EPSON

Copyright:
EPSON Deutschland GmbH
Am Seestern 24
D-4000 Düsseldorf
Tel.: 02 11/5 96 10 01

HAND-HELD-COMPUTER

HX-20 Software

ASSEMBLER
DISASSEMBLER

EPSON

I. Allgemeines zum Programm

Das Assembler-Programm zum HX-20 ermöglicht die direkte Eingabe der Mnemonics über die Tastatur. Die Eingabe erfolgt durch ein BASIC-Programm und ein Binär-Programm.

Direkt mit der Mnemonic Eingabe wird disassembliert, d.h. die Befehle werden im OP-Code angezeigt.

Zur Eingabe werden nur die Datentypen: DEZ (ohne Prefix), $HEX, ` ASCII, % Binär zugelassen.

Mit HELP können die Assembler-Direktiven angezeigt werden.

Nach der Erfassung besteht die Möglichkeit, das erstellte HEX-File Programm automatisch auf der Kassette zu sichern.

II. Programmdaten in Stichworten

- Test und Definitionen MEMSET
- Definition Procedure NAME+ASEG
- Warten auf Kommando
- aktivieren Assembler
- dekodieren der Adressen Arten
- dekodieren der Anweisungen
- SUBROUTINES
- Umwandlung der Datenformate
- Adressen Mode Select
- Zeichen ab "SP" isolieren
- asymetrische Adressen modifiziert dekodieren
- restaurieren der Führungs-Null
- Branch-Offset berechnen (Assembler)
- select 3 byte Instruction
- 4 Stellen (incl. Vornullen) anzeigen
- indirect adressierte Detektion (Dis)
- Dezimal _65535 in WORD-Format
- ERROR + zugegriffene Ziffer ausgeben
- Ausdruck formatieren
- Offset auf Drucker mit relativem Wert
- Dez-Bereich sichern
- Hex-Bereich einhalten!
- außerhalb Systemgrenzen Zahl Instruction byte=0 halten! (DIS)
- Darstellbare Zeichen isolieren (DIS)
- vorbereiten Parameter-Übergabe an/von Maschinenprogramm (ASS)
- Branch berechnen (DIS)
- Vornullen
- Test System-Grenzen (_$4E_MEMSET)
- Test ob Name eingegeben + Datensicherung erwartet wird
- Dialog-Test: EDIT
-     - LIST
- DUMP modifiziertes MEMORY
- VERIFY
- nur LCD Ausgabe
- Aussprung Monitor (regelmäßig)

III   Programmbeschreibung

Das Assembler Programm beinhaltet ein HEX-File Programm (LINA3.HEX) und ein BASIC-Programm (LINA3.BAS)

Programme wie folgt laden:

HX-20 einschalten

Funktion "2" BASIC anwählen

MEMSET auf _= &H1093 setzen

Programmkassette Assembler einlegen

Programm laden :

```
LOADM"CAS0:LINA3.HEX"

RUN"CAS0:LINA3.BAS"
```

Handhabung und Benutzerführung:

Folgende Datentypen sind zugelassen:

- DEZ (ohne Prefix) $HEX 'ASCII % Binär

Bereich Procedure Definition

- MEMSET? (CR) - RETURN - angezeigter Wert unverändert

- MEMSET? (ADR)(CR) - neue HEX-Adresse eingeben - RETURN -

- NAME ? (CR) - RETURN - Sprung zu ASEG-ORG

- NAME ? (nnn..) - Eingabe für Procedure - Name

- HEX-FILE ? (CR) - RETURN - keine Datei-Sicherung

- HEX-FILE ? (Y)(CR) - Y-RETURN - Sicherung mit Namen

- ASEG-ORG (ADR)(CR) - ADR -RETURN- Procedure Start-Adresse

- ASEG-ORG (CR) - RETURN - nur zulässig, wenn Adresse bereits vorhanden. Diese wird im laufenden Befehls-Zähler eingesetzt.

- ORG (CR) - Befehlslzähler auf letzten aktuellen ORG Wert

- ENDP (CR) - ersetzt TOP - Befehlszähler auf ORG Wert im Procedure-Definitions-Bereich (erstes ORG)

- PROC - Adressen - erste freie Position neu festsetzen.

- MON - Auśsprung Monitor - Ass.Parameter bleiben erhalten - zurück "B"(Back-Kommando)

- LIST (CR) - gibt disassembler OP-Code in den Grenzen ASEG...ENDP aus

- END (CR):
- keine Datensicherung vorbereitet, kein Listing, dann LCD Ausgabe der Procedure grenzen (ASEG/NEXT)
- Datensicherung ,aktivieren des Cassetten drive.
- Listing, Ausdruck der dokumentierten Bytes (total) + der modifizierten (per Assembler) + Adresse der nächsten Instruktion

- Help - Anzeige der Assembler Direktionen - Rückkehr zur laufenden Zeile
- WORD-Bereich Überschreitung wird abgefangen (ERR235)

- VARIABLE (W=WORK//restliche global)

- A	W
- AO	W
- A2	1. oder einziges op-code Folgebyte
- A3	2.Folgebyte
- A4	byte für immed/mem adresse (aim,tim...)
- A$	Tastatur-String
- AO$	isoliertes Zahlenargument
- A1$	dito Anweisung-Mnemo
- A2$	Hex-String von A3
- A3$	dito von a3
- A4$	dito
- B	aktueller Befehls-Zähler (mit Rundg.Korr)
- B$	dito in Hex
- C	gen. Opcode
- C$	dito in Hex
- D	Schleifenwert aus Adress-Schlüssel(1-8 zulässig)
- D3	Offset aus Branch
- D$	Suchstring aus Tabelle isoliert
- D1$	Adress-Schlüssel (Disassabler)
- D2$	1. Folgebyte
- D3$	2. Folgebyte
- D4$	immed. Adr (hält "£")
- D5$	Mnemoteil des Ausdruckes
- D6$	Summenstring (für Print)
- D7$	ind.Adr. (hält ".x")
- D8$	immed. Memory (hält ",") nur AIM,OIM,TIM,EIM
- DE	dez.Wert von X$
- DE$	
- E$	Adressen Modification
- ER	laufender Fehlercode
- F	Flag (8=Rückkehr zur Eingabe ohne weitere Verarbeitung)
- FO	Wert der 1.Org.-Anweisung (ASEG)
- G	Flag (Rücksprung aus Monitor)
- I	Laufvariable
- Gl	direkt FIND (unbenutzt)
- K	unbenutzt
- L	Länge String vor Convert
- LI	Adressen erster freier Speicherplatz
- LO	Adressen laufende ORG-Anweisung
- M	modifizierender Wert MEMSET
- M$	IST-Wert MEMSET
- N	Zahl (dez) ausgeführten Modifikationen (Bytes)
- O	Instruktions-Byte (Disassambler)
- P	Flag (1=edit 2=list 0=LCD
- S	W
- T$	Proedure-Name für Protokoll
- X$	Zahlenwert von Konvertierung //W
- Y	Basis des Arguments
- Y$	Flag (Y$=Y _Cassette in Position

Programm HX-20 Disassembler

Dieses BASIC-Programm disassembliert den Bereich von der eingegebenen Startadresse bis zur Endadresse.

Programm laden:

```
RUN"CAS0:DISASSEM.BAS"
```

Eingabe Startadresse - Dezimal (CR)

Eingabe Endadresse - Dezimal (CR)

```
DOCUMENT OF USING I/O SUBROUTINE

DATE          07.02.1981
AUTHOR        KENJI AKAHNE
REVISION      C-1
UPDATE        10.15.1982
FILE NAME    MAINIO HXlD

NOTE.                                                                  %
 UPDATE MARK = 72 COLUMN '%'    (TO REVISION B).                       %
 IN 'ON ENTRY' OR 'ON EXIT' NOTATION. (A) MEANS REGISTER A. (B) MEANS  %
REGSTER B, (C) MEANS CARRY BIT, (Z) MEANS Z FLAG, (X) MEANS INDEX      %
REGISTER.                                                              %
```

```
(1) RS232
1. RS232C SUBROUTINE
  1. MODE SET
     NAME   RSMST
     DATA SET
      (A) MODE    0,1: STOP BITS  1:1   2:2
                  2:   CARRIER DETECT  0:CHECK 1:IGNORE
                  3:   RTS    0:LOW   1:HIGH
                  4:   DSR    0:CHECK   1:IGNORED
                  5:   CTS    0:CHECK   1:IGNORED
                  7,6: PARITY  00:EVEN  10:ODD    11:NONE
      (B) BIT LENGTH AND MODE
          LSB 4 BITS:BIT LENGTH (5, 6, 7 OR 8)
          MSB 4 BITS:BAUD 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 (NOT AVAILABLE)
    REGISTER PRESERVE   A,B,X

2. RS232C POWER ON/OFF
  1. POWER ON OR POWER OFF
     NAME  RSONOF
      PARAMETER
      ON ENTRY (A): 0:OFF   1:ON   (IF NOT 0, TREATED AS 1)
      ON EXIT  (C): 0:NORMAL ON      1:ERROR
               (A): 0:NORMAL   OTHERS:ERROR
               (Z): DEPEND ON VALUE OF (A)
    REGISTER PRESERVE  B,X

3. RS232C OPEN/CLOSE
  1. OPEN
     NAME  RSOPN
     PARAMETER
               (X): BUFFER ADDRESS
               (A,B): BUFFER SIZE (BYTES) (FROM 01 TO ?)
      RESULT   (C):   0:NORMAL     1:I/O ERROR
               (A): ERROR CODE  0:OK    1:DRIVER POWER OFF
               (Z): SET DEPEND ON (A)
    REGISTER PRESERVE  B,X

  2. CLOSE
    NAME  RSCLOS
   ON ENTRY PARAMETER NONE
   ON EXIT    (C):   0:NORMAL    1:I/O ERROR
              (A):   0:NORMAL   OTHERS:ERROR
              (Z):   DEPEND ON VALUE OF (A)
   REGISTER PRESERVE   B,X
```

9

```
4. TO SEND/RECEIVE
  1. RECEIVE ONE CHARACTER
    DESCRIPTION
      GET ONE CHARACTER FORM RS232C READ BUFFER. THIS ROUTINE DO NOT
     WAIT TO BE RECIVED DATA ALTHOUGH BUFFER IS EMPTY.
    NAME  RSGET
    PARAMETER  NONE
    RESULT    (A):WHEN RECEIVED     (B):STATUS
                                    ¥00:RECEIVED
                                    ¥01:BUFFER IS EMPTY
                                    MSB=1: ERROR
                                    BIT 0 - 6:ERROR CODE
              (C):   0:NORMAL   1:I/O ERROR
              (Z): DEPEND ON VALUE OF (B)
     REGISTER PRESERVE   X

  2.  SEND ONE CHARACTER
     NAME  RSOUT
     PARAMETER
        (A):DATA
     RESULT
        (B):STATUS    BIT 0  (1: DSR OFF)
                      BIT 1  (1: CTS OFF)
                      BIT 2 - 7 (ALWAYS 0)
        (Z):DEPEND ON VALUE OF (B)
     REGISTER PRESERVE  X,A
     NOTE. IF (B) IS NOT ZERO, TRANSMITTED DATA WAS NOT TRANSMITTED.


   PROCEDURE OF USING RS232C

   1. SET RS232C MODE   (CALL 'RSMST')
   2. DRIVER ON
   3. OPEN (FOR READ)
   4. SEND/RECEIVE
   5. CLOSE

    AFTER OPENED TO READ, IF WE CHANGE 'RS232C MODE' (BIT RATE ..ETC),
   TRANSMITTED DATA MODE WILL CHANGE, BUT RECEIVED DATA MODE WILL
   NOT CHANGE.
    YOU DON'T NEED TO OPEN OR CLOSE IF YOU DO NOT RECEIVED DATA.
    VALUE OF 'RTS' IS SET BY OPEN ROUTINE.
    WHEN SLAVE DEVICE(PRINTER, CASSETTE OR SPEEKER) AND SERIAL
   COMMUNICATION ARE WORKING, RS232C RECEIVED DATA ARE LOST.

 AFTER EXECUTED 'OPEN RS232C' ROUTINE, RECEIVED DATA IS PUSHED INTO
 RS232C DATA STACK IN THE MAIN MEMORY.
```

```
(2) SPEAKER
;1. BEEP FOR KEY ACCEPT (DELETED)
;   NAME  KEYBEP
;  ON ENTRY PARAMETER  NONE
;  ON EXIT    (C):  0:ERROR  1:NORMAL
;  REGISTER PRESERVE   A,B,X

  2. SOUND BY MUSICAL SCALE AND TIME
     NAME  SOUND
    PARAMETER   (A):MUSICAL SCALE
                     0:PAUSE
                     1:DO  2:RE  3:MI 4:FA 5:SO 6:RA (440 HZ)..
                     ..  28:TI
                     29:#(2) (VALUE OF 2) 30:#3  31:#4 .....
                     56:#28
                  (B):TIME 1=0.1 SEC (0 - 255) 0:NOTHING
    ON EXIT     (C):  0:NORMAL   1:ERROR
    REGISTER PRESERVE  A,B,X

;3. BEEP BY FREQUENCY AND TIME  (DELETED)
;   NAME   BEEP
;  PARAMETER    (X):ADDREESS OF FREQENCY AND TIME DATA
;                 DATA  BYTE0: FREQUENCY (H)
;                       BYTE1: FREQUENCY (L)
;                       BYTE2: TIME (H)
;                       BYTE3: TIME (L)
;            FREQUENCY: 1=1.6 MICRO SEC  880 HZ= 1000000/880/1.6/2
;            TIME:      1=1.6*256 MICRO SEC
;  ON EXIT    (C):  0:NORMAL  1:ERROR

   NOTE. KEY BEEP, AND BEEP BY FREQUENCY ROUTINE ARE ERASED
```

```
(3) PRINTER
 1. PRINT ONE CHARACTER TO THE INTERNAL MICRO PRINTER.
    NAME  CHPRNT
    PARAMETER   (A): CHARACTER CODE (ASCII CODE)
    ON EXIT   (C):  0:NORMAL   1:I/O ERROR
    REGISTER PRESERVE  A,B,X

    NOTE. CHARACTERS ARE ACTUALY PRINTED ON RECEIVE 'LF' (¥0A) OR
     OVER 24 CHARACTERS OF PRINT BUFFER.
     EFFECTIVE CONTROL CODES ARE ¥0D(CR:MOVE COLUMN POSITION TO THE
     TOP OF THE LINE BUFFER) AND ¥0A(LF:PRINT)

 2. PRINT ONE LINE  (CHARACTERS) TO HTE MICRO PRINTER.
    NAME  LNPRNT
    PARAMETER   (X): BUFFER ADDRESS (BUFFER SIZE = 24 BYTES)
    ON EXIT   (C):  0:NORMAL   1:I/O ERROR
    REGISTER PRESERVE  A,B,X

 3. PRINT ONE DOT LINE (GRAPHIC IMAGE) TO MICRO PRINTER.
    NAME  PRTDOT
    PARAMETER   (X): BUFFER ADDRESS (BUFFER SIZE = 24 BYTES)
    ON EXIT   (C):  0:NORMAL   1:I/O ERROR
    REGISTER PRESERVE  A,B,X

    TYPE OF DATA
     BYTE 0: FIRST 6 DOTS GRAPH IMAGE
     BYTE 1: SECOND 6 DOT GRAPH IMAGE
       :
     BYTE 23: 24 TH 6 DOT GRAPH IMAGE

     EACH 6 DOTS:
        BIT 0: FIRST DOT
        BIT 1: SECOND DOT
          :
        BIT 5: 6 TH DOT
        BIT 6, BIT7 : NOT USED

 4. SCREEN COPY
    NAME  SCRCPY
    PARAMETER  NONE
    ON EXIT   (C):  0:NORMAL   1:I/O ERROR
    REGISTER PRESERVE  A,B,X

 5. PAPER FEED
    NAME  PAPFED
    PARAMETER  (A): FEED DOT LINES (LIMIT 1 - 255)
    ON EXIT   (C):  0:NORMAL   1:I/O ERROR
    REGISTER PRESERVE   A,B,X
```

```
(4) EXTERNAL CASSETTE

  4.1  CASSETTE REMOTE  ON/OFF
       ONLY ON/OFF CASSETTE REMOTE SWITCH
       NAME
         PONFCS
       PARAMETER
        ON ENTRY
         (A):    0:OFF  1:ON
        ON EXIT
         (C):  0:NORMAL   1:ERROR

  4.2  OPEN TO READ
       SEARCH TARGET FILE AND OPEN READ CASETTE FILE AND WITHOUT
       ANSWERING FILE NAME
       NAME
         OPNRCS
       PARAMETER
         (X): PACKET ADDRESS
       PACKET: BYTE 0: READ MODE
                       00:STOP EACH BLOCK
                       01:NON STOP
                       FF:DEPEND ON THE HEADER
               BYTE 1: BUFFER ADDRESS (HIGH)
               BYTE 2: BUFFER ADDRESS (LOW)
                        BUFFER SIZE = 256 BYTES + 4 BYTES
               BYTE 3 - BYTE 10: FILE NAME
               BYTE 11- BYTE 18: FILE TYPE
       RESULT
         (A): RETURN CODE
              00: NORMAL    ELSE ERROR
         (C): I/O ERROR FLAG
              0:NORMAL    1:ERROR
         (Z): DEPEND ON VALUE OF (A)

  4.3  READ OPEN WITH ANSWER FILE NAME
       SEARCH FIRST FILE, WHEN FOUND FILE, IF FILE IS TARGET FILE,
       OPEN FOR READ. IF NOT TARGET FILE, ONLY ANSWER FILE NAME.
       NAME
         SRCRCS
       PARAMETER
         (X): PACKET ADDRESS
       PACKET: BYTE 0: READ MODE
                       00:STOP EACH BLOCK
                       01:NON STOP
                       FF:DEPEND ON HEADER
               BYTE 1: BUFFER ADDRESS (HIGH)
               BYTE 2: BUFFER ADDRESS (LOW)
                        BUFFER SIZE = 256 BYTES + 4 BYTES
               BYTE 3 - BYTE 10: FILE NAME
               BYTE 11- BYTE 18: FILE TYPE
               BYTE 19- BYTE 26: FOUND FILE NAME(FILLED WHEN RETURNED)
               BYTE 27- BYTE 34: FOUND FILE TYPE(FILLED WHEN RETURNED)
       RESULT
         (A): RETURN CODE
              00: NORMAL    ¥80:FOUND OTHER FILE (SKIPPED)
                  OTHERS: ERROR
         (C): I/O ERROR FLAG
              0:NORMAL    1:ERROR
         (Z): DEPEND ON VALUE OF (A)
```

```
   NOTE.  ON OPEN TO READ CASSETTE ROUTINE, IN FILE NAME OR IN FILE
    TYPE, '*' CHARACTER IS DETECTED, MATCHING OF 'FILE NAME' OR
    'FILE TYPE' IS TERMANATED. '*' CHARACTER MAY BE DECLARED IN
    'FILE NAME' 'FILE TYPE' EACH.


;  4.4  SCAN AND MOVE EOF (DELETED)
;       NAME
;         SCANCS
;       PARAMETER
;         (X): PACKET ADDRESS
;       PACKET: BYTE 0: BUFFER ADDRESS (HIGH)
;               BYTE 1: BUFFER ADDRESS (LOW)
;                        BUFFER SIZE = 256 BYTES + 4 BYTES
;               BYTE 2- BYTE 9: FOUND FILE NAME (SET AFTER CALLED)
;               BYTF 10- BYTE 17: FOUND FILE TYPE (SET AFTER CALLED)
;       RESULT
;         (A): RETURN CODE
;              00: NORMAL    OTHERS:ERROR
;         (C): I/O ERROR FLAG
;               0:NORMAL    1:ERROR

   4.5  READ ONE CHARACTER
        NAME
          READCS
        PARAMETER
          INPUT: NONE
        RESULT
          (A):CHARACTER CODE
          (B):ERROR CODE
               00: NORMAL    ELSE ERROR
          (Z): SET DEPEND ON (B)
          (C): I/O ERROR FLAG
               0:NORMAL   1:ERROR
        REGISTER PRESERVE  X

   4.6  CLOSE
        NAME
          CLOSCS
        PARAMETER NONE
        RESULT
          (C): I/O ERROR FLAG      0:NORMAL    1:ERROR
```

```
4.7  OPEN TO WRITE
     NAME
       OPNWCS
     PARAMETER
       (X): PACKET ADDRESS
     PACKET: BYTE 0: WRITE MODE
                     00:STOP EACH BLOCK
                     01:NON STOP
             BYTE 1: BUFFER ADDRESS (HIGH)
             BYTE 2: BUFFER ADDRESS (LOW)
                      BUFFER SIZE = 256 BYTES + 4 BYTES
             BYTE 3 - BYTE 10: FILE NAME
             BYTE 11- BYTE 18: FILE TYPE
     RESULT
       (A): RETURN CODE
            00: NORMAL    ELSE ERROR
       (Z): SET DEPEND ON VALUE OF (A)
       (C): I/O ERROR FLAG    0:NORMAL   1:ERROR

4.8  WRITE ONE CHARACTER
     NAME
       WRITCS
     PARAMETER
       (A): WRITE CHARACTER
     RESULT
       (B): RETURN CODE
            00: NORMAL    ELSE ERROR
       (Z): SET DEPEND ON VALUE OF (B)
       (C): I/O ERROR STATUS   0:NORMAL  1:ERROR
     REGISTER PRESERVE
       (X)
```

```
(5) INTERNAL MICRO CASSETTE

  5.1  OPEN TO READ
       NAME
         OPNMCS
       PARAMETER
         (X): PACKET ADDRESS
       PACKET: BYTE 0: READ MODE
                       00:STOP EACH BLOCK
                       01:NON STOP
                       FF:DEPEND ON HEADER
               BYTE 1: BUFFER ADDRESS (HIGH)
               BYTE 2: BUFFER ADDRESS (LOW)
                        BUFFER SIZE = 256 BYTES + 4 BYTES
               BYTE 3 - BYTE 10: FILE NAME
               BYTE 11- BYTE 18: FILE TYPE
       RESULT
         (A): RETURN CODE
              00: NORMAL    ELSE ERROR
         (C): I/O ERROR FLAG
               0:NORMAL    1:ERROR
         (Z): DEPEND ON VALUE OF (A)

  5.2  SEARCH AND OPEN TO READ
       NAME
         SRCMCS
       PARAMETER
         (X): PACKET ADDRESS
       PACKET: BYTE 0: READ MODE
                       00:STOP EACH BLOCK
                       01:NON STOP
                       FF:DEPEND ON HEADER
               BYTE 1: BUFFER ADDRESS (HIGH)
               BYTE 2: BUFFER ADDRESS (LOW)
                        BUFFER SIZE = 256 BYTES + 4 BYTES
               BYTE 3 - BYTE 10: FILE NAME    ('*': MATCHING TERMAINTE)
               BYTE 11- BYTE 18: FILE TYPE    ('*': MATCHING TERMINATE)
               BYTE 19- BYTE 26: FOUND FILE NAME (SET AFTER CALLED)
               BYTE 27- BYTE 34: FOUND FILE TYPE (SET AFTER CALLED)
       PESULT
         (A): RETURN CODE
              00: NORMAL    ¥80:FOUND OTHER FILE (SKIPPED)
                   OTHERS: ERROR
         (C): I/O ERROR FLAG
               0:NORMAL    1:ERROR
         (Z): DEPEND ON VALUE OF (A)
```

```
;   5.3  SCAN AND MOVE EOF (DELETED)
;        NAME
;          SCNMCS
;        PARAMETER
;          (X): PACKET ADDRESS
;        PACKET: BYTE 0: BUFFER ADDRESS (HIGH)
;                BYTE 1: BUFFER ADDRESS (LOW)
;                          BUFFER SIZE = 256 BYTES + 4 BYTES
;                BYTE 2- BYTE 9: FOUND FILE NAME (SET AFTER CALLED)
;                BYTE 10- BYTE 17: FOUND FILE TYPE (SET AFTER CALLED)
;        RESULT
;          (A): RETURN CODE
;               00: NORMAL   OTHERS:ERROR
;          (C): I/O ERROR FLAG
;                0:NORMAL   1:ERROR

    5.4  READ ONE CHARACTER
         NAME
           REDMCS
         PARAMETER
           INPUT: NONE
         RESULT
           (A):CHARACTER CODE
           (B):ERROR CODE
                00: NORMAL   ELSE ERROR
           (Z): SET DEPEND ON (B)
           (C): I/O ERROR FLAG
                0:NORMAL  1:ERROR
         REGISTER PRESERVE  X
    5.5  CLOSE
         NAME
           CLSMCS
         PARAMETER NONE
         RESULT
           (C): I/O ERROR FLAG      0:NORMAL   1:ERROR
    5.6  OPEN TO WRITE
         NAME
           OPNWMC
         PARAMETER
           (X): PACKET ADDRESS
         PACKET: BYTE 0: WRITE MODE
                         00:STOP EACH BLOCK
                         01:NON STOP
                 BYTE 1: BUFFER ADDRESS (HIGH)
                 BYTE 2: BUFFER ADDRESS (LOW)
                           BUFFER SIZE = 256 BYTES + 4 BYTES
                 BYTE 3 - BYTE 10: FILE NAME
                 BYTE 11- BYTE 18: FILE TYPE
         RESULT
           (A): RETURN CODE
                00: NORMAL   ELSE ERROR
           (Z): SET DEPEND ON VALUE OF (A)
           (C): I/O ERROR FLAG   0:NORMAL  1:ERROR
```

```
    5.7  WRITE ONE CHARACTER
         NAME
           WRTMCS
         PARAMETER
           (A): WRITE CHARACTER
         RESULT
           (B): RETURN CODE
                00: NORMAL   ELSE ERROR
           (Z): SET DEPEND ON VALUE OF (B)
           (C): I/O ERROR STATUS   0:NORMAL  1:ERROR
         REGISTER PRESERVE
           (X)

   5.9  REWIND TO TOP OF THE TAPE
        NAME
          REWMCS
        PARAMETER
         ON ENTRY
           NONE
         ON EXIT
           (C): I/O ERROR STATUS   0:NORMAL   1:ERROR
           (A): ERROR CODE   0:NON ERROR  1:ERROR
           (Z): DEPEND ON VALUE OF (A)

    5.10 SEEK BY COUNTER VALUE
         NAME
           SEKMCS
         PARAMETER
          ON ENTRY
            (X): TARGET COUNTER VALUE
          ON EXIT
            (A): ERROR CODE   0:NON ERROR  1:CASSETTE ERROR
                              2:TOP OF TAPE POSITION  3:LAST POSITION
          RESISTER PRESERVE
            NONE

    5.11 GET/SET VALUE OF COUNTER
         NAME
           CNTMCS
         PARAMETER
          ON ENTRY
            (A):  0:GET COUNTER VALUE
                  1:SET COUNTER VALUE
            (X):  VALUE OF COUNTER (IF SET)
          ON EXIT
            (X):  VALUE OF COUNTER (IF GET)
          REGISTER PRESERVE   B

    PROCEDURE OF READ/WRITE CASSETTE (INTERNAL/EXTERNAL)
     (A) WRITE
       1. OPEN TO WRITE
       2. WRITE
       3. CLOSE

     (B) READ
       1. OPEN TO READ (ONE OF TWO OPEN ROUTINES)
       2. READ ONE CHARACTERS
       3. CLOSE
```

```
(6) ROM CASSETTE
 6.1 OPEN TO READ
    SUBROUTINE NAME
      OPNPRM
    ON ENTRY
      (X): PACKET ADDRESS
      (A): READ MODE  (0:NOT RETURN FILE