ1
STBY
RES
HD6301V MCU
Port 1 To 8 Address Lines
Port 3 8 Data Lines
R/W
Port 2 5 Parallel I/O SCI Timer
Port 4 To 8 Address Lines or To 7 I/O Lines (Inputs Only)
Vss

(a) Mode 5

(b) Mode 1

Fig. 17 HD6301V MCU expanded nonmultiplexed mode



Function Table

Output Control	Enable G	D	Output Q
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

Fig. 18 Latch connection

- Outline of the modes and ports and MCU signals

This section explains the MCU signals operating in various modes. The SCI and SC2 signals vary with the state of the chip mode.

Table 3 Features of each mode and line

MODE		PORT 1 Eight Lines	PORT 2 Five Lines	PORT 3 Eight Lines	PORT 4 Eight Lines	SC_1	SC_2
SINGLE CHIP		I O	I O	I O	I O	$\overline{IS3}$ (I)	$\overline{OS3}$ (O)
EXPANDED MUX		I O	I O	ADDRESS BUS ($A_0 \sim A_7$) DATA BUS ($D_0 \sim D_7$)	ADDRESS BUS* ($A_8 \sim A_{15}$)	AS(O)	R $\overline{W}$(O)
EXPANDED	Mode 5	I O	I O	DATA BUS ($D_0 \sim D_7$)	ADDRESS BUS* ($A_0 \sim A_7$)	$\overline{IOS}$(O)	R $\overline{W}$(O)
NON-MUX	Mode 1	ADDRESS BUS ($A_0 \sim A_7$)	I O	DATA BUS ($D_0 \sim D_7$)	ADDRESS BUS* ($A_8 \sim A_{15}$)	Not Used	R $\overline{W}$(O)

*: If these lines (pins) are unnecessary to be used as the address lines, they can be used as the I/O lines (only for input) starting with the MSB (excluding modes 0 and 4).

I=Input	IS3=Input Strobe	SC=Strobe Control
O=Output	OS3=Output Strobe	AS=Address Strobe
R/W=Read/Write	IOS=I/O Select	

Table 4 Mode selection Summary

Mode	P_{22} (PC2)	P_{21} (PC1)	P_{20} (PC0)	ROM	RAM	Interrupt Vectors	Bus Mode	Operating Mode
7	H	H	H	I	I	I	I	Single Chip
6	H	H	L	I	I	I	MUX(4)	Multiplexed/Partial Decode
5	H	L	H	I	I	I	NMUX(4)	Non-Multiplexed/Partial Decode
4	H	L	L	E(2)	I(1)	E	MUX	Multiplexed/RAM
3	L	H	H	—	—	—	—	Not Used
2	L	H	L	—	—	—	—	Not Used
1	L	L	H	I	I	I	NMUX (4)	Non-Multiplexed/Partial Decode
0	L	L	L	I	I	I(3)	MUX	Multiplexed Test

HX-20 is set to Mode 4.

Legends
- I-Internal
- E-External
- MUX-Multiplexed
- NMUX-Nonmultiplexed
- L-Logic "0"
- H-Logic "1"

Notes:
1) The internal RAM is addressed at $0080.
2) The internal ROM cannot be used.
3) The RES vector is external for 3 cycles after RES goes High.
4) Unnecessary address output lines (pins) of port 4 can be used as the input port.

MEMORY MAP

The MCU can address up to 65K bytes depending on the operation mode used. Fig. 19 shows a memory map for each operation mode. The first 32 addresses of each memory map are for exclusive use of the internal register as shown in Table 5.

Table 5 Internal register Area

Register	Address
Port 1 Data Direction Register****	00*
Port 2 Data Direction Register****	01
Port 1 Data Register	02*
Port 2 Data Register	03
Port 3 Data Direction Register****	04**
Port 4 Data Direction Register****	05***
Port 3 Data Register	06**
Port 4 Data Register	07***
Timer Control and Status Register	08
Counter (High Byte)	09
Counter (Low Byte)	0A
Output Compare Register (High Byte)	0B
Output Compare Register (Low Byte)	0C
Input Capture Register (High Byte)	0D
Input Capture Register (Low Byte)	0E
Port 3 Control and Status Register	0F**
Rate and Mode Control Register	10
Transmit/Receive Control and Status Register	11
Receive Data Register	12
Transmit Data Register	13
RAM Control Register	14
Reserved	15-1F

*: These addresses become external addresses in mode 1.
**: These addresses become external addresses in modes 0, 1, 4, 5 and 6 but cannot be accessed in mode 5.
***: These addresses become external addresses in modes 0 and 4.
****: 1: Output 0: Input

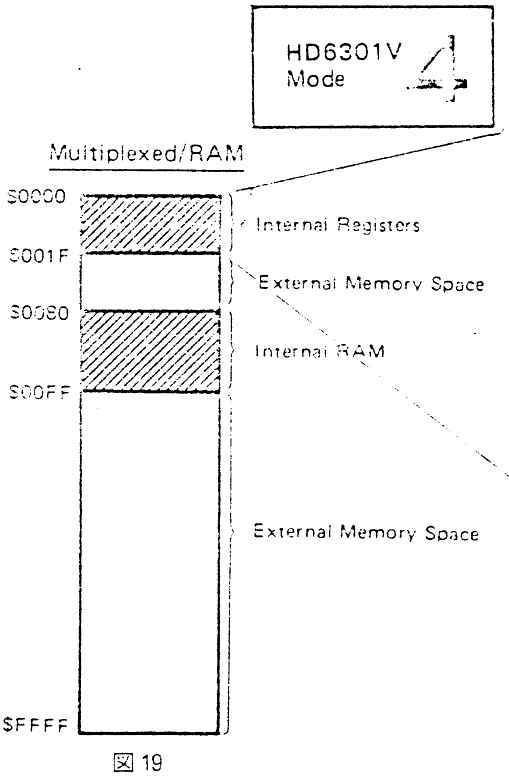


図19

Fig. 19

Note: Addresses $04, $05, $06, $07 and $OF which can be externally used are not excluded.

Programmable Options

The HD6301V provides the following programmable options for the serial I/O section:

- Data format: Standard mark/space (NRZ)
- Clock source: External or internal
- Baud rate: One of 4 baud rates is selected for the E clock of the MCU or 1/8 of the external clock is selected.
- Wake-up function: Enable or disable
- Interrupt request: Used to enable or mask the transmit and receive data registers.
- Clock output: Used to enable or disable the output to bit 2 of internal-clock port 2.
- Port 2 (bit 3, 4): Indicates whether or not to use bits 3 and 4 of port 2 for the receiver and transmitter.

Serial Communication Hardware

The serial communication hardware is controlled by the following 4 registers as shown in Fig. 22.

- 8-bit control/status register
- 4-bit transfer rate/mode control register (write-only)
- 8-bit read-only receive register
- 8-bit write-only transmit data register

In addition to these 4 registers, the serial communication section uses bit 3 (input) and bit 4 (output) of port 2. Bit 2 is used when an option that externally outputs an internal clock or an option that internally inputs an internal clock is selected.

Transmit/Receive Control Status Register (TRCSR)

The TRCSR consits of 8 bits. All these bits can be read and bits 0 through 4 can also be written. RES initializes this register for $20. The structure of the TSCSR is:

Transmit/receive control status register (TRCSR)

7	6	5	4	3	2	1	0	
RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU	ADDR $0011

Bit 0: WU (Wake-up)

This bit is set by software and cleared by hardware when 10 1s are consecutively received. Note that this bit must be set before the RE flag is set.

Bit 1: TE (Transmit Enable)

When this bit is set, a preamble of 10 consecutive 1 bits is generated, enabling the output of transmitter data to bit 4 of port 2 independently of the value of the DDR bit corresponding to this bit. When this bit is cleared, the serial I/O section does not affect bit 4 of port 2 at all.

Bit 2: TIE (Transmit Interrupt Enable)

When this bit and bit 5 (TDRE) are both set, internal interrupt IRQ2 can be generated. When this bit is cleared, internal interrupt IRQ2 of bit 5 (TDRE) is masked.

Bit 3: RE (Receive Enable)
When this bit is set, the bit 3 signal of port 2 is sent to the receiver regardless of the contents of the DDR corresponding to this bit. If this bit is cleared, the serial I/O section does not affect bit 3 of port 2 at all.

Bit 4: RIE (Receive Interrupt Enable)
If bit 7 (RDRF) or bit 6 (ORFE) is set when this bit is set, IRQ2 can be generated. When this bit is cleared, IRQ2 is masked.

Bit 5: TDRE (Transmit Data Register Empty)
This bit is set by hardware when data is sent from the transmit data register to the output shift register. This bit is cleared when next new data is written into the transmit data register following the status register RES sets this bit to the 1.

Bit 6: ORFE (Overrun Framing Error)
This bit is set by hardware when an overrun or framing error occurs. The overrun error occurs if new data is sent to the receive register when bit 7 (RDRF) is set.
The framing error occurs when the bit counter does not synchronize with the byte boundary of the bit stream. This bit is cleared when the receive data register is read following the status register. RES can also be used to clear the bit.

Bit 7: RDRF (Receive Data Register Full)
This bit is set by hardware when data is sent from the receive shift register to the receive data register. The bit is cleared when the receive data register is read following the status register. RES can also be used to clear the bit.

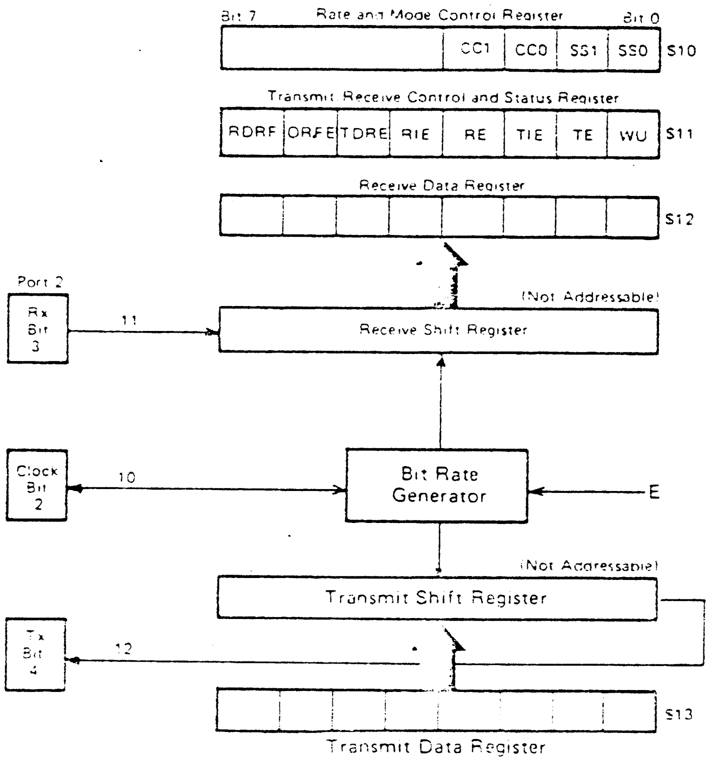


Fig. 22 Serial I/O register

Transfer rate/mode control register

7	6	5	4	3	2	1	0	
X	X	X	X	CC1	CC0	SS1	SS0	ADDR : $0010

Table 6 SCI bit times and transfer rates

		XTAL	2.4576 MHz	4.0 MHz	4.9152MHz
SS1	SS0	E	614.4 kHz	1.0 MHz	1.2288MHz
0	0	E ÷ 16	26 μs/38,400 Baud	16 μs/62,500 Baud	13 μs 76,800Baud
0	1	E ÷ 128	208μs/4,800 Baud	128 μs/7812.5 Baud	104.2μs, 9,600Baud
1	0	E ÷ 1024	1.67ms/600 Baud	1.024ms/976.6 Baud	833.3μs, 1,200Baud
1	1	E ÷ 4096	6.67ms/150 Baud	4.096ms/244.1 Baud	3.333ms 300Baud

Table 7 SCI format and clock source control

CC1, CC0	Format	Clock Source	Port 2 Bit 2	Port 2 Bit 3	Port 2 Bit 4
00	–	–	–	–	–
01	NRZ	Internal	Not Used	**	**
10	NRZ	Internal	Output*	**	**
11	NRZ	External	Input	**	**

*: A clock is output regardless of the status of TE in the TRCSR.

**: When RE bit of TRCS is "1", bit is used as the serial input.
When TE bit of TRCS is "1", bit 4 if is used as the serial output.

Transfer Rate/Mode Control Register (RMCR)

This register controls the following variables of the serial I/O section:

- Baud rates
- Data format
- Clock source
- Bit 2 function of port 2

This 4-bit write-only register is cleared by RES. Each 2 bits consistutes one field. The lower 2 bits are used to control the bit rates for the internal clocks while the upper 2 bits are used to control the data format and the clock selection logic.

Bit 0: SSO
Bit 1: SS1 Speed selection

These bits are used to select the baud rates for the internal clocks. These selectable 4 rates can be used as the functions of the E clock frequencies in the MPU. Table 6 lists the baud rates to be selected.

Bit 2: CC0
Bit 3: CC1 Clock control/format selection

These bits are used to control the data format and the clock source selection logic. Table 7 defines these two bits.

- Internally generated Clock

The following points should be taken into consideration when the user externally uses an internal clock for the serial I/O:

- The values of bit 1 (RE) and bit 3 (RE) can be ignored.
- CC0 and CC1 are set to "10".
- The available maximum clock rate is E ÷ 16.
- The clock is one time the bit rate.

- Externally Generated Clock

The following should be taken into consideration when the user supplies an external clock to the serial I/O:

. Bits 2 (CC0) and 3 (CC1) of the RMC register are set to "11" (Table 7).
. The external clock frequency is 8 times the baud rate to be used.
. The maximum external clock frequency is the same as the E clock frequency.

- Serial Operation

The serial I/O hardware must be initialized by the HD6301V software in the following sequence before operation:

. A desired operation control bit is written into the RMC register.
. A desired operation control bit is written into the TRCS register.

If bits 3 and 4 of port 2 are exclusively used for the serial I/O hardware, bit 1 (TE) and bit 3 (RE) need not be cleared. If these bits (TE and RE) are temporarily cleared during the SCI operation and then set again, they must be set at least one bit after the current baud rate. Note that if bit 1 (TE) and bit 3 (RE) are set within one bit interval, the internal status for reception and transmission may not be initialized.

- Transmit Operation

When bit 1 (TE) of the TRCS register is set, the transmit operation starts. In other words, when this bit is set, the contents of the serial transmit shift register are unconditionally output to bit 4 of port 2 regardless of the value of the DDR corresponding to bit 4 of port 2.

After RES is output, both the RMC and TRCS registers must be set to the desired operation conditions. If bit 1 (TE) is set during this procedure, a preamble of 10 consecutive "1" bits is transmitted. After the preamble transmission, internal synchronization is established and the transmitter enters the operable state.

At this point, one of the following operations is performed:

(1) If the transmit data register is empty (TDRE=1), 1 bits are consecutively sent, indicating that the idle state is kept.

(2) If data is already stored in the transmit data register (TDRE=0), data is sent to the output shift register to start the data transmission.

The "0" start bit is first sent, and then 8 bit data (beginning with bit 0), and the stop bit follows.

If the transmit data register becomes empty, hardware sets the TDRE flag bit.

If the MCU fails to respond to the flag within a specified time, 1 bits are consecutively output instead of the 0 start bit until data is supplied to the data register. Note that in this case, the TDRE flag bit remains to be set until next correct data is sent from the parallel data register to the serial output register. When the TDRE bit keeps 1, no 0 bits are sent.

- Receive Operation

When bit 3 (RE) is set, the receive operation starts and enables the bit 3 of port 2 to be used as the serial input.

The operation conditions of the receiver are determined according to the contents of the TRCS and RMC registers. In ordinary non-biphase mode, the flow of the received bit is synchronized by the first input Ø (space). During 1Ø-bit time, each bit is strobed in the almost middle of it.
If the tenth bit is not 1 (stop bit), a framing error assumes to occur, causing the ORFE bit to be set. When the tenth bit is 1, data is transmitted to the receive data register to set the RDRF interrupt flag. If the RDRF flag still remains to be set although the tenth bit of next data is received, the ORFE flag indicating that an overrun has occurred is set.
If the MCU reads the data register following the status register in response to the RDRF or ORFE flag, the RDRF or ORFE flag is cleared.

RAM CONTROL REGISTER
The register assigned to address $ØØ14 specifies information on the standby RAM.

RAM control register

	7	6	5	4	3	2	1	0
$0014	STBY PWR	RAME	x	x	x	x	x	x

Bit Ø: Not used.
Bit 1: Not used.
Bit 2: Not used.
Bit 3: Not used.
Bit 4: Not used.
Bit 5: Not used.
Bit 6: RAM Enable
This bit is used to disable the standby RAM. When the MPU is reset, 1 is set in this bit, enabling the standby RAM. 1 and Ø can also be written into the bit by using the program control.
When the RAM is disabled (logic Ø), the RAM address is internally invalidated, allowing the MPU to read data from external memory.
Bit 7: Standby bit
This bit is cleared when no Vcc voltage is impressed. The user can also read this bit using the read/write flags. When this bit is set, it indicates that the Vcc voltage has been impressed and data in the standby RAM is effective.

OUTLINE OF INSTRUCTION SET
The HD63Ø1V has HD68Ø1 upward compatible object codes to use all HMCS68ØØ instruction sets and the execution time of the key instructions is designed to improve the system throughput.
Bit manipulation instructions, change instructions and sleep instructions for indexes and accumulators are also added.
This section describes:
- MCU programming model (Fig. 23).
- Addressing mode
- Manipulation instructions for accumulators and memory (Table 8).
- New instructions.
- Manipulation instructions for index registers and stacks (Table 9).
- Jump and branch instructions (Table 1Ø).
- Manipulation instructions for condition code registers (Table 11).
- Op-code Map (Table 12).

- MCU Programming Model

Fig. 23 shows the HD6301V programming model. Since double accumulator D is a combination of accumulators A and B, the contents of A and B are destroyed when the instructions that use accumulator D are specified.

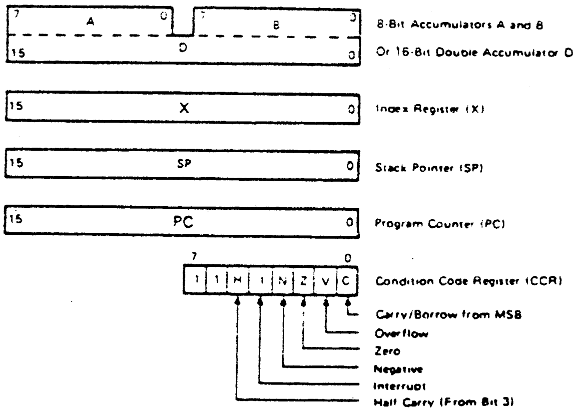


Fig. 23 MCU programming model

MCU Addressing Mode

The HD6301 is provided with 7 addressing modes each of which is selected according to both the instruction type and code. Tables 8 through 12 show the addressing mode for each instruction together with the execution time represented by the number of machine cycles. When the clock frequency is 4 MHz, microseconds replace the machine cycles.

Accumulator (ACCX) Addressing

Only the accumulator is addressed. In other words, either accumulator A or B is specified. 1-byte instructions are used to specify this mode.

Immediate Addressing

In this mode, an operand is included in the second byte of an instruction, but in the LDS and LDX, the operand is included both in the second and third bytes. 2- or 3-byte instructions are used to specify this mode.

Direct Addressing

The second byte of an instruction indicates the address that contains the operand in this mode so that the 256-byte area from address 0 through address 255 can be directly accessed. Since storing data in this area reduces the execution time, it is recommended that the user define this area as the RAM area when designing the system configuration and use it as the user data area. 2-byte instructions are usually used to specified this mode but 3-byte instructions such as AIM, OIM, EIM, and TIM can also be used.

Extended Addressing

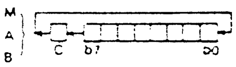
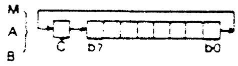
In this mode, the second byte indicates the high-order 8 bits of the address that contains the operand and the third byte indicates the low-order 8 bits. This address indicates an absolute address in memory. 3-byte instructions are used to specify this mode.

Indexed Addressing
In this mode, the contents of the second byte of an instruction and the low-order 8 bits of the indexed register are added. The contents of the third of an instruction byte and the low-order 8 bits are added for the AIM, OIM, EIM, and TIM,.
This carry is also added to the high-order 8 bits of the index register and the resulting values are used as memory addresses. The contents of the index register do not change because the temporary address register holds this modified address. 2-byte instructions are usually used to specify this mode but 3-byte instructions such as AIM, OIM, EIM, and TIM can also be used.

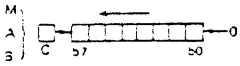
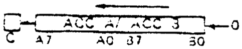
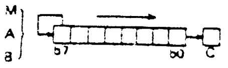
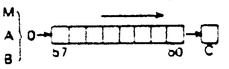
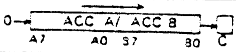
Implied Addressing
In this mode, the instruction itself specifies the address, i.e., it addresses the stack pointer and index registers, etc. 1-byte instructions are used to specify this mode.

Relative Addressing
In this mode, the contents of the second byte of an instruction and the low-order 8 bits of the program counter are added and then the resulting carry or borrow is added to the high-order 8 bits. This means that the user can address data within the range of -126 through +129 bytes of the instruction currently being executed. 2-byte instructions are used to specify this mode.

Table 8 Manipulation instructions for accumulators and memory

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Double	ADDD	C3	4	3	D3	4	2	E3	5	2	F3	5	3				A : B + M : M + 1 → A : B	•	•	↕	↕	↕	↕
Add Accumulators	ABA													1B	1	1	A + B → A	↕	•	↕	↕	↕	↕
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	89	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3				A·M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3				B·M → B	•	•	↕	↕	R	•
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3				A·M	•	•	↕	↕	R	•
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3				B·M	•	•	↕	↕	R	•
Clear	CLR							6F	5	2	7F	5	3				00 → M	•	•	R	S	R	R
	CLRA													4F	1	1	00 → A	•	•	R	S	R	R
	CLRB													5F	1	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Accumulators	CBA													11	1	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	6	2	73	6	3				$\overline{M}$ → M	•	•	↕	↕	R	S
	COMA													43	1	1	$\overline{A}$ → A	•	•	↕	↕	R	S
	COMB													53	1	1	$\overline{B}$ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	6	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	1	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	1	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	1	1	Converts binary add of BCD characters into BCD format	•	•	↕	↕	↕	③
Decrement	DEC							6A	6	2	7A	6	3				M − 1 → M	•	•	↕	↕	④	•
	DECA													4A	1	1	A − 1 → A	•	•	↕	↕	④	•
	DECB													5A	1	1	B − 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3				A ⊕ M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3				B ⊕ M → B	•	•	↕	↕	R	•
Increment	INC							6C	6	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	1	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	1	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3				M → B	•	•	↕	↕	R	•
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3				M + 1 → B, M → A	•	•	↕	↕	R	•
Multiply Unsigned	MUL													3D	6	1	A x B → A : B	•	•	•	•	•	⑪
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → Msp, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → Msp, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	3	1	SP + 1 → SP, Msp → A	•	•	•	•	•	•
	PULB													33	3	1	SP + 1 → SP, Msp → B	•	•	•	•	•	•
Rotate Left	ROL							69	6	2	79	6	3					•	•	↕	↕	⑥	↕
	ROLA													49	1	1		•	•	↕	↕	⑥	↕
	ROLB													59	1	1		•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	6	2	76	6	3					•	•	↕	↕	⑥	↕
	RORA													46	1	1		•	•	↕	↕	⑥	↕
	RORB													56	1	1		•	•	↕	↕	⑥	↕

Note: Table 11 lists the manipulation instructions for the condition code register.

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Shift Left Arithmetic	ASL							68	6	2	78	6	3					●	●	↕	↕	⑤	↕
	ASLA													48	1	1		●	●	↕	↕	⑤	↕
	ASLB													58	1	1		●	●	↕	↕	⑤	↕
Double Shift Left, Arithmetic	ASLD													05	1	1		●	●	↕	↕	⑤	↕
Shift Right Arithmetic	ASR							67	6	2	77	6	3					●	●	↕	↕	⑥	↕
	ASRA													47	1	1		●	●	↕	↕	⑤	↕
	ASRB													57	1	1		●	●	↕	↕	⑥	↕
Shift Right Logical	LSR							64	6	2	74	6	3					●	●	R	↕	⑥	↕
	LSRA													44	1	1		●	●	R	↕	⑥	↕
	LSRB													54	1	1		●	●	R	↕	⑥	↕
Double Shift Right Logical	LSRD													04	1	1		●	●	R	↕	⑥	↕
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3				A → M	●	●	↕	↕	R	●
	STAB				D7	3	2	E7	4	2	F7	4	3				B → M	●	●	↕	↕	R	●
Store Double Accumulator	STD				DD	4	2	ED	5	2	FD	5	3				A → M B → M + 1	●	●	↕	↕	R	●
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A − M → A	●	●	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B − M → B	●	●	↕	↕	↕	↕
Double Subtract	SUBD	83	4	3	93	4	2	A3	5	2	B3	5	3				A : B − M : M + 1 → A : B	●	●	↕	↕	↕	↕
Subtract Accumulators	SBA													10	1	1	A − B → A	●	●	↕	↕	↕	↕
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A − M − C → A	●	●	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B − M − C → B	●	●	↕	↕	↕	↕
Transfer Accumulators	TAB													16	1	1	A → B	●	●	↕	↕	R	●
	TBA													17	1	1	B → A	●	●	↕	↕	R	●
Test Zero or Minus	TST							6D	4	2	7D	4	3				M − 00	●	●	↕	↕	R	R
	TSTA													4D	1	1	A − 00	●	●	↕	↕	R	R
	TSTB													5D	1	1	B − 00	●	●	↕	↕	R	R
And Immediate	AIM				71	6	3	61	7	3							M·IMM→M	●	●	↕	↕	R	●
OR Immediate	OIM				72	6	3	62	7	3							M+IMM→M	●	●	↕	↕	R	●
EOR Immediate	EIM				75	6	3	65	7	3							M⊕IMM→M	●	●	↕	↕	R	●
Test Immediate	TIM				7B	4	3	6B	5	3							M·IMM	●	●	↕	↕	R	●

New Instructions

In addition to the HD6801 instruction sets, the HD6301V has the following new instructions:

AIM ..... (M)·(IMM)→(M)

This instruction ANDs immediate data and the memory contents and then stores the result in memory.

OIM ..... (M)+(IMM)→(M)

This instruction ORs between immediate data and the memory contents and then stores the result in memory.

EIM ..... (M) + (IMM)→(M)

This instruction EORs between immediate data and the memory contents and then stores the results in memory.

TIM ..... (M)·(IMM)

This instruction ANDs between immediate data and the memory contents and then moddfies the flags of the associated condition code register. All these instructions each consist of 3 bytes. The first byte of each instruction is an op-code, the second byte is immediate data, and the third byte is the address modification field.

XGDX ..... (ACCD)↔(IX)

This instruction exchange the contents of the accumulator with those of the index register.

SLP

This instruction places the MPU in sleep mode. For details on the sleep mode, see the SLEEP MODE section.

Table 9 Manipulation instructions for the index register and stacks

Table 9 Manipulation instructions for the index register and stacks

Pointer Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED.			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	=	OP	~	=	OP	~	#	OP	~	=	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	4	3	9C	4	2	AC	5	2	BC	5	3				X−M:M−1	●	●	↕	↕	↕	↕
Decrement Index Reg	DEX													09	1	1	$X - 1 \rightarrow X$	●	●	●	↕	●	●
Decrement Stack Pntr	DES													34	1	1	$SP - 1 \rightarrow SP$	●	●	●	●	●	●
Increment Index Reg	INX													08	1	1	$X + 1 \rightarrow X$	●	●	●	↕	●	●
Increment Stack Pntr	INS													31	1	1	$SP + 1 \rightarrow SP$	●	●	●	●	●	●
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3				$M \rightarrow X_H, (M+1) \rightarrow X_L$	●	●	⑦	↕	R	●
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3				$M \rightarrow SP_H, (M+1) \rightarrow SP_L$	●	●	⑦	↕	R	●
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3				$X_H \rightarrow M, X_L \rightarrow (M+1)$	●	●	⑦	↕	R	●
Store Stack Pntr	STS				9F	4	2	AF	5	2	BF	5	3				$SP_H \rightarrow M, SP_L \rightarrow (M+1)$	●	●	⑦	↕	R	●
Index Reg → Stack Pntr	TXS													35	1	1	$X - 1 \rightarrow SP$	●	●	●	●	●	●
Stack Pntr → Index Reg	TSX													30	1	1	$SP + 1 \rightarrow X$	●	●	●	●	●	●
Add	ABX													3A	1	1	$B + X \rightarrow X$	●	●	●	●	●	●
Push Data	PSHX													3C	5	1	$X_L \rightarrow M_{sp}, SP - 1 \rightarrow SP$ $X_H \rightarrow M_{sp}, SP - 1 \rightarrow SP$	●	●	●	●	●	●
Pull Data	PULX													38	4	1	$SP + 1 \rightarrow SP, M_{sp} \rightarrow X_H$ $SP + 1 \rightarrow SP, M_{sp} \rightarrow X_L$	●	●	●	●	●	●
Exchange	XGDX													18	2	1	ACCD↔IX	●	●	●	●	●	●

Note: Table 11 lists the manipulation instructions for the condition code register.

Table 10 Jump and branch instructions

Operations	Mnemonic	Addressing Modes															Branch Test	Condition Code Register					
		RELATIVE			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	=	OP	~	=	OP	~	#	OP	~	=	OP	~	=		H	I	N	Z	V	C
Branch Always	BRA	20	3	2													None	●	●	●	●	●	●
Branch Never	BRN	21	3	2													None	●	●	●	●	●	●
Branch If Carry Clear	BCC	24	3	2													C = 0	●	●	●	●	●	●
Branch If Carry Set	BCS	25	3	2													C = 1	●	●	●	●	●	●
Branch If = Zero	BEQ	27	3	2													Z = 1	●	●	●	●	●	●
Branch If > Zero	BGE	2C	3	2													$N \oplus V = 0$	●	●	●	●	●	●
Branch If > Zero	BGT	2E	3	2													$Z + (N \oplus V) = 0$	●	●	●	●	●	●
Branch If Higher	BHI	22	3	2													C + Z = 0	●	●	●	●	●	●
Branch If < Zero	BLE	2F	3	2													$Z + (N \oplus V) = 1$	●	●	●	●	●	●
Branch If Lower Or Same	BLS	23	3	2													C + Z = 1	●	●	●	●	●	●
Branch If < Zero	BLT	2D	3	2													$N \oplus V = 1$	●	●	●	●	●	●
Branch If Minus	BMI	2B	3	2													N = 1	●	●	●	●	●	●
Branch If Not Equal Zero	BNE	26	3	2													Z = 0	●	●	●	●	●	●
Branch If Overflow Clear	BVC	28	3	2													V = 0	●	●	●	●	●	●
Branch If Overflow Set	BVS	29	3	2													V = 1	●	●	●	●	●	●
Branch If Plus	BPL	2A	3	2													N = 0	●	●	●	●	●	●
Branch To Subroutine	BSR	8D	6	2														●	●	●	●	●	●
Jump	JMP							6E	3	2	7E	3	3				See Special Operations	●	●	●	●	●	●
Jump To Subroutine	JSR				9D	5	2	AD	5	2	BD	6	3					●	●	●	●	●	●
No Operation	NOP													01	2	1	Advances Prog. Cntr. Only	●	●	●	●	●	●
Return From Interrupt	RTI													3B	10	1	See Special Operations	8					
Return From Subroutine	RTS													39	5	1		●	●	●	●	●	●
Software Interrupt	SWI													3F	12	1		●	S	●	●	●	●
Wait for Interrupt*	WAI													3E	9	1		●	9	●	●	●	●
Sleep	SLP													1A	4	1		●	●	●	●	●	●

*: WAI places R/W in "High" level; the address bus changes to FFFF; the data bus changes to the three-state level.

Note: Table 11 lists the manipulation instructions for the condition code register.

Table 11 Manipulation instructions for the condition code register

Operations	Mnemonic	Addressing Modes			Boolean Operation	Condition Code Register					
		IMPLIED				5	4	3	2	1	0
		OP	~	=		H	I	N	Z	V	C
Clear Carry	CLC	0C	1	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	1	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	1	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	1	1	1 → C	•	•	•	•	•	S
Set Interruot Mask	SEI	0F	1	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	1	1	1 → V	•	•	•	•	S	•
Accumulator A → CCR	TAP	06	1	1	A → CCR	10					
CCR → Accumulator A	TPA	07	1	1	CCR → A	•	•	•	•	•	•

Note: Each bit of the condition code register is set when the test result is true and cleared otherwise.

1 (Bit V) Test: Result=10000000?

2 (Bit C) Test: Result=00000000?

3 (Bit C) Test: Is the BCD character of the high-order byte greater than 10? If this bit is set beforehand, it is not cleared.

4 (Bit V) Test: Operand before execution=10000000?

5 (Bit V) Test: Operand before execution=01111111?

6 (Bit V) Test: Set when N+C=1 after execution of an instruction.

7. (Bit N) Test: Is the result less than 0? (Bit 15=1)

8 (All) Data is loaded from the stack to the condition code register.

9 (Bit 1) T This bit is set when an interrupt occurs. When the bit is set beforehand, to clear the set bit, a non maskable interrupt is required.

10 (all bits) All the bits are set according to the contents of accumulator A.

11 (Bit C) Bit 7 of the ACCB after multiplication=1?

Table 12 Op-code map

OP CODE						ACC A	ACC B	IND	EXT/DIR	ACCA or SP				ACCB or X				
										IMM	DIR	IND	EXT	IMM	DIR	IND	EXT	
HI		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
1000	0	⧄	SBA	BRA	TSX	NEG				SUB								0
0001	1	NOP	CBA	BRN	INS	⧄	⧄	AIM		CMP								1
0010	2	⧄	⧄	BHI	PULA	⧄	⧄	OIM		SBC								2
0011	3	⧄	⧄	BLS	PULB	COM				SUBD				ADDD				3
0100	4	LSRD	⧄	BCC	DES	LSR				AND								4
0101	5	ASLD	⧄	BCS	TXS	⧄	⧄	EIM		BIT								5
0110	6	TAP	TAB	BNE	PSHA	ROR				LDA								6
0111	7	TPA	TBA	BEQ	PSHB	ASR				⧄	STA			⧄	STA			7
1000	8	INX	XGDX	BVC	PULX	ASL				EOR								8
1001	9	DEX	DAA	BVS	RTS	ROL				ADC								9
1010	A	CLV	SLP	BPL	ABX	DEC				ORA								A
1011	B	SEV	ABA	BMI	RTI	⧄	⧄	TIM		ADD								B
1100	C	CLC	⧄	BGE	PSHX	INC				CPX				LDD				C
1101	D	SEC	⧄	BLT	MUL	TST				BSR	JSR			⧄	STD			D
1110	E	CLI	⧄	BGT	WAI	⧄	⧄	JMP		LDS				LDX				E
1111	F	SEI	⧄	BLE	SWI	CLR				⧄	STS			⧄	STX			F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

UNDEFINED OP CODE ⧄

- Only AIM, OIM, EIM, and TIM can be used.

LOW POWER CONSUMPTION MODE

The HD6301V has two low power consumption modes: sleep and standby.

- Sleep Mode

When an SLP instruction is executed, the MCU enters sleep mode. In sleep mode, the MPU operation stops but the contents of the registers in the MPU remain unchanged. Since other peripheral functions do not stop, the counter operation, data transmission, and data reception, etc., are not interrupted. In this mode, the power consumption decreases to approx. 1/10 the consumption required for the operation. Interrupts RES and STBY are used to change this mode.
RES resets the MCU and STBY brings it to standby mode. When the MPU receives RES or STBY, sleep mode is released, allowing the MPU to return to operation mode and to vector to the interrupt routine. If the MPU masks this routine, it executes the next instruction after releasing sleep mode. Since no interrupt can be issued to the MPU when the timer disables its interrupt, sleep mode cannot be released.

This sleep mode is especially effective for reducing the average power consumption when the HD63Ø1vV is not always driven.

- Standby Mode

When STBY terminal goes "Low", all the HD63Ø1V clocks stop, causing the MPU to be reset and enter standby mode. The use of this mode greatly reduces the HD6301V power consumption.

Since power is continuously supplied to the HD63Ø1V in standby mode, the RAM contents is kept. The reset start function is used to reset standby mode. The following is a typical example of applying this mode.

First, NM1 saves internal MCU information and the SP contents to the RAM, disables the RAME bit of the RAM control register, and sets the STBY bit to place the MPU in standby mode. If the STBY bit is already set at reset a start time, it indicates that power has been continuously supplied to the HD63Ø1V and the RAM contents have correctly been retained. Therefore, the system can be recovered by returning the SP and restoring the internal conditions to those before standby mode.

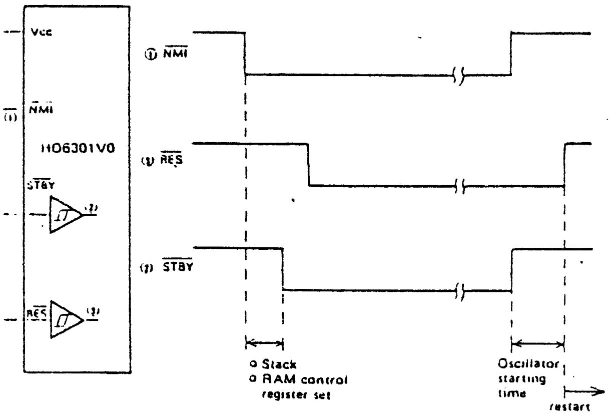


Fig. 24 shows the timing of each line (pin) in standby mode.

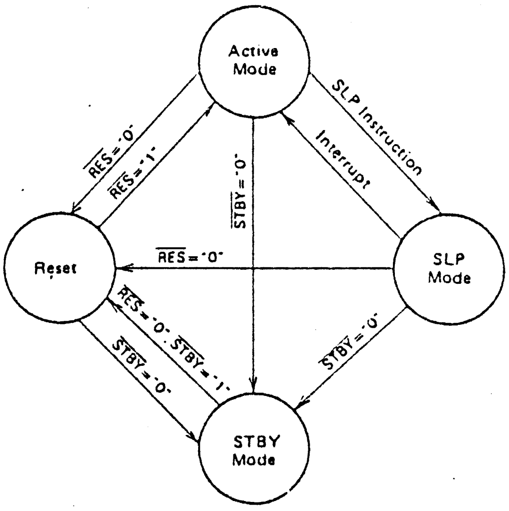


Fig. 25 Transitions between the active mode, sleep mode, standby mode and reset

ERROR PROCESSING

The HD6301V generates a highest-priority internal interrupt to prevent the system overrun that occurs due to a noise or a program error when fetching an undefined instr ction or an instruction from the unmounted memory area.

- Op-code Error

When an undefined op-code is fetched, the HD6301V saves the MPU register as in the case of a normal interrupt and vectors to the TRAP ($FFEE, $FFEF). This function has the second highest priority next to the reset function.

- Address Error

If an instruction is fetched from other than the built-in ROM, the built-in RAM, and the external memory area, execution branches to the same interrupt operation as for an op-code error. If an instruction is fetched from the external memory area when the external memory is not mounted, this function is not applied.
Table 13 shows the relationship between addresses where an address error occurs and modes.
This function applies only to the instruction fetch but does not apply to the normal data read/write access operations.

Table 13 Error addresses and modes

Mode	0	1	4	5	6	7
Address	$0000 ~ $001F	$0000 ~ $001F	$0000 ~ $001F	$0000 ~ $007F $0200 ~ $EFFF	$0000 ~ $001F	$0000 ~ $007F $0100 ~ $EFFF

Figs. 26 through 29 show the system configurations of various modes.

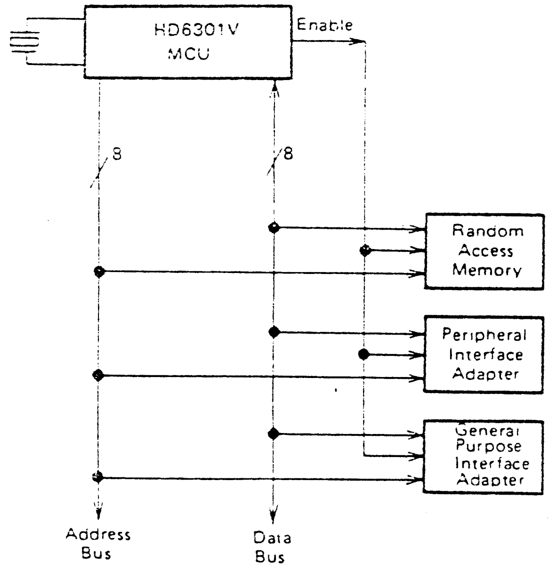

HD6301V
MCU
Enable
8
8
Random Access Memory
Peripheral Interface Adapter
General Purpose Interface Adapter
Address Bus
Data Bus

Fig. 27 HD6301V MCU expanded nonmultiplexed mode (mode 5)

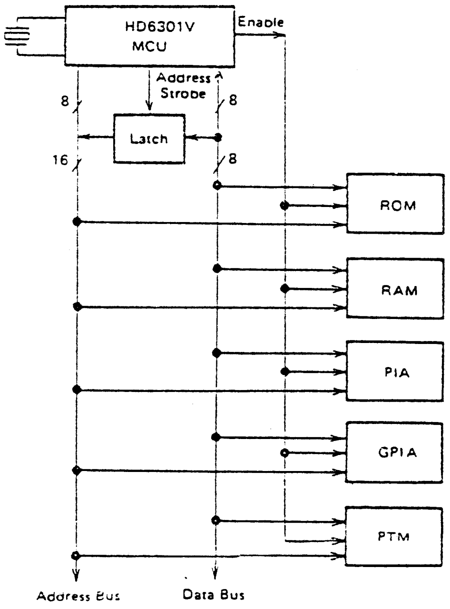


Fig. 28 HD6301V MCU expanded mutiplexed mode

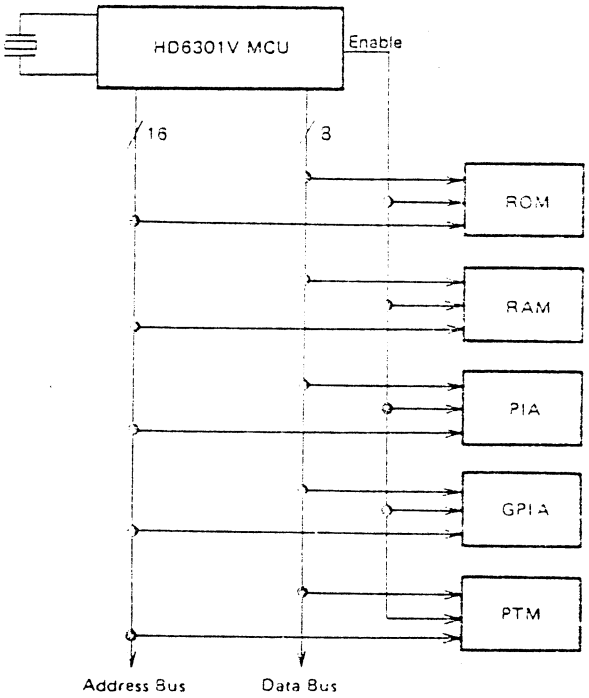


Fig. 29 HD6301V MCU expanded nonmultiplexed mode (mode 1)

4. Mnemonic Codes

Add accumulator B to accumulator A ABA

Type	Function
Arithmetic	ACCA ← (AACA) + (ACCB)
operation	Adds the ACCB contents and the ACCA contents and
(two operands)	stores the result in the ACCA.
	Effect on condition codes

$H = A_3 \ \ B_3 \cdot B_3 \cdot \overline{R_3} \cdot \overline{R_3} \cdot A_3$: When bit 3 is carried, this bit is set and otherwise cleared.

I : Not affected.

$N = R_7$: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$: When the result is 0, bit Z is set and otherwise cleared

$V = A_7 \cdot B_7 \cdot \overline{R_7} \odot \overline{A_7} \cdot \overline{B_7} \cdot R_7$: When the result overflows, bit V is set and otherwise cleared.

$C = A_7 \cdot B_7 \odot R_7 \odot R_7 \cdot A_7$: When the highest-order bit (MSB) of the result is carried, bit C is set and otherwise cleared.

Add accumulator B to indeX register ABX

Type	Function
	1X ← (1X) + (ACCB)
Arithmetic	Adds the unsigned contents of the ACCB to the
operation	index register contents considering the low-order
	byte carry of the index register and then stores the
	result in the index register.
	Effect on condition codes

H : Not affected.

Add with Carry ADC

Type	Function
Arithmetic	ACCX ← (ACCX) + (M) + (C)
operation	Adds carry bit C to a combination of the ACCX and
(two operands)	memory M and then stores the result in the ACCX.
	Effect on condition codes

$H = X_3 \cdot M_3 \cdot M_3 \cdot \overline{R_3} \cdot \overline{R_3} \cdot X_3$: When bit 3 is carried, bit H is set and otherwise cleared.

I : Not affected.

$N = R_7$: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$: When the result is 0, bit Z is set and otherwise cleared.

$V = X_7 \cdot M_7 \cdot \overline{R_7} \odot \overline{X_7} \cdot \overline{M_7} \cdot R_7$: When the result overflows. bit V is set and otherwise cleared.

$C = X_7 \cdot M_7 \odot \overline{R_7} \odot \overline{R_7} \cdot X_7$: When the highest-order bit (MSB) of the result is carried, bit C is set and otherwise cleared.

ADD without carry ADD

Type Function

Arithmetic operation (two operands)

ACCX ← (ACCX) + (M)

Adds contents of the ACCX to memory M and stores the result in the ACCX.

Effect on condition codes

H = $X_3 \cdot M_3 \odot M_3 \cdot \overline{R}_3 \odot \overline{R}_3 \cdot X_3$: When bit 3 is carried, bit H is set and otherwise cleared.

I : Not affected.

N = R7: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

Z = $\overline{R}^7 \cdot \overline{R}^6 \cdot \overline{R}^5 \cdot \overline{R}^4 \cdot \overline{R}^3 \cdot \overline{R}^2 \cdot \overline{R}^1 \cdot \overline{R}^0$: When the result is 0, bit Z is set and otherwise cleared.

V = $X_7 \cdot M_7 \cdot \overline{R}_7 \odot \overline{X}_7 \cdot \overline{M}_7 \cdot R_7$: When the result overflows, bit V is set and otherwise cleared.

C = $X_7 \cdot M_7 \odot M_7 \cdot \overline{R}_7 \odot \overline{R}_7 \cdot X_7$: When the highest-order bit (MSB) of the result is carried, bit C is set and otherwise cleared.

Double ADD without carry ADDD

Type Function

Arithmetic operation

ACCAB ← (ACCAB) + (M:M + 1)

Adds the contents of the M:M +1 to ACCAB and stores the result in the ACCAB.

Effect on condition codes

H : Not affected.

I : When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

Z = $\overline{R}_{15} \cdot \overline{R}_{14} \cdot \overline{R}_{13} \cdots\cdots \overline{R}_0$: When the result is 0, bit Z is set and otherwise cleared.

V = $AB_{15} \cdot M_{15} \odot \overline{R}_{15} \cdot \overline{AB}_{15} \cdot \overline{M}_{15} \cdot R_{15}$: When the result overflows, bit V is set and otherwise cleared.

C = $AB_{15} \cdot M_{15} \odot M_{15} \cdot \overline{R}_{15} \odot \overline{R}_{15} \cdot AB_{15}$: When the highest-order bit (MSB) of the result is carried, bit C is set and otherwise cleared.

Logical AND AND

Type Function

Logical operation

ACCX ← (ACCX) . (M)

ANDs the contents of the ACCX with those of memory M and stores the result in the ACCX.

Effect on condition codes

H : Not affected.

I : Not affected.

N = R7: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

Z = $\overline{R}_7 \cdot \overline{R}_6 \cdot \overline{R}_5 \cdot \overline{R}_4 \cdot \overline{R}_3 \cdot \overline{R}_2 \cdot \overline{R}_1 \cdot \overline{R}_0$: When the result is 0, bit Z is set and otherwise cleared.

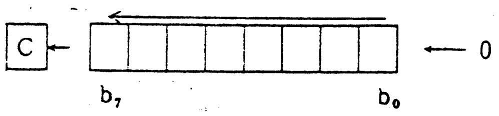
V = 0 : Bit V is cleared.

C : Not affected.

Arithmetic Shift Left ASL

Type
Shift and
rotation

Function



Shifts the ACCX or memory M to the left by one bit and stores 0 in bit 0 and ACCX or bit 7 of memory M in the carry bit.

Effect on condition codes

H : Not affected.

I : Not affected.

$N = R_7$: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$: When the result is 0, bit Z is set and otherwise cleared.

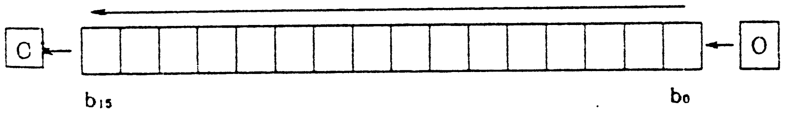
$V = N + C$: Bit V is set: When bit N is 1 and bit C is 0 after the shift or when bit N is 0 and bit C is 1 after the shift. In all other cases, this bit is cleared.
Note: Bits N and C are those after the operation.

$C = M_7$: When the highest-order bit (MSB) of the ACCX or memory before shift is 1, bit C is set and otherwise cleared.

Arithmetic Double Shift Left A:B ASLD

Type
Shift and
rotation

Function



Shifts the ACCAB to the left by one bit and stores 0 in bit 0 and bit 15 of the ACCAB in the carry bit.

Effect on condition codes

H : Not affected.

I : Not affected.

$N = R_{15}$: When the highest-order bit (MSB) of the result is 1, N is set and otherwise cleared.

$Z = \overline{R_{15}} \cdot \overline{R_{14}} \cdot \overline{R_{13}} \cdots\cdots \overline{R_0}$: When the result is 0, bit Z is set and otherwise cleared.

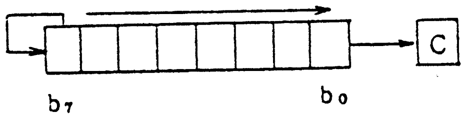
$Z = N \oplus C$: Bit V is set: When bit N is 1 and bit C is 0 after the shift or when bit N is 0 and bit C is 1 after the shift. In all other cases, this bit is cleared.
Note: Bits N and C are those after the operation.

$C = AB_{15}$: When the highest-order bit (MSB) of the ACCAB before shift is 1, bit C is set and otherwise cleared.

Arithmetic Shift Right ASR

Type
Shift and
rotation

Function



Shifts the contents of the ACCX or memory M to the right by one byte and stores bit Ø in the carry bit. In this case, bit 7 does not change.

Effect on condition codes

H : Not affected.
I : Not affected.
N = R_7: When the highest-order bit (MBS) of the result is 1, bit N is set and otherwise cleared.
Z = $\overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_Ø}$: When the result is Ø, bit Z is set and otherwise cleared.
V = N + C: Bit V is set: When N is 1 and C is Ø after the shift or when N is Ø and C is 1 after the shift.
Note: N and C are those after the operation.
C = $M_Ø$: When the lowest-order bit (LSB) before shift is 1, bit C is set and otherwise cleared.

Branch if Carry Clear BCC

Type
Conditional
branch

Function
PC ← (PC) + ØØØ2 + Rel if (C) = Ø
Judges bit C and branches when C is Ø.
Effect on condition codes
H : Not affected.

Branch if Carry Set BCS

Type
Conditional
branch

Function
PC ← (PC) + ØØØ2 + Rel if (C) = 1
Judges bit C and branches when C is 1.
Effect on condition codes
H : Not affected.

Branch if Equal BEQ

Type	Function
Conditional branch	PC ← (PC) + 0002 + Rel if (Z) = 1 Judges bit Z and branches to the specified address if Z is 1. Effect on condition codes

H : Not affected.

Branch if Greater than or Equal to zero BGE

Type	Function
	PC ← (PC) + 0002 + Rel if (N) + (V) = 0 In this case, (ACCX) ≥ (M). (Two's complement)
Conditional branch	Branches to the specified address when bits N and V are both 1 or are both 0. In other words, a branch occurs if minuend (ACCX) represented by two's complement is greater than or equal to subtrahend (M) represented by two's complement when the BGE instruction is executed immediately after execution of the CBA, CMP, SBA, and SUB instructions, etc. Effect on condition codes

H : Not affected.

Branch if Greater Than zero BGT

Type	Function
	PC ← (PC) + 0002 + Rel if (Z) . [(N) + (V)] = 0 In this case, (ACCX) > (M). (Two's complement)
Conditional branch	Branches to the specified address when bit Z is 0 and bits N and V are both 1 or are both 0. In other words, a branch occurs if minuend (ACCX) represented by two's complement is greater than subtrahend (M) represented by two's complement when the BGT instruction is executed immediately after execution of the CBA, CMP, SBA, and SUB instructions, etc. Effect on condition codes

H : Not affected.

Branch if HIgher BHI

Type

Conditional branch

Function

PC ← (PC) + 0002 + Rel if (C) ⊙ (Z) = 0

In this case, (ACCX) > (M). (Unsigned binary number)

Branches to the specified address when bits C and Z are both 0. In other words, a branch occurs if minuend (ACCX) represented by an unsigned binary number is greater than subtrahend (M) represented by an unsigned binary number when the BHI instruction is executed immediately after execution of the CBA, CMP, SBA, and SUB instructions, etc.

Effect on condition codes

H : Not affected.

BIt Test BIT

Type

Logical operation

Function

A (ACCX) . (M)

ANDs the contents of the ACCX and those of memory M and modifies the corresponding condition code. In this case, the contents of the ACCX and memory M remain unchanged.

Effect on condition codes

H : Not affected.

I : Not affected.

N = R_7: When the highest-order bit (MSB) of the AND result is 1, bit N is set and otherwise cleared.

Z = $\overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$: If all the bits are 0 after the AND operation, bit Z is set and otherwise cleared.

V = 0: Bit V is cleared.

C : Not affected.

Branch if Less than or Equal to zero BLE

Type

Conditional branch

Function

PC (PC) + 0002 + Rel if (Z) ⊙ [(N) ⊕ (V)] = 1

In this case, (ACCX) ⩽ (M) (Two's complement)

Branches to the specified address when Z is 1, N is 1 and V is 0, or N is 0 and V is 1.

In other words, a branch occurs if minuend (ACCX) represented by two's complement is less than or equal to subtrahend (M) represented by two's complement when the BLE instruction is executed immediately after execution of the CBA, CMP, SBA, and SUB instructions, etc.

Effect on condition codes

H : Not affected.

Branch if Lower or Same BLS

Type Function

PC ← (PC) + 0002 + Rel if (C) ⊙ (Z) = 1

In this case, (ACCX) ≤ (M). (Unsigned binary number)

Conditional branch

Branches to the specified address when bit C or bit Z is 1. In other words, a branch occurs if minuend (ACCX) represented by an unsigned binary number is less than or equal to subtrahend (M) represented by an unsigned binary number when the BLS instruction is executed immediately after execution of the CBA, CMP, SBA, and SUB instructions, etc.

Effect on condition codes

H : Not affected.

Branch if Less Than zero BLT

Type Function

PC ← (PC) + 0002 + Rel if (N) ⊕ (V) = 1

In this case, (Two's complement)
(ACCX) < (M).

Conditional branch

Branches to the specified address:
When N is 1 and V is 0 or when N is 0 and V is 1.
In other words, a branch occurs if minuend (ACCX) represented by two's complement is less than subtrahend (M) represented by two's complement when the BLT instruction is executed immediately after execution of the CBA, CMP, SBA, and SUB instructions, etc.

Effect on condition codes

H : Not affected.

Branch if MInus BMI

Type Function

Conditional branch

PC ← (PC) + 0002 + Rel if (N) = 1

Judges bit N and branches to the specified address when bit N is 1.

Effect on condition codes

H : Not affected.

Branch if Not Equal BNE

Type Function

Conditional branch

PC ← (PC) + 0002 + Rel if (Z) = 0

Judges bit Z and branches to the specified address when bit Z is 0.

Effect on condition codes

H : Not affected.

Branch if PLus BPL

Type	Function
Conditional branch	PC ← (PC) + 0002 + Rel if (N) = 0 Judges bit N and branches to the specified address when N is 0. Effect on condition codes

H : Not affected.

Branch Always BRA

Type	Function
Unconditional branch and jump	PC ← (PC) + 0002 + Rel Unconditionally branches to the address obtained by the above expression. Rel indicates the relative address stored in the second byte of the machine code of the branch instruction as two's complement. Effect on condition codes

H : Not affected.

BRanch Never BRN

Type	Function
Unconditional branch and jump	PC ← (PC) + 0002 The BRN instruction is treated as if NOP were used twice but it is actually a 2-byte, 3-cylce instruction. This instruction is one of the 6801 characteristics and is the reversed version of the BRA instruction. Note: 0 through $FF (possible branch range) must be specified in the second byte of the instruction code. Effect on condition codes

H : Not affected.

Branch to SubRoutine BSR

Type: Subroutine control

Function:

PC ← (PC) + 0002
1. 2 is added to the program counter.

↓ (PCL)
2. The low-order bytes of the program counter are saved to the stack.

SP ← (SP) - 0001
3. 1 is subtracted from the stack pointer.

↓ (PCH)
4. The high-order bytes of the program counter are saved to the stack.

SP ← (SP) - 0001
5. 1 is subtracted from the stack pointer.

PC ← (PC) + Rel
6. The instruction branches to the address indicated by the program.

Effect on condition codes

H : Not affected.

Branch if oVerflow Clear BVC

Type: Conditional branch

Function:

PC ← (PC) + 0002 + Rel if (V) = 0

Judges bit V and branches to the specified address when bit V is 0.

Effect on condition codes

H : Not affected.

Branch if oVerflow Set BVS

Type: Conditional branch

Function:

PC ← (PC) + 0002 + Rel if (V) = 1

Judges bit V and branches to the specified address when bit V is 1.

Effect on condition codes

H : Not affected.

Compare Accumulators CBA

Type: Comparison and Test

Function:

(ACCA) - (ACCB)

Compares the contents of the ACCA with those of the ACCB and sets an appropriate condition code. This instruction applies to the numeric and logical conditional branch. Both the operands are not affected at all.

Effect on condition codes

H : Not affected.

I : Not affected.

N = R7: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

Z = $\bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$: When the result is 0, bit Z is set and otherwise cleared.

V = $A_7 \cdot \bar{B}_7 \cdot \bar{R}_7 \cdot \bar{A}_7 \cdot B_7 \cdot R_7$: When an overflow occurs during subtraction, bit V is set and otherwise cleared.

C = $A_7 \cdot B_7 \cdot B_7 \cdot R_7 \cdot R_7 \cdot A_7$: When a borrow occurs, bit C is set and otherwise cleared.

CLear Carry CLC

Type	Function
Bit control	Bit C ← 0 Clears carry bit C Effect on condition codes

C = 0: Bit C is cleared.

CLear Interrupt mask CLI

Type	Function
Bit control	Bit I ← 0 Clears the interrupt mask bit of a condition code. The microprocessor can receive the interrupt request issued from a peripheral device. Effect on condition codes

H : Not affected.
I = 0 : Bit I is cleared.

CLeaR CLR

Type	Function
Arithmetic operation (one operand)	ACCX ← 00 or M ← 00 Clears the contents of the ACCX or memory M. Effect on condition codes

H : Not affected.
I : Not affected.
N = 0: Bit N is cleared.
Z = 1: Bit Z is set.
V = 0: Bit V is cleared.
C = 0: Bit C is cleared.

CLear two's complement oVerflow bit CLV

Type	Function
Bit control	Clears the overflow bit of a condition code. Effect on condition codes

H : Not affected.
V = 0: Bit V is cleared.
C : Not affected.

CoMPare CMP

Type
Comparison
and test

Function
(ACCX) - (M)
Compares the contents of the ACCX with those of memory M and changes the condition code. A conditional branch instruction can be used to reference the contents of this condition code. In this case, both the operands do not change.
Effect on condition codes

H : Not affected.
I : Not affected.
$N = R_7$: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.
$Z = \overline{R_7}\cdot \overline{R_6}\cdot \overline{R_5}\cdot \overline{R_4}\cdot \overline{R_3}\cdot \overline{R_2}\cdot \overline{R_1}\cdot \overline{R_0}$: When the result is 0, bit Z is set and otherwise cleared.
$V = X_7\cdot \overline{M_7}\cdot \overline{R_7} \oplus \overline{X_7}\cdot M_7\cdot R_7$: When an overflow occurs during subtraction, bit V is set and otherwise cleared.
$C = \overline{X_7}\cdot M_7 \oplus M_7\cdot R_7 \oplus R_7\cdot \overline{X_7}$: When the absolute value of memory is greater than that of the accumulator, bit C is set and otherwise cleared.

COMplement COM

Type
Logical
operation

Function
ACCX ← =(ACCX) = FF - (ACCX)
or M ← =(M) = F - (M)
Replaces the contents of the ACCX or memory M with one's complement.
Effect on condition codes

H : Not affected.
I : Not affected.
$N = R7$: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.
$Z = \overline{R_7}\cdot \overline{R_6}\cdot \overline{R_5}\cdot \overline{R_4}\cdot \overline{R_3}\cdot \overline{R_2}\cdot \overline{R_1}\cdot \overline{R_0}$: When the result is 0, bit Z is set and otherwise cleared.
$V = 0$: Bit V is cleared.
$C = 1$: Bit C is set.

ComPare indeX register CPX

Type
Index register
control

Function
(IX) - (M:M + 1)
Compares the contents of the index register with those of the 2-byte memory.
Bits N, Z, V, and C are set or cleared according to the compared result.
Effect on condition codes

H : Not affected.
I : Not affected.
N = RH: When the highest-order bit of (MSB) of the result is 1, bit N is set and otherwise cleared.
$Z = (\overline{RH_7}\ \overline{RH_6}\cdot \overline{RH_5}\cdot \overline{RH_4}\cdot \overline{RH_3}\cdot \overline{RH_2}\cdot \overline{RH_1}\cdot \overline{RH_0})\cdot (\overline{RL_7}\cdot \overline{RL_6}\cdot \overline{RL_5}\cdot \overline{RL_4}\cdot \overline{RL_3}\cdot \overline{RL_2}\cdot \overline{RL_1}\cdot \overline{RL_0})$: When the subtraction results of the high- and low-order bytes are 0, bit Z is set and otherwise cleared.

$V = IXH_7 . \overline{M7} . \overline{RH7} \odot \overline{IXH7} . M7 . RH7$: If an overflow occurs when a high-order byte subtraction is performed, bit V is set and otherwise cleared.

$C = \overline{IXH}_7 . MH7 \odot MH7 . RH7 \odot RH7 . \overline{IXH7}$: When the absolute value of memory is greater than that of the index register, bit C is set and otherwise cleared.

Decimal Adjust ACCA DAA

Type

Arithmetic operation

Function

Adds 00, 06, 60 and 66 to the ACCA (hex number) as shown in the table below when the results obtained by performing a binary coded decimal (BCD) addition with the ABA, ADD and ADC instructions, etc., stored in bits C and H of the ACCA.

Bit C state before DAA execution	4 high-order bits 4 through 7	Initial 4 bits bits 0 through 3	4 low-order bits 0 through 3	Hex. values added to the ACCX by executing DAA	Bit C state after DAA execution
0	0-9	0	0-9	00	0
0	0-8	0	A-F	06	0
0	0-9	1	0-3	06	0
0	A-F	0	0-9	60	1
0	9-F	0	A-F	66	1
0	A-F	0	0-9	66	1
1	0-2	0	0-9	60	1
1	0-2	0	A-F	66	1
1	0-3	1	0-3	66	1

Effect on condition codes

H : Not affected.

I : Not affected.

$N = R_7$: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$: When the result is 0, bit Z is set and otherwise cleared.

C : Bit C is set or cleared according to the same rules of replacing the DAA, ABA, ADD, and ADC instructions by the BCD addition (see the above table).

DECrement DEC

Type

Arithmetic operation

Function

ACCX (ACCX) - 01

or M (M) - 01

Subtracts 1 from the ACCX or memory M. In this case, bits N, Z, and V are set or cleared according to the result of the subtraction but bit C is not affected by the result at all.

Effect on condition codes

H : Not affected.

I : Not affected.

$N = R_7$: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$: When the result is 0, bit Z is set and otherwise cleared.

$V = X_7 \cdot \overline{X_6} \cdot \overline{X_5} \cdot \overline{X_4} \cdot \overline{X_3} \cdot \overline{X_2} \cdot \overline{X_1} \cdot \overline{X_0} = \overline{R_7} \cdot R_6 \cdot R_5 \cdot R_4 \cdot R_3 \cdot R_2 \cdot R_1 \cdot R_0$: If the result overflow, bit V is set and otherwise cleared. If the contents of the ACCX or memory M are 80, an overflow occurs.

C : Not affected.

DEcrement Stack pointer DES

Type
Stack
pointer
control

Function
SP ← (SP) − 0001
Substracts 1 from the stack pointer.

Effect on condition codes

H : Not affected.

DEcrement indeX register DEX

Type
Index
register
contol

Function
IX (IX) − 0001
Substracts 1 from the index register.
Only bit Z is set or cleared according to the result.
Effect on condition codes

H : Not affected.
I : Not affected.
N : Not affected.
$Z = (\overline{RH_7} \cdot \overline{RH_6} \cdot \overline{RH_5} \cdot \overline{RH_4} \cdot \overline{RH_3} \cdot \overline{RH_2} \cdot \overline{RH_1} \cdot \overline{RH_0}) \cdot (\overline{RL_7} \cdot \overline{RL_6} \cdot \overline{RL_5} \cdot \overline{RL_4} \cdot \overline{RL_3} \cdot \overline{RL_2} \cdot \overline{RL_1} \cdot \overline{RL_0})$: When the result is 0, bit Z is set and otherwise cleared.
V : Not affected.
C : Not affected.

Exclusive OR EOR

Type
Logical
operation

Function
ACCX ← (ACCX) ⊕ (M)
EORs the contents of the ACCX with those of memory M and stores the result in the ACCX.
Effect on condition codes

H : Not affected.
I : Not affected.
$N = R_7$: When the highest-order bit (MSB) of the result is 1, bit N is set ond otherwise cleared.
$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$: When the result is 0, bit Z is set and otherwise cleared.
V = 0: Bit V is cleared.
C : Not affected.

INCrement INC

Type Function

Arithmetic operation

ACCX ← (ACCX) + 01

or M ← (M) + 01

Adds 1 to the ACCX or memory M.

In this case, bits N, Z, and V are set or cleared according to the result of the addition but bit C is not affected at all by the result.

Effect on condition codes

H : Not affected.

I : Not affected.

N = R_7: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

Z = $\overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_\emptyset}$: When the result is 0, bit Z is set and otherwise cleared.

V = $\overline{X_7} \cdot X_6 \cdot X_5 \cdot X_4 \cdot X_3 \cdot X_2 \cdot X_1 \cdot X_\emptyset = R_7 \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_\emptyset}$:
When the result overflow, bit V is set and otherwise cleared.
If the contents of the ACCX or memory M are 7F, an overflow occurs.

C : Not affected.

INcrement Stack pointer INS

Type Function

Stack pointer control

SP ← (SP) + 0001

Adds 1 to the stack pointer (SP).

Effect on condition codes

H : Not affected.

INcrement indeX register INX

Type Function

Index register control

IX ← (IX) + 0001

Adds 1 to the index register.

Only bit Z is set or cleared according to the result of the addition but no other bits.

Effect on condition codes

H : Not affected.

I : Not affected.

N : Not affected.

Z =$(\overline{RH_7} \cdot \overline{RH_6} \cdot \overline{RH_5} \cdot \overline{RH_4} \cdot \overline{RH_3} \cdot \overline{RH_2} \cdot \overline{RH_1} \cdot \overline{RH_\emptyset}) \cdot (\overline{RL_7} \cdot \overline{RL_6} \cdot \overline{RL_5} \cdot \overline{RL_4} \cdot RL_3 \cdot \overline{RL_2} \cdot \overline{RL_1} \cdot \overline{RL_\emptyset})$: When bit 16 of the result is 0, bit Z is set and otherwise cleared.

V : Not affected.

C : Not affected.

JuMP JMP

Type Function

Unconditional branch and jump

PC ← numeric address

Branches to the instruction with the specified numeric address. The numeric address is specified according to the rules of the extended or index addressing.

Effect on condition codes

H : Not affected.

Jump to SubRoutine JSR

Type Function

Subroutine control

PC ← (PC) + ØØØ3 (EXTND)
or
PC ← (PC) + ØØØ2 (INDEX)
↓ (PCL)
SF ← (SP) - ØØØ1
↓ (PCH)
SP ← (SP) - ØØØ1
PC ← Numeric address

Adds 2 or 3 to the program counter in address mode, saves 2 bytes to the stack, updates the stack pointer, and branches to the specified address. The specified address is obtained according to the rules of the extended or index addressing.

Effect on condition codes

H : Not affected.

LoaD Accumulator LDA

Type Function

Load

ACCX (M)

Loads the contents of memory M to the accumulator.

Effect on condition codes

H : Not affected.

I : Not affected.

$N = R_7$: When the highest-order bit (MBS) of the result is 1, bit N is set and otherwise cleared.

$Z = \overline{R}_7 \cdot \overline{R}_6 \cdot \overline{R}_5 \cdot \overline{R}_4 \cdot \overline{R}_3 \cdot \overline{R}_2 \cdot \overline{R}_1 \cdot \overline{R}_Ø$: When the result is Ø, bit Z is set and otherwise cleared.

V = Ø: Bit V is cleared.

C : Not affected.

Double LoaD accumulator A:B LDD

Type
Load and
storage

Function
ACCB ← (M:M + 1)
Loads 2 bytes of memory M and of M+1 to the ACCAB.
Effect on condition codes

H : Not affected.

I : Not affected.

N + R_{15}: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

Z = $\overline{R}_{15} \cdot \overline{R}_{14} \cdot \overline{R}_{13} \cdot\cdot\cdot\cdot\cdot \overline{R}_{0}$: When the result is 0, bit Z is set and otherwise cleared.

V = 0: Bit V is cleared.

C : Not affected.

LoaD Stack pointer LDS

Type
Stack pointer
control

Function
SPH ← (M)
SPL ← (M + 1)
First loads the contents of memory M to the high-order 8 bits of the SP and then the contents of updated address M+1 (memory M address plus 1) to the low-order 8 bits of the SP.
Effect on condition codes

H : Not affected.

I : Not affected.

N = RH_7: When the highest-order bit (MSB) of the SP result is 1, bit N is set and otherwise cleared.

Z = $(\overline{RH}_7 \cdot \overline{RH}_6 \cdot \overline{RH}_5 \cdot \overline{RH}_4 \cdot \overline{RH}_3 \cdot \overline{RH}_2 \cdot \overline{RH}_1 \cdot \overline{RH}_0) \cdot (\overline{RL}_7 \cdot \overline{RL}_6 \cdot \overline{RL}_5 \cdot \overline{RL}_4 \cdot \overline{RL}_3 \cdot \overline{RL}_2 \cdot \overline{RL}_1 \cdot \overline{RL}_0)$: When all the bits of the SP result are 0, bit Z is set and otherwise cleared.

V = 0: Bit V is cleared.

C : Not affected.

LoaD indeX register LDX

Type
Index registe
control

Function
IXH ← (M)
IXL ← (M + 1)
First loads the contents of memory M to the high-order 8 bits of the index register and then the contents of updated address M+1 (memory M address plus 1) to the low-order 8 bits of the index register.
Effect on condition codes

H : Not affected.

I : Not affected.

N = RH_7: When the highest-order bit (MSB) of the index register is 1, bit N is set and otherwise cleared.

Z = $(\overline{RH}_7 \cdot \overline{RH}_6 \cdot \overline{RH}_5 \cdot \overline{RH}_4 \cdot \overline{RH}_3 \cdot \overline{RH}_2 \cdot \overline{RH}_1 \cdot \overline{RH}_0) \cdot (\overline{RL}_7 \cdot \overline{RL}_6 \cdot \overline{RL}_5 \cdot \overline{RL}_4 \cdot \overline{RL}_3 \cdot \overline{RL}_2 \cdot \overline{RL}_1 \cdot \overline{RL}_0)$: When all the bits of the index register are 0, bit Z is set and otherwise cleared.

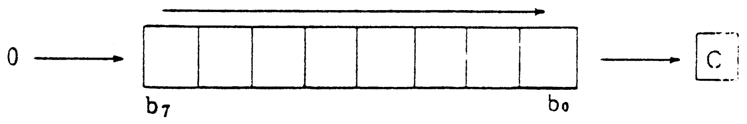
V = 0: Bit V is cleared.

C : Not affected.

Logical Shift Right   LSR

Type
Shift and
rotation

Function



Shifts the contents of the ACCX or memory M to the right by one bit, stores 0 in bit 7, loads the lowest-order bit (LSB) of the ACCX or memory M to bit C.

Effect on condition codes

H : Not affected.

I : Not affected.

N = 0: Bit N is cleared.

Z = $\overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$: When the result is 0, bit Z is set and otherwise cleared.

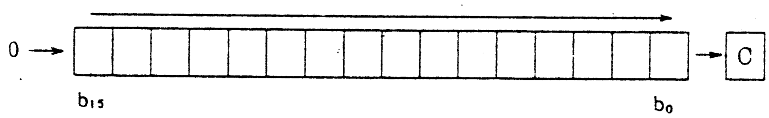
V = N + C: When N is 1 and C is 0 or N is 0 and C is 1, bit V is set and otherwise cleared.
Note: Bits N and C are those after the operation.

C = M_0: When the lowest-order bit (LSB) of the ACCX or memory M before execution of an instruction is 1, bit C is set and otherwise cleared.

Logical Double Shift Right A:B   LSRD

Type
Shift and
rotaton

Function



Shifts the contents of the ACCAB to the right by one bit, stores 0 in bit 15, and loads the lowest-order bit (LSB) of the ACCAB to bit C.

Effect on condition codes

H : Not affected.

I : Not affected.

Z = $\overline{R_{15}} \cdot \overline{R_{14}} \cdot \overline{R_{13}} \cdot \cdots \cdot \overline{R_0}$: When the result is 0, bit Z is set and otherwise cleared.

V = N + C: When bit N is 1 and bit C is 0 or bit N is 0 and bit C is 1, bit V is set and otherwise cleared.
Note: Bits N and C are those after the operation.

C = AB0: When the lowest-order bit (LSB) of the ACCAB or memory M is 1, bit C is set and otherwise cleared.

MULtiply unsigned   MUL

Type
Arithmetic
operation

Function

ACCAB ← (ACCA) * (ACCB)

Multiplies the contents of the ACCA by those of the ACCB, stores unsigned bit 16 in the ACCB according to the result, and stores the highest-order byte of the result in the ACCA.

Effect on condition codes

C = R7: When bit 7 of the result is 1, bit C is set and otherwise cleared.

NEGate NEG

Type Function
Arithmetic operation (one operand)

ACCX ← (ACCX) = OO - (ACCX)
or M ← (M) = OO - (M)
Performs two's complement for the cotents of the ACCX or memory M and stores the result in the ACCX or M. However, if the contents of the ACCX or memory M are $80(-128), no two's complement is performed.

Effect on condition codes

H : Not affected.
I : Not affected.
N = R_7: When the highest-order bit of the result is 1, bit N is set and otherwise cleared.
Z = $\overline{R_7}\cdot\overline{R_6}\cdot\overline{R_5}\cdot\overline{R_4}\cdot\overline{R_3}\cdot\overline{R_2}\cdot\overline{R_1}\cdot\overline{R_0}$: When the result is 0, bit Z is set and otherwise cleared.
V = $\overline{R_7}\cdot\overline{R_6}\cdot\overline{R_5}\cdot\overline{R_4}\cdot\overline{R_3}\cdot\overline{R_3}\cdot\overline{R_1}\cdot\overline{R_0}$: When the result overflows, bit V is set and otherwise cleared.
C = $\overline{R_7}\cdot\overline{R_6}\cdot\overline{R_5}\cdot\overline{R_4}\cdot\overline{R_3}\cdot\overline{R_2}\cdot\overline{R_1}\cdot\overline{R_0}$: When a borrow occurs, bit C is set and otherwise cleared. This bit is also set when the contents of the ACCX or memory M are not 0.

No OPeration NOP

Type Function
Unconditional branch

Updates the program counter but does not affect other registers.

Effect on condition codes

H : Not affected.

inclusive OR ORA

Type Function
Logical operation

ACCX - (ACCX)   (M)
ORs the contents of the ACCX with those of memory M and stores the result in the ACCX.

Effect on condition codes

H : Not affected.
I : Not affected.
N = R7: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.
Z = $\overline{R_7}\cdot\overline{R_6}\cdot\overline{R_5}\cdot\overline{R_4}\cdot\overline{R_3}\cdot\overline{R_2}\cdot\overline{R_1}\cdot\overline{R_0}$: When all the bits of the result are 0, bit Z is set and otherwise cleared.
V = 0: Bit V is cleared.
C : Not affected.

PuSH data onto stack PSH

Type Function
Storage ↓ (ACCX)
SP ← (SP) - 0001
Stores the contents of the ACCX in the stack indicated by the stack pointer (SP) and subtracts 1 from the SP.
Effect on condition codes
H : Not affected.

PuSH indeX register onto stack PSHX

Type Function
Storage ↓ (IXL), SP ← (SP) - 0001
↓ (IXL), SP ← (SP) - 0001
Stores the contents of the index register in the stack indicated by the SP and subtracts 2 from the SP.
Effect on condition codes
H : Not affected.

PULl data from stack PUL

Type Function
Load SP ← (SP) + 0001
↑ ACCX
Adds 1 to the SP and loads the contents of the stack indicated by the SP to the ACCX.
Effect on condition codes
H : Not affected.

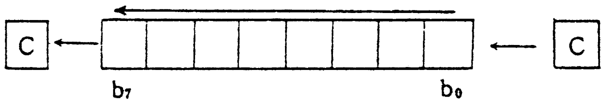
PULl indeX register from stack PULX

Type Function
Load SP ← (SP) + 0001: ↑ IXH
SP ← (SP) + 0001: ↑ IXL
Adds 1 to the SP and loads the contents of the stack indicated by the SP to the index register.
This means that the SP is totally updated twice.
Effect on condition codes
H : Not affected.

ROtate Left   ROL

Type
Shift and
rotation

Function



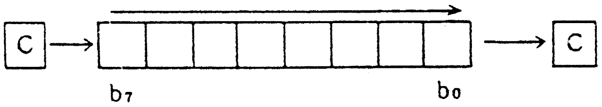
Shifts the contents of the ACCX or memory M to the left by one byte and transfers bit C to bit 0 (b0) and bit 7 (b7) to bit C.
Effect on condition codes

H : Not affected.
I : Not affected.
N = R_7: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.
Z = $\overline{R_7}\cdot\overline{R_6}\cdot\overline{R_5}\cdot\overline{R_4}\cdot\overline{R_3}\cdot\overline{R_2}\cdot\overline{R_1}\cdot\overline{R_0}$: When all bits of the results are 0, bit Z is set and otherwise cleared.
V = N $\oplus$ C: When N is 1 and C is 0 or N is 0 and C is 1 after execution of an instruction, bit V is set and otherwise cleared. Note: N and C are those after the operation.
C = M_7: When the lower-order bit (LSB) of the ACCX or memory M is 1 before execution of an instruction, bit C is set and otherwise cleared.

ROtate Right   ROR

Type
Shift and
rotation

Function



Shifts the contents of the ACCX or memory M to the right by one bit and transfers bit C to bit 7 (b7) and bit 0 (b0) to bit C.
Effect on condition codes

H : Not affected.
I : Not affected.
N = R7: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.
Z = $\overline{R_7}\cdot\overline{R_6}\cdot\overline{R_5}\cdot\overline{R_4}\cdot\overline{R_3}\cdot\overline{R_2}\cdot\overline{R_1}\cdot\overline{R_0}$: When all bits of the result are 0, bit Z is set and otherwise cleared.
V = N $\oplus$ C: When N is 1 and C is 0 or N is 0 and C is 1 after execution of an instruction, bit V is set and otherwise cleared. Note: N and C are those after the operation.
C = M_0: When the lowest-order bit (LSB) of the ACCX or memory M is 1 before execution of an instruction, bit C is set and otherwise cleared.

ReTurn from Interrupt RTI

Type	Function

Interrupt
control
	SP ← (SP) + 0001, ↑ CC
	SP ← (SP) + 0001, ↑ ACCB
	SP ← (SP) + 0001, ↑ ACCA
	SP ← (SP) + 0001, ↑ IXH
	SP ← (SP) + 0001, ↑ IXL
	SP ← (SP) + 0001, ↑ PCH
	SP ← (SP) + 0001, ↑ PCL

Stores the contents of the stack indicated by the SP to the CC, ACCB, ACCA, IXH, IXL, PCH, and PCL while adding 1 to the SP one by one. Note that if the interrupt mask bit of the CC saved to the stack is 1, I = 0 is set.

Effect on condition codes

H : The contents saved to the stack return.

ReTurn from Subroutine RTS

Type	Function
Subroutine control	SP ← (SP) + 0001 ↑ PCH SP ← (SP) + 0001 ↑ PCL

Adds 1 to the SP, sets the contents of the address indicated by the SP in the high-order 8 bits of the PC, adds 1 to the SP again, and stores the contents of the address indicated by the SP in the low-order 8 bits of the PC.

Effect on condition codes

H : Not affected.

SuBtract Accumulators SBA

Type	Function
Arithmetic operation	ACCA (ACCA) - (ACCB)

Subtracts the contents of the ACCB from those of the ACCA and stores the result in the ACCA, but the contents of the ACCB do not change.

Effect on condition codes

H : Not affected.

I : Not affected.

$N = R_7$: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$: When the result is 0, bit Z is set and otherwise cleared.

$V = A_7 \cdot \overline{B_7} \cdot \overline{R_7} \odot \overline{A_7} \cdot B_7 \cdot R_7$: When the result overflows, bit V is set and otherwise cleared.

$C = \overline{A_7} \cdot B_7 \odot B_7 \cdot R_7 \odot R_7 \cdot \overline{A_7}$: When the absolute value of the ACCB is greater than that of the ACCA, bit C is set and otherwise cleared.

SuBtract with Carry SBC

Type Function

Arithmetic operation

ACCX ← (ACCX) - (M) - (C)

Subtracts the contents of memories M and C from the ACCX and stores the result in the ACCX.

Effect on condition codes

H : Not affected.

I : Not affected.

N = R7: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

$Z = \overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$: When the result is 0, bit Z is set and otherwise cleared.

$V = X_7 \cdot \overline{M_7} \cdot \overline{R_7} \odot \overline{X_7} \cdot M_7 \cdot R_7$: When the result overflows, bit V is set and otherwise cleared.

$C = \overline{X_7} \cdot M_7 \odot M_7 \cdot R_7 \odot R_7 \cdot \overline{X_7}$: When the value obtained by combining the absolute value of memory M and that of memory C is greater than the absolute value of the ACCX, bit C is set and otherwise cleared.

SEt Carry SEC

Type Function

Bit control

C bit ← 1

Sets carry bit C of the conditions code register.

Effect on condition codes

H : Not affected.

SEt Interrupt mask SEI

Type Function

Bit control

I bit ← 1

Sets the interrupt mask bit of the condition code register.

When this bit is set, the interrupt from a peripheral device is disabled until it is cleared.

Effect on condition codes

H : Not affected.

I = 1: Bit I is set.

SEt two's complement oVerflow bit SEV

Type Function

Bit control

V bit ← 1

Sets the overflow bit of the condition code register.

Effect on condition codes

H : Not affected.

STore Accumulator STA

Type	Function
Load and storage	M (ACCX) Stores the contents of the ACCX to memory M but the contents of the ACCX do not change. Effect on condition codes

H : Not affected.

I : Not affected.

N = X_7: When the highest-order bit (MSB) of the ACCX is 1, bit N is set and otherwise cleared.

Z = $\overline{X_7}.\ \overline{X_6}.\ \overline{X_5}.\ \overline{X_4}.\ \overline{X_3}.\ \overline{X_2}.\ \overline{X_1}.\ \overline{X_0}$: When the contents of the ACCX are 0, bit Z is set and otherwise cleared.

V = 0: Bit V is cleared.

C : Not affected.

Double STore accumulator A:B STD

Type	Function
Load and storage	M : M + 1 (ACCAB) Stores the contents of the ACCAB to memories M and M+1 but the contents of the ACCAB do not change. Effect on condition codes.

H : Not affected.

I : Not affected.

N = AB_{15}: When the highest-order bit (MSB) of the ACCAB is 1, bit N is set and otherwise cleared.

Z = $\overline{AB_{15}}.\ \overline{AB_{14}}.\ \overline{AB_{13}}\ .....\ \overline{AB_0}$: When the contents of the ACCAB are 0, bit Z is set and otherwise cleared.

V = 0: Bit V is cleared.

C : Not affected.

STore Stack pointer STS

Type	Function
Stack pointer control	M (SPH) M + 1 (SPL) Stores the high-order 8 bits of the SP in memory M and then the low-order 8 bits of the SP in address M+1 (memory M address plus 1). Effect on condition codes

H : Not affected.

I : Not affected.

N = SPH_7: When the highest-order bit (MSB) of the SP is 1, bit N is set and otherwise cleared.

Z = $(\overline{SPH_7}.\ \overline{SPH_6}.\ \overline{SPH_5}.\ \overline{SPH_4}.\ \overline{SPH_3}.\ \overline{SPH_2}.\ \overline{SPH_1}.\ \overline{SPH_0}.)\quad(\overline{SPL_7}.\ \overline{SPL_6}.\ \overline{SPL_5}.\ \overline{SPL_4}.\ \overline{SPL_3}.\ \overline{SPL_2}.\ \overline{SPL_1}.\ \overline{SPL_0})$: When the contents of the SP are 0, bit Z is set and otherwise cleared.

V = 0: Bit V is cleared.

C : Not affected.

STore indeX register STX

Type: Index register control

Function:
M ← (IXH)
M + 1 ← (IXL)

Stores the high-order 8 bits of the index register in memory M and then the low-order 8 bits of the index register in address M + 1 (memory M address plus 1).

Effect on condition codes

H : Not affected.

N = IXH_7: When the highest-order bit (MSB) of the index register is 1, bit N is set and otherwise cleared.

$Z = (\overline{IXH_7} \cdot \overline{IXH_6} \cdot \overline{IXH_5} \cdot \overline{IXH_4} \cdot \overline{IXH_3} \cdot \overline{IXH_2} \cdot \overline{IXH_1} \cdot \overline{IXH_\emptyset}) \cdot (\overline{IXL_7} \cdot \overline{IXL_6} \cdot \overline{IXL_5} \cdot \overline{IXL_4} \cdot \overline{IXL_3} \cdot \overline{IXL_2} \cdot \overline{IXL_1} \cdot \overline{IXL_\emptyset})$: When the contents of the index register are Ø, bit Z is set and otherwise cleared.

V = Ø: Bit V is cleared.

C : Not affected.

SUBtract SUB

Type: Arithmetic operation

Function:
ACCX (ACCX) - (M)

Subtracts the contents of memory M from the ACCX and stores the result in the ACCX.

Effect on condition codes

H : Not affected.

I : Not affected.

N = R7: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

$Z = \overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_\emptyset}$: When the result is Ø, bit Z is set and otherwise cleared.

$V = R_7 \cdot \overline{M_7} \cdot \overline{R_7} \odot \overline{X_7} \cdot M_7 \cdot R_7$: When the result overflows, bit V is set and otherwise cleared.

$C = \overline{X_7} \cdot M_7 \odot M_7 \cdot R_7 \odot R_7 \cdot \overline{X_7}$: When the absolute value of memory M is greater than that of the ACCX, bit C is set and otherwise cleared.

Double SUBtract without carry SUBD

Type: Arithmetic operation

Function:
ACCAB (ACCAB) - (M:M + 1)

Subtracts 2 bytes of memory M from the ACCAB and stores the result in the ACCAB.

Effect on condition codes

H : Not affected.

I : Not affected.

N = R_{15}: When the highest-order bit (MSB) of the result is 1, bit N is set and otherwise cleared.

$Z = \overline{R_{15}} \cdot \overline{R_{14}} \cdot \overline{R_{13}} \cdots\cdots \overline{R_\emptyset}$: When the result is Ø, bit Z is set and otherwise cleared.

$V = AB_{15}: \overline{M_{15}} \cdot \overline{R_{15}} \cdot \odot \overline{AB_{15}} \cdot M_{15} \cdot R_{15}$: When the result overflows, bit V is set and otherwise cleared.

$C = \overline{AB_{15}} \cdot M_{15} \odot M_{15} \cdot R_{15} \odot R_{15} \cdot \overline{AB_{15}}$: When the absolute value of memory M is greater than that of the ACCAB, bit C is set and otherwise cleared.

SoftWare Interrupt SWI

Type Function

Interrupt contr PC ← (PC) + 0001

↓ (PCL), SP ← (SP) - 0001

↓ (IXL), SP ← (SP) - 0001

↓ (IXH), SP ← (SP) - 0001

↓ (ACCA), SP ← (SP) - 0001

↓ (ACCB), SP ← (SP) - 0001

↓ (CC), SP ← (SP) - 0001

I ← 1

PCH ← (highest-order address - 0005)

PCL ← (highest-order address - 0004)

Adds 1 to the program counter (PC), saves the PCL, PCH, IXL, IXH, ACCA, ACCB, AND CC to the stacks indicated by the SP in turn, subtracts 1 from the saved PCL one by one, saves the set bits 6 and 7 of a condition code, sets the interrupt mask bit, and loads the contents of two highest-order addresses -5 and -4 to the program counter.

Effect on condition codes

H : Not affected.

I = 1: Bit I is set.

Transfer from accumulator A to accumulator B TAB

Type Function

Transfer ACCB (ACCA)—

Transfers the contents of the ACCA to the ACCB but the contents of the ACCA remain unchanged.

Effect on condition codes

H : Not affected.

I : Not affected.

N = R_7: When the highest-order bit (MSB) of the ACCA is 1, bit N is set and otherwise cleared.

Z = $\overline{R_7} \cdot \overline{R_6} \cdot \overline{R_5} \cdot \overline{R_4} \cdot \overline{R_3} \cdot \overline{R_2} \cdot \overline{R_1} \cdot \overline{R_0}$: When the contents of the ACCA are 0, bit Z is set and otherwise cleared.

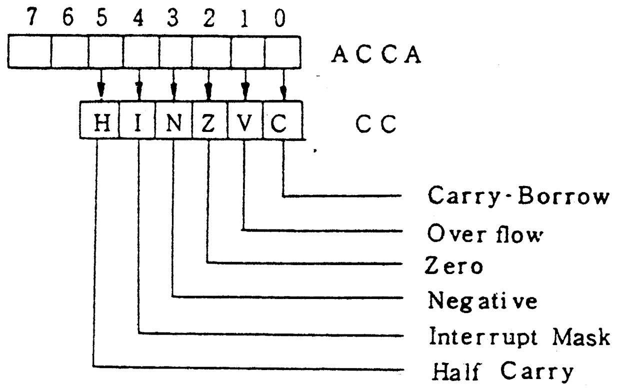
V = 0: Bit V is cleared.

C : Not affected.

Transfer from accumulator A to Processor condition codes register　　　　TAP

Type　　　　Function

Transfer　　CC ← (ACCA)



Transfers bit 0 through 5 (b_0 through b_5) of the ACCA to the corresponding bits of the condition code register but the contents of the ACCA remain unchanged.

Effect on condition codes

H : Bit 5 of the ACCA
I : Bit 4 of the ACCA
N : Bit 3 of the ACCA
Z : Bit 2 of the ACCA
V : Bit 1 of the ACCA
C : Bit 0 of the ACCA

Transfer from accumulator B to accumulator A　　　　TBA

Type　　　　Function

Transfer　　ACCA ← (ACCB)

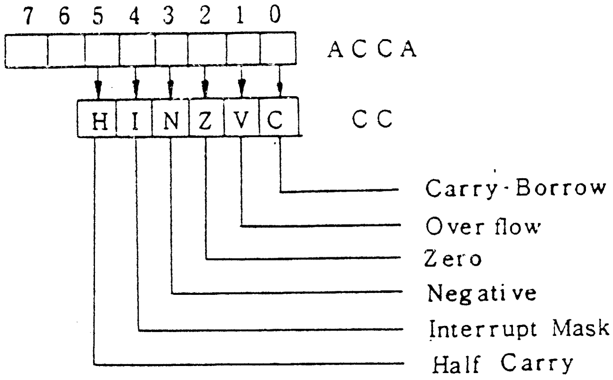
Transfers the contents of the ACCB to ACCA but the contents of the ACCB remain unchanged.

Effect on condition codes

H : Not affected.
I : Not affected.
N = R_7: When the highest-order bit (MSB) of the ACCA is 1, bit N is set and otherwise cleared.
Z = $\overline{R_7}\cdot\overline{R_6}\cdot\overline{R_5}\cdot\overline{R_4}\cdot\overline{R_3}\cdot\overline{R_2}\cdot\overline{R_1}\cdot\overline{R_0}$: When the contents of the ACCA are 0, bit Z is set and otherwise cleared.
V = 0: Bit V is cleared.
C : Not affected.

Transfer from accumulator A to Processor condition codes register   TAP

Type   Function
Transfer   CC ← (ACCA)



Transfers bits Ø through 5 of the ACCA to the corresponding bits of the condition code (CC) register but the contents of the ACCA remain unchanged.
Effect on condition codes

H : Not affected.

TeST   TST

Type   Function
Comparison and test   (ACCX) - ØØ
                    (M)   - ØØ

Sets bits N and Z of the condition code according to the contents of the ACCX or memory M.
Effect on condition codes

H : Not affected.
I : Not affected.
N = M_7: When the highest-order bit (MSB) of the ACCX or memory M is_1, bit N is_set_and_otherwise cleared.
Z = $\overline{M}_7 \cdot M_6 \cdot M_5 \cdot M_4 \cdot M_3 \cdot M_2 \cdot M_1 \cdot M_Ø$: When the contents of the ACCX or memory M are Ø, bit Z is set and otherwise cleared.
V = Ø : Bit V is cleared.
C = Ø : Bit C is cleared.

Transfer from Stack pointer to indeX register   TSX

Type   Function
Transfer   IX   (SP) + ØØØ1

Adds 1 to the stack pointer (SP) and sets the result in the index register but the contents of the SP remain unchanged.
Effect on condition codes

H : Not affected.

Transfer from indeX register to Stack pointer TXS

Type	Function
Transfer	SP ← (IX) ← 0001 Subtracts 1 from the index register and sets the result in the stack pointer (SP) but the contents of the index register remain unchanged. Effect on condition codes

Not affected.

WAIt for interrupt WAI

Type Function

Interrupt

```
PC   (PC) + 0001
↓    (PCL), SP ← (SP) - 0001
↓    (PCH), SP ← (SP) - 0001
↓    (IXL), SP ← (SP) - 0001
↓    (IXH), SP ← (SP) - 0001
↓    (ACCA), SP ← (SP) - 0001
↓    (ACCB), SP ← (SP) - 0001
↓    (CC),   SP ← (SP) - 0001
```

This instruction:

1) Adds 1 to the program counter (PC).
2) Saves the PCL, PCH, IXL, IXH, ACCA, ACCB, and CC indicated by the SP in turn.
3) Subtracts 1 from the SP one by one when the PCL is saved.
4) Transfers bits 0 through 7 (b_0 through b_7) of the CC. In this case, bits 6 and 7 (b_6 and b_7) must be set.
5) Temporarily terminates the execution program until the peripheral device issues an interrupt.
6) Sets bit I if bit I is 0 when an interrupt is issued.
7) Loads the interrupt vectoring address to the PC.

Effect on condition codes

H : Not affected.

I : Bit I is not affected until an interrupt is issued. In other words, this bit can be set if it is 0 when an interrupt is issued.

N : Not affected.

5. INSTRUCTION CODES

Mode Selection

Mode	P$_{22}$ (PC2)	P$_{21}$ (PC1)	P$_{20}$ (PC0)	ROM	RAM	Interrupt Vectors	Bus Mode	Operating Mode
7	H	H	H	I	I	I	I	Single Chip
6	H	H	L	I	I	I	MUX(5)	Multiplexed/Partial Decode
5	H	L	H	I	I	I	NMUX(5)	Non-Multiplexed/Partial Decode
4	H	L	L	I(2)	I(1)	I	I	Single Chip Test
3	L	H	H	E	E	E	MUX(4)	Multiplexed. No RAM & ROM
2	L	H	L	E	I	E	MUX(4)	Multiplexed/RAM
1	L	L	H	I	I	E	MUX(4)	Multiplexed/RAM & ROM
0	L	L	L	I	I	I(3)	MUX(4)	Multiplexed Test

Legend:
I: Internal
E: External
MUX: Multiplexed
NMUX: Non-multiplexed
L: Logic 0
H: Logic 1

Notes:

1) The internal RAM is addressed at SXX80 through SXFF.
2) The internal ROM cannot be used.
3) The reset vector is used as the external vector for 2E clock cycles after RES becomes High level.
4) In modes 0, 1, 2, and 3, the addresses related to ports 3 and 4 are used as the external addresses.
5) In modes 5 and 6, the addresses related to port 3 are used as the external addresses while the addresses related to port 4 are used as the internal addresses. 1 must be written into the data direction register when an address is output because the unnecessary addresses of port 4 can be used as an input port.
6) Only mode 2 can apply to the HD6803.

SCI bit time and transfer rates

SS1 : SS0		XTAL	2.4576MHz	4.0MHz	4.9152MHz*
		E	614.4kHz	1.0MHz	1.2288MHz
0	0	E ÷ 16	26μs/38.400Baud	16μs/62.500Baud	13.0μs/76.800Baud
0	1	E ÷ 128	208μs/4.800Baud	128μs/7812.5Baud	104.2μs/9.600Baud
1	0	E ÷ 1024	1.67ms/600Baud	1.024ms/976.6Baud	833.3μs/1.200Baud
1	1	E ÷ 4096	6.67ms/150Baud	4.096ms 244.1Baud	3.33ms/300Baud

*: 49152 MHz can apply only to the 125 MHz version.

SCI formats and clock source control

CC1. CC0		Format	Clock Source	Port2 Bit2	Port2 Bit3	Port2 Bit4
0	0	—	—	—	—	—
0	1	NRZ	Internal	Not Used	**	**
1	0	NRZ	Internal	Output*	**	**
1	1	NRZ	External	Input	**	**

*: A clock is output regardless of the status of bits RE and TE of the TRCS register.

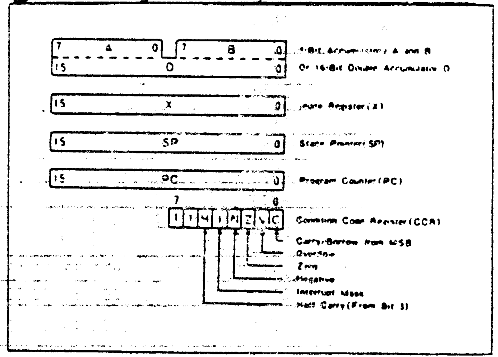
**: When bit RE of the TRCS register is 1, bit 3 is used as a serial input. When bit TE of the TRCS register is 1, bit 4 is used as a serial output.

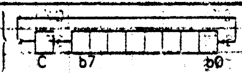
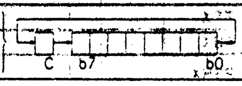
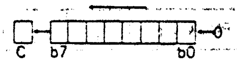
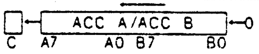
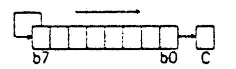
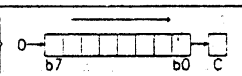
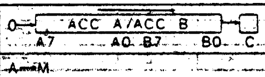
Jump & Branch Operations	Mnemonic	Addressing Modes															Branch Test	Condition Code Register					
		RELATIVE			DIRECT			INDEX			EXTND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	3	2													None	●	●	●	●	●	●
Branch Never	BRN	21	3	2													None	●	●	●	●	●	●
Branch If Carry Clear	BCC	24	3	2													C=0	●	●	●	●	●	●
Branch If Carry Set	BCS	25	3	2													C=1	●	●	●	●	●	●
Branch If =Zero	BEQ	27	3	2													Z=1	●	●	●	●	●	●
Branch If ≧Zero	BGE	2C	3	2													$N\oplus V=0$	●	●	●	●	●	●
Branch If >Zero	BGT	2E	3	2													$Z+(N\oplus V)=0$	●	●	●	●	●	●
Branch If Higher	BHI	22	3	2													C+Z=0	●	●	●	●	●	●
Branch If ≦Zero	BLE	2F	3	2													$Z+(N\oplus V)=1$	●	●	●	●	●	●
Branch If Lower Or Same	BLS	23	3	2													C+Z=1	●	●	●	●	●	●
Branch If ≦Zero	BLT	2D	3	2													$N\oplus V=1$	●	●	●	●	●	●
Branch If Minus	BMI	2B	3	2													N=1	●	●	●	●	●	●
Branch If Not Equal Zero	BNE	26	3	2													Z=0	●	●	●	●	●	●
Branch If Overflow Clear	BVC	28	3	2													V=0	●	●	●	●	●	●
Branch If Overflow Set	BVS	29	3	2													V=1	●	●	●	●	●	●
Branch If Plus	BPL	2A	3	2													N=0	●	●	●	●	●	●
Branch To Subroutine	BSR	8D	6	2														●	●	●	●	●	●
Jump	JMP							6E	3	2	7E	3	3					●	●	●	●	●	●
Jump To Subroutine	JSR				9D	5	2	AD	6	2	BD	6	3					●	●	●	●	●	●
No Operation	NOP													01	2	1	Advances Prog. Cntr. Only	●	●	●	●	●	●
Return From Interrupt	RTI													3B	10	1		⑧					
Return From Subroutine	RTS													39	5	1		●	●	●	●	●	●
Software Interrupt	SWI													3F	12	1		●	S	●	●	●	●
Wait for Interrupt	WAI													3E	9	1		●	⑨	●	●	●	●

Index Register & Stack Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	4	3	9C	5	2	AC	6	2	8C	6	3				X-M : M+1	●	●	↕	↕	↕	↕
Decrement Index Reg	DEX													09	3	1	$X-1\rightarrow X$	●	●	●	↕	●	●
Decrement Stack Pntr	DES													34	3	1	$SP-1\rightarrow SP$	●	●	●	●	●	●
Increment Index Reg	INX													08	3	1	$X+1\rightarrow X$	●	●	●	↕	●	●
Increment Stack Pntr	INS													31	3	1	$SP+1\rightarrow SP$	●	●	●	●	●	●
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3				$M\rightarrow X_H,\ (M+1)\rightarrow X_L$	●	●	⑦	↕	R	●
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3				$M\rightarrow SP_H,\ (M+1)\rightarrow SP_L$	●	●	⑦	↕	R	●
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3				$X_H\rightarrow M,\ X_L\rightarrow (M+1)$	●	●	⑦	↕	R	●
Store Stack Pntr	STS				9F	4	2	AF	5	2	BF	5	3				$SP_H\rightarrow M,\ SP_L\rightarrow (M+1)$	●	●	⑦	↕	R	●
Index Reg→Stack Pntr	TXS													35	3	1	$X-1\rightarrow SP$	●	●	●	●	●	●
Stack Pntr→Index Reg	TSX													30	3	1	$SP+1\rightarrow X$	●	●	●	●	●	●
Add B to Index Reg	ABX													3A	3	1	$B+X\rightarrow X$	●	●	●	●	●	●
Push Data	PSHX													3C	4	1	$X_L\rightarrow M_{sp},\ SP-1\rightarrow SP$ $X_H\rightarrow M_{sp},\ SP-1\rightarrow SP$	●	●	●	●	●	●
Pull Data	PULX													38	5	1	$SP+1\rightarrow SP,\ M_{sp}\rightarrow X_H$ $SP+1\rightarrow SP,\ M_{sp}\rightarrow X_L$	●	●	●	●	●	●

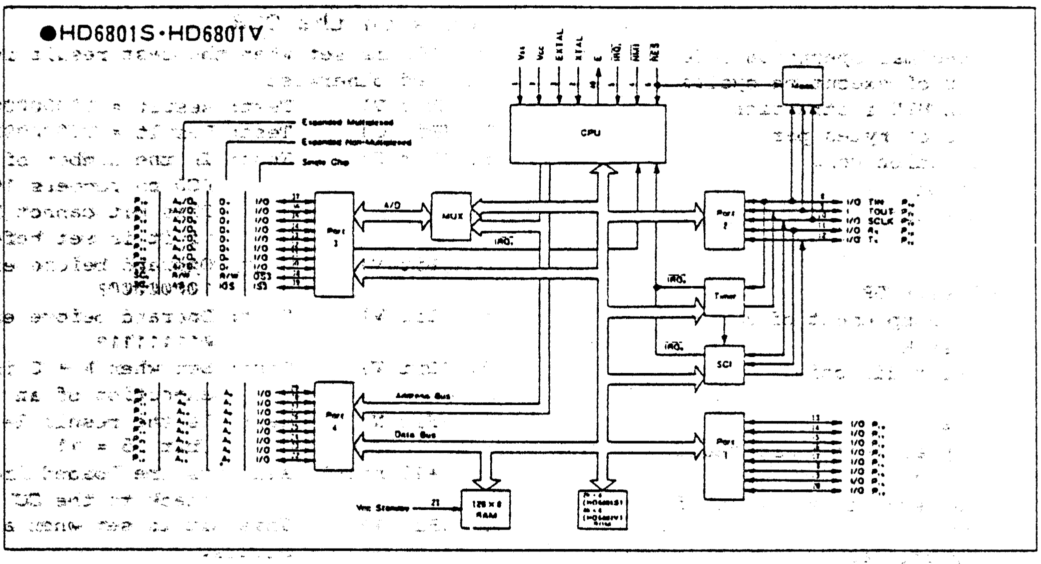
Condition Code Register Operations	Mnemonic	Addressing Modes			Boolean Operation	Condition Code Register					
		IMPLIED				5	4	3	2	1	0
		OP	~	#		H	I	N	Z	V	C
Clear Carry	CLC	0C	2	1	0→C	●	●	●	●	●	R
Clear Interrupt Mask	CLI	0E	2	1	0→I	●	R	●	●	●	●
Clear Overflow	CLV	0A	2	1	0→V	●	●	●	●	R	●
Set Carry	SEC	0D	2	1	1→C	●	●	●	●	●	S
Set Interrupt Mask	SEI	0F	2	1	1→I	●	S	●	●	●	●
Set Overflow	SEV	0B	2	1	1→V	●	●	●	●	S	●
Accumulator A→CCR	TAP	06	2	1	A→CCR	⑩					
CCR→Accumulator A	TPA	07	2	1	CCR→A	●	●	●	●	●	●

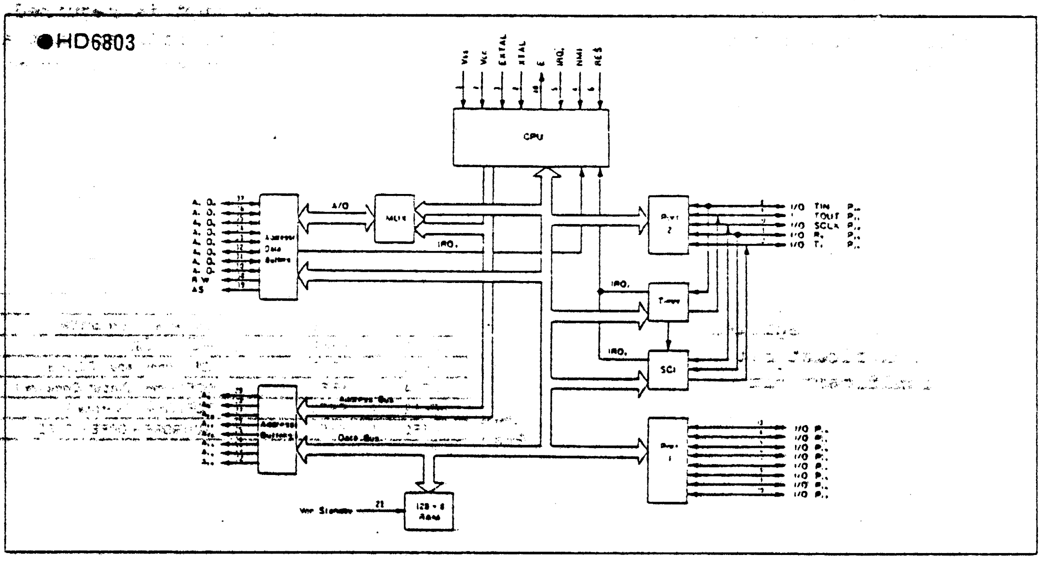
MPU Programming model



Accumulator & Memory Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3				A+M→A	↕	●	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3				B+M→B	↕	●	↕	↕	↕	↕
Add Double	ADDD	C3	4	3	D3	5	2	E3	6	2	F3	6	3				A : B+M : M+1→A : B	●	●	↕	↕	↕	↕
Add Accumulators	ABA													1B	2	1	A+B→A	↕	●	↕	↕	↕	↕
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3				A+M+C→A	↕	●	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3				B+M+C→B	↕	●	↕	↕	↕	↕
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3				A·M→A	●	●	↕	↕	R	●
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3				B·M→B	●	●	↕	↕	R	●
Bit Test	BITA	85	2	2	95	3	2	A5	4	2	B5	4	3				A·M	●	●	↕	↕	R	●
	BITB	C5	2	2	D5	3	2	E5	4	2	F5	4	3				B·M	●	●	↕	↕	R	●
Clear	CLR							6F	6	2	7F	6	3				00→M	●	●	R	S	R	R
	CLRA													4F	2	1	00→A	●	●	R	S	R	R
	CLRB													5F	2	1	00→B	●	●	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3				A-M	●	●	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3				B-M	●	●	↕	↕	↕	↕
Compare Accumulators	CBA													11	2	1	A·B	●	●	↕	↕	↕	↕
Complement.1's	COM							63	6	2	73	6	3				$\overline{M}$→M	●	●	↕	↕	R	S
	COMA													43	2	1	$\overline{A}$→A	●	●	↕	↕	R	S
	COMB													53	2	1	$\overline{B}$→B	●	●	↕	↕	R	S
Complement.2's (Negate)	NEG							60	6	2	70	6	3				00-M→M	●	●	↕	↕	①	②
	NEGA													40	2	1	00-A→A	●	●	↕	↕	①	②
	NEGB													50	2	1	00-B→B	●	●	↕	↕	①	②
Decimal Adjust.A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	●	●	↕	↕	↕	③
Decrement	DEC							6A	6	2	7A	6	3				M-1→M	●	●	↕	↕	④	●
	DECA													4A	2	1	A-1→A	●	●	↕	↕	④	●
	DECB													5A	2	1	B-1→B	●	●	↕	↕	④	●
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3				A⊕M→A	●	●	↕	↕	R	●
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3				B⊕M→B	●	●	↕	↕	R	●
Increment	INC							6C	6	2	7C	6	3				M+1→M	●	●	↕	↕	⑤	●
	INCA													4C	2	1	A+1→A	●	●	↕	↕	⑤	●
	INCB													5C	2	1	B+1→B	●	●	↕	↕	⑤	●
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3				M→A	●	●	↕	↕	R	●
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3				M→B	●	●	↕	↕	R	●
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3				M+1→B, M→A	●	●	↕	↕	R	●
Multiply Unsigned	MUL													3D	10	1	A×B→A : B	●	●	●	●	●	⑪
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3				A+M→A	●	●	↕	↕	R	●
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3				B+M→B	●	●	↕	↕	R	●
Push Data	PSHA													36	3	1	A→Msp, SP-1→SP	●	●	●	●	●	●
	PSHB													37	3	1	B→Msp, SP-1→SP	●	●	●	●	●	●
Pull Data	PULA													32	4	1	SP+1→SP, Msp→A	●	●	●	●	●	●
	PULB													33	4	1	SP+1→SP, Msp→B	●	●	●	●	●	●
Rotate Left	ROL							69	6	2	79	6	3				M  C b7 b0	●	●	↕	↕	⑥	↕
	ROLA													49	2	1	A	●	●	↕	↕	⑥	↕
	ROLB													59	2	1	B	●	●	↕	↕	⑥	↕
Rotate Right	ROR							66	6	2	76	6	3				M  C b7 b0	●	●	↕	↕	⑥	↕
	RORA													46	2	1	A	●	●	↕	↕	⑥	↕
	RORB													56	2	1	B	●	●	↕	↕	⑥	↕
Shift Left Arithmetic	ASL							68	6	2	78	6	3				M  C b7 b0	●	●	↕	↕	⑥	↕
	ASLA													48	2	1	A	●	●	↕	↕	⑥	↕
	ASLB													58	2	1	B	●	●	↕	↕	⑥	↕
Double Shift Left, Arithmetic	ASLD													05	3	1	 ACC A/ACC B C A7 A0 B7 B0	●	●	↕	↕	⑥	↕
Shift Right Arithmetic	ASR							67	6	2	77	6	3				M  b7 b0 C	●	●	↕	↕	⑥	↕
	ASRA													47	2	1	A	●	●	↕	↕	⑥	↕
	ASRB													57	2	1	B	●	●	↕	↕	⑥	↕
Shift Right Logical	LSR							64	6	2	74	6	3				M  b7 b0 C	●	●	R	↕	⑥	↕
	LSRA													44	2	1	A	●	●	R	↕	⑥	↕
	LSRB													54	2	1	B	●	●	R	↕	⑥	↕
Double Shift Right Logical	LSRD													04	3	1	 ACC A/ACC B A7 A0 B7 B0 C	●	●	R	↕	⑥	↕
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3				A→M	●	●	↕	↕	R	●
	STAB				D7	3	2	E7	4	2	F7	4	3				B→M	●	●	↕	↕	R	●
Store Double Accumulator	STD				DD	4	2	ED	5	2	FD	5	3				A→M B→M+1	●	●	↕	↕	R	●
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A-M→A	●	●	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B-M→B	●	●	↕	↕	↕	↕
Double Subtract	SUBD	83	4	3	93	5	2	A3	6	2	B3	6	3				A : B-M : M+1→A : B	●	●	↕	↕	↕	↕
Subtract Accumulators	SBA													10	2	1	A-B→A	●	●	↕	↕	↕	↕
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A-M-C→A	●	●	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B-M-C→B	●	●	↕	↕	↕	↕
Transfer Accumulators	TAB													16	2	1	A→B	●	●	↕	↕	R	●
	TBA													17	2	1	B→A	●	●	↕	↕	R	●
Test Zero or Minus	TST							6D	6	2	7D	6	3				M-00	●	●	↕	↕	R	R
	TSTA													4D	2	1	A-00	●	●	↕	↕	R	R
	TSTB													5D	2	1	B-00	●	●	↕	↕	R	R

Block Diagram


●HD6801S·HD6801V


●HD6803

Symbols:

OP : Hexadecimal operation code
~ : Number of execution cycles for an MPU instruction
# : Number of bytes per instruction word
+ : Addition
- : Subtraction
. : AND
+ : OR
+ : Exclusive OR
M : One's complement of memory address M
→ : Transfer direction
0 : Bit =
00 : Byte =
MSD : Memory address indicated by the stack pointer (SP)
H : Bit 3 carry to bit 4 (Half Carry)
I : Interrupt mask
N : Negative display
Z : Zero display
V : Overflow display for two's complement
C : Bit 7 carry or borrow
R : Always reset (= 0).
S : Always set (= 1).
: Set the bit when true after the test and otherwise cleared.
: The contents remain unchanged during execution of an instruction.
CCR : Condition code register
LS : Least significant bit
MS : Most significant bit

Notes on the CCR

The CCR is set when the test result is true and cleared otherwise.

1. (Bit V) Test: Result = 10000000?
2. (Bit C) Test: Result = 00000000?
3. (Bit C) Test: Is the number of high-order BCD characters 10? This bit cannot be cleared if it is set beforehand.
4. (Bit V) Test: Operand before execution = 10000000?
5. (Bit V) Test: Operand before execution = 01111111?
6. (Bit V) Test: Set when N + C is 1 after execution of an instruction.
7. (Bit N) Test: Is the result less than 0? (bit 15 = 1)
8. (All bits) All bits are loaded from the stack to the CCR.
9. (Bit 1) This bit is set when an interrupt occurs. To reset the set bit, a nonmaskable interrupt is required.
10. (All bits) All the bits are set according to the contents of accumulator A.
11. (Bit C) Bit 7 of the ACCB after multiplication = 1?

Memory Map for An Interrupt Vector

	Vector		Interrupt
	MSB	LSB	
Highest Priority	FFFE	FFFF	$\overline{RES}$
	FFFC	FFFD	$\overline{NMI}$
	FFFA	FFFB	Software Interrupt(SWI)
	FFF8	FFF9	$\overline{IRQ_1}$ (or $\overline{IS3}$)
	FFF6	FFF7	ICF(Timer Input Capture)
	FFF4	FFF5	OCF(Timer Output Compare)
	FFF2	FFF3	TOF(Timer Overflow)
Lowest Priority	FFF0	FFF1	SCI(RDRF + ORFE + TDRE)

Op-code map

OP CODE						ACC A	ACC B	IND	EXT	ACCA or SP				ACCB or X				
										IMM	DIR	IND	EXT	IMM	DIR	IND	EXT	
	HI	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
LO		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000	0		SBA	BRA	TSX	NEG				SUB								0
0001	1	NOP	CBA	BRN	INS					CMP								1
0010	2			BHI	PULA(+1)					SBC								2
0011	3			BLS	PULB(+1)	COM				*	SUBD(+2)			*	ADDD(+2)			3
0100	4	LSRD(+1)		BCC	DES	LSR				AND								4
0101	5	ASLD(+1)		BCS	TXS					BIT								5
0110	6	TAP	TAB	BNE	PSHA	ROR				LDA								6
0111	7	TPA	TBA	BEQ	PSHB	ASR					STA				STA			7
1000	8	INX(+1)		BVC	PULX(+2)	ASL				EOR								8
1001	9	DEX(+1)	DAA	BVS	RTS(+2)	ROL				ADC								9
1010	A	CLV		BPL	ABX	DEC				ORA								A
1011	B	SEV	ABA	BMI	RTI(+7)					ADD								B
1100	C	CLC		BGE	PSHX(+1)	INC				*	CPX(+2)			*	LDD(+1)			C
1101	D	SEC		BLT	MUL(+7)	TST				BSR (+4)	JSR(+2)			* (+1)	STD(+1)			D
1110	E	CLI		BGT	WAI(+6)	*	*	JMP(+3)		*	LDS(+1)			*	LDX(+1)			E
1111	F	SEI		BLE	SWI(+9)	CLR				* (+1)	STS(+1)			* (+1)	STX(+1)			F
BYTE/CYCLE		1/2	1/2	2/3	1/3	1/2	1/2	2/6	3/6	2/2	2/3	2/4	3/4	2/2	2/3	2/4	3/4	

	ABA DAA SEC ASL DEC SEI ASR INC SEV CBA LSR TAB CLC NEG TAP CLI NOP TBA CLR ROL TPA CLV ROR TST COM SBA	2	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
	ABX	3	1	Op Code Address	1	Op Code
			2	Op Code Address − 1	1	Irrelevant Data
			3	Address Bus FFFF	1	Low Byte of Restart Vector
	ASLD LSRD	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Irrelevant Data
			3	Address Bus FFFF	1	Low Byte of Restart Vector
	DES INS	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
			3	Previous Register Contents	1	Irrelevant Data
	INX DEX	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
			3	Address Bus FFFF	1	Low Byte of Restart Vector
	PSHA PSHB	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
			3	Stack Pointer	0	Accumulator Data
	TSX	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
			3	Stack Pointer	1	Irrelevant Data
	TXS	3	1	Op Code Address	1	Op Code
			2	Op Code Address − 1	1	Op Code of Next Instruction
			3	Address Bus FFFF	1	Low Byte of Restart Vector
	PULA PULB	4	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
			3	Stack Pointer	1	Irrelevant Data
			4	Stack Pointer + 1	1	Operand Data from Stack
	PSHX	4	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Irrelevant Data
			3	Stack Pointer	0	Index Register (Low Order Byte)
			4	Stack Pointer − 1	0	Index Register (High Order Byte)
	PULX	5	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Irrelevant Data
			3	Stack Pointer	1	Irrelevant Data
			4	Stack Pointer + 1	1	Index Register (High Order Byte)
			5	Stack Pointer + 2	1	Index Register (Low Order Byte)
IMPLIED	RTS	5	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Irrelevant Data
			3	Stack Pointer	1	Irrelevant Data
			4	Stack Pointer + 1	1	Address of Next Instruction (High Order Byte)
			5	Stack Pointer + 2	1	Address of Next Instruction (Low Order Byte)
	WAI	9	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
			3	Stack Pointer	0	Return Address (Low Order Byte)
			4	Stack Pointer − 1	0	Return Address (High Order Byte)
			5	Stack Pointer − 2	0	Index Register (Low Order Byte)
			6	Stack Pointer − 3	0	Index Register (High Order Byte)
			7	Stack Pointer − 4	0	Contents of Accumulator A
			8	Stack Pointer − 5	0	Contents of Accumulator B
			9	Stack Pointer -- 6	0	Contents of Cond. Code Register
	MUL	10	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Irrelevant Data
			3	Address Bus FFFF	1	Low Byte of Restart Vector
			4	Address Bus FFEF	1	Low Byte of Restart Vector
			5	Address Bus FFFF	1	Low Byte of Restart Vector
			6	Address Bus FFFF	1	Low Byte of Restart Vector
			7	Address Bus FFFF	1	Low Byte of Restart Vector
			8	Address Bus FFFF	1	Low Byte of Restart Vector
			9	Address Bus FFFF	1	Low Byte of Restart Vector
			10	Address Bus FFFF	1	Low Byte of Restart Vector
	RTI	10	1	Op Code Address	1	Op Code
			2	Op Code Address − 1	1	Irrelevant Data
			3	Stack Pointer	1	Irrelevant Data
			4	Stack Pointer − 1	1	Contents of Cond. Code Reg. from Stack
			5	Stack Pointer − 2	1	Contents of Accumulator B from Stack
			6	Stack Pointer − 3	1	Contents of Accumulator A from Stack
			7	Stack Pointer − 4	1	Index Register from Stack (High Order Byte)
			8	Stack Pointer − 5	1	Index Register from Stack (Low Order Byte)
			9	Stack Pointer − 6	1	Next Instruction Address from Stack (High Order Byte)
			10	Stack Pointer − 7	1	Next Instruction Address from Stack (Low Order Byte)
	SWI	12	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Irrelevant Data
			3	Stack Pointer	0	Return Address (Low Order Byte)
			4	Stack Pointer − 1	0	Return Address (High Order Byte)
			5	Stack Pointer − 2	0	Index Register (Low Order Byte)
			6	Stack Pointer − 3	0	Index Register (High Order Byte)
			7	Stack Pointer − 4	0	Contents of Accumulator A
			8	Stack Pointer − 5	0	Contents of Accumulator B
			9	Stack Pointer − 6	0	Contents of Cond. Code Register
			10	Stack Pointer − 7	1	Irrelevant Data
			11	Vector Address FFFA (Hex)	1	Address of Subroutine (High Order Byte)
			12	Vector Address FFFB (Hex)	1	Address of Subroutine (Low Order Byte)

Operation of Commands at Each Cycle

Address Mode	Instructions	Cycles	Cycle #	Address Bus	R/W Line	Data Bus
IMMEDIATE	ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	2	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Operand Data
	LDS LDX LDD	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Operand Data (High Order Byte)
			3	Op Code Address + 2	1	Operand Data (Low Order Byte)
	CPX SUBD ADDD	4	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Operand Data (High Order Byte)
			3	Op Code Address + 2	1	Operand Data (Low Order Byte)
			4	Address Bus FFFF	1	Low Byte of Restart Vector
DIRECT	ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Address of Operand
			3	Address of Operand	1	Operand Data
	STA	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Destination Address
			3	Destination Address	0	Data from Accumulator
	LDS LDX LDD	4	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Address of Operand
			3	Address of Operand	1	Operand Data (High Order Byte)
			4	Operand Address + 1	1	Operand Data (Low Order Byte)
	STS STX STD	4	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Address of Operand
			3	Address of Operand	0	Register Data (High Order Byte)
			4	Address of Operand + 1	0	Register Data (Low Order Byte)
	CPX SUBD ADDD	5	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Address of Operand
			3	Operand Address	1	Operand Data (High Order Byte)
			4	Operand Address + 1	1	Operand Data (Low Order Byte)
			5	Address Bus FFFF	1	Low Byte of Restart Vector
	JSR	5	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Irrelevant Data
			3	Subroutine Address	1	First Subroutine Op Code
			4	Stack Pointer	0	Return Address (Low Order Byte)
			5	Stack Pointer + 1	0	Return Address (High Order Byte)
INDEXED	JMP	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Offset
			3	Address Bus FFFF	1	Low Byte of Restart Vector
	ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	4	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Offset
			3	Address Bus FFFF	1	Low Byte of Restart Vector
			4	Index Register Plus Offset	1	Operand Data
	STA	4	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Offset
			3	Address Bus FFFF	1	Low Byte of Restart Vector
			4	Index Register Plus Offset	0	Operand Data
	LDS LDX LDD	5	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Offset
			3	Address Bus FFFF	1	Low Byte of Restart Vector
			4	Index Register Plus Offset	1	Operand Data (High Order Byte)
			5	Index Register Plus Offset + 1	1	Operand Data (Low Order Byte)
	STS STX STD	5	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Offset
			3	Address Bus FFFF	1	Low Byte of Restart Vector
			4	Index Register Plus Offset	0	Operand Data (High Order Byte)
			5	Index Register Plus Offset + 1	0	Operand Data (Low Order Byte)
	ASL LSR ASR NEG CLR ROL COM ROR DEC TST* INC	6	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Offset
			3	Address Bus FFFF	1	Low Byte of Restart Vector
			4	Index Register Plus Offset	1	Current Operand Data
			5	Address Bus FFFF	1	Low Byte of Restart Vector
			6	Index Register Plus Offset	0	New Operand Data
	CPX SUBD ADDD	6	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Offset
			3	Address Bus FFFF	1	Low Byte of Restart Vector
			4	Index Register + Offset	1	Operand Data (High Order Byte)
			5	Index Register + Offset + 1	1	Operand Data (Low Order Byte)
			6	Address Bus FFFF	1	Low Byte of Restart Vector
	JSR	6	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Offset
			3	Address Bus FFFF	1	Low Byte of Restart Vector
			4	Index Register + Offset	1	First Subroutine Op Code
			5	Stack Pointer	0	Return Address (Low Order Byte)
			6	Stack Pointer − 1	0	Return Address (High Order Byte)

IMPLIED	ABA DAA SEC ASL DEC SEI ASR INC SEV CBA LSR TAB CLC NEG TAP CLI NOP TBA CLR ROL TPA CLV ROR TST COM SBA	2	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
	ABX	3	1	Op Code Address	1	Op Code
			2	Op Code Address - 1	1	Irrelevant Data
			3	Address Bus FFFF	1	Low Byte of Restart Vector
	ASLD LSRD	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Irrelevant Data
			3	Address Bus FFFF	1	Low Byte of Restart Vector
	DES INS	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
			3	Previous Register Contents	1	Irrelevant Data
	INX DEX	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
			3	Address Bus FFFF	1	Low Byte of Restart Vector
	PSHA PSHB	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
			3	Stack Pointer	0	Accumulator Data
	TSX	3	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
			3	Stack Pointer	1	Irrelevant Data
	TXS	3	1	Op Code Address	1	Op Code
			2	Op Code Address - 1	1	Op Code of Next Instruction
			3	Address Bus FFFF	1	Low Byte of Restart Vector
	PULA PULB	4	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
			3	Stack Pointer	1	Irrelevant Data
			4	Stack Pointer + 1	1	Operand Data from Stack
	PSHX	4	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Irrelevant Data
			3	Stack Pointer	0	Index Register (Low Order Byte)
			4	Stack Pointer - 1	0	Index Register (High Order Byte)
	PULX	5	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Irrelevant Data
			3	Stack Pointer	1	Irrelevant Data
			4	Stack Pointer + 1	1	Index Register (High Order Byte)
			5	Stack Pointer + 2	1	Index Register (Low Order Byte)
	RTS	5	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Irrelevant Data
			3	Stack Pointer	1	Irrelevant Data
			4	Stack Pointer + 1	1	Address of Next Instruction (High Order Byte)
			5	Stack Pointer + 2	1	Address of Next Instruction (Low Order Byte)
	WAI	9	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Op Code of Next Instruction
			3	Stack Pointer	0	Return Address (Low Order Byte)
			4	Stack Pointer - 1	0	Return Address (High Order Byte)
			5	Stack Pointer - 2	0	Index Register (Low Order Byte)
			6	Stack Pointer - 3	0	Index Register (High Order Byte)
			7	Stack Pointer - 4	0	Contents of Accumulator A
			8	Stack Pointer - 5	0	Contents of Accumulator B
			9	Stack Pointer - 6	0	Contents of Cond. Code Register
	MUL	10	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Irrelevant Data
			3	Address Bus FFFF	1	Low Byte of Restart Vector
			4	Address Bus FFFF	1	Low Byte of Restart Vector
			5	Address Bus FFFF	1	Low Byte of Restart Vector
			6	Address Bus FFFF	1	Low Byte of Restart Vector
			7	Address Bus FFFF	1	Low Byte of Restart Vector
			8	Address Bus FFFF	1	Low Byte of Restart Vector
			9	Address Bus FFFF	1	Low Byte of Restart Vector
			10	Address Bus FFFF	1	Low Byte of Restart Vector
	RTI	10	1	Op Code Address	1	Op Code
			2	Op Code Address - 1	1	Irrelevant Data
			3	Stack Pointer	1	Irrelevant Data
			4	Stack Pointer - 1	1	Contents of Cond. Code Reg. from Stack
			5	Stack Pointer - 2	1	Contents of Accumulator B from Stack
			6	Stack Pointer - 3	1	Contents of Accumulator A from Stack
			7	Stack Pointer - 4	1	Index Register from Stack (High Order Byte)
			8	Stack Pointer - 5	1	Index Register from Stack (Low Order Byte)
			9	Stack Pointer - 6	1	Next Instruction Address from Stack (High Order Byte)
			10	Stack Pointer - 7	1	Next Instruction Address from Stack (Low Order Byte)
	SWI	12	1	Op Code Address	1	Op Code
			2	Op Code Address + 1	1	Irrelevant Data
			3	Stack Pointer	0	Return Address (Low Order Byte)
			4	Stack Pointer - 1	0	Return Address (High Order Byte)
			5	Stack Pointer - 2	0	Index Register (Low Order Byte)
			6	Stack Pointer - 3	0	Index Register (High Order Byte)
			7	Stack Pointer - 4	0	Contents of Accumulator A
			8	Stack Pointer - 5	0	Contents of Accumulator B
			9	Stack Pointer - 6	0	Contents of Cond. Code Register
			10	Stack Pointer - 7	1	Irrelevant Data
			11	Vector Address FFFA(Hex)	1	Address of Subroutine (High Order Byte)
			12	Vector Address FFFB(Hex)	1	Address of Subroutine (Low Order Byte)

EXTENDED	JMP	3	1	Op Code Address	1	Op Code
			2	Op Code Address+1	1	Jump Address (High Order Byte)
			3	Op Code Address+2	1	Jump Address (Low Order Byte)
	ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	4	1	Op Code Address	1	Op Code
			2	Op Code Address+1	1	Address of Operand (High Order Byte)
			3	Op Code Address+2	1	Address of Operand (Low Order Byte)
			4	Address of Operand	1	Operand Data
	STA	4	1	Op Code Address	1	Op Code
			2	Op Code Address+1	1	Destination Address (High Order Byte)
			3	Op Code Address+2	1	Destination Address (Low Order Byte)
			4	Operand Destination Address	0	Data from Accumulator
	LDS LDX LDD	5	1	Op Code Address	1	Op Code
			2	Op Code Address+1	1	Address of Operand (High order Byte)
			3	Op Code Address+2	1	Address of Operand (Low Order Byte)
			4	Address of Operand	1	Operand Data (High Order Byte)
			5	Address of Operand+1	1	Operand Data (Low Order Byte)
	STS STX STD	5	1	Op Code Address	1	Op Code
			2	Op Code Address+1	1	Address of Operand (High Order Byte)
			3	Op Code Address+2	1	Address of Operand (Low Order Byte)
			4	Address of Operand	0	Operand Data (High Order Byte)
			5	Address of Operand+1	0	Operand Data (Low Order Byte)
	ASL LSR ASR NEG CLR ROL COM ROR DEC TST* INC	6	1	Op Code Address	1	Op Code
			2	Op Code Address+1	1	Address of Operand (High Order Byte)
			3	Op Code Address+2	1	Address of Operand (Low Order Byte)
			4	Address of Operand	1	Current Operand Data
			5	Address Bus FFFF	1	Low Byte of Restart Vector
			6	Address of Operand	0	New Operand Data
	CPX SUBD ADDD	6	1	Op Code Address	1	Op Code
			2	Op Code Address+1	1	Operand Address (High Order Byte)
			3	Op Code Address+2	1	Operand Address (Low Order Byte)
			4	Operand Address	1	Operand Data (High Order Byte)
			5	Operand Address+1	1	Operand Data (Low Order Byte)
			6	Address Bus FFFF	1	Low Byte of Restart Vector
	JSR	6	1	Op Code Address	1	Op Code
			2	Op Code Address+1	1	Address of Subroutine (High Order Byte)
			3	Op Code Address+2	1	Address of Subroutine (Low Order Byte)
			4	Subroutine Starting Address	1	Op Code of Next Instruction
			5	Stack Pointer	0	Return Address (Low Order Byte)
			6	Stack Pointer−1	0	Return Address (High Order Byte)

* : The sixth cycle changes for the TST instruction as follows:
AB = FFFF, R/W = 1, DB = Low Byte of Restart Vector

**: Due to the WAIT instruction, the cycle changes in the wait state as follows:
AB = Stack Pointer-7, R/W = 1, DB = Interrupt data

RELATIVE	BCC BHI BNE BCS BLE BPL BEQ BLS BRA BGE BLT BVC BGT BMI BVS BRN	3	1	Op Code Address	1	Op Code
			2	Op Code Address+1	1	Branch Offset
			3	Address Bus FFFF	1	Low Byte of Restart Vector
	BSR	6	1	Op Code Address	1	Op Code
			2	Op Code Address−1	1	Branch Offset
			3	Address Bus FFFF	1	Low Byte of Restart Vector
			4	Subroutine Starting Address	1	Op Code of Next Instruction
			5	Stack Pointer	0	Return Address (Low Order Byte)
			6	Stack Pointer−1	0	Return Address (High Order Byte)

This manual contains parts of the following materials:

- Hitachi Microcomputer System HMCS6800
- HD6801 User Manual published by Hitachi
- HD6301V MCU Temporary Specifications published by Hitachi (May 1981)
- Instruction Pocket Book published by Hitachi
- HD146818 RTC Temporary Specifications published by Hitachi
- μPD7227G Specifications published by NEC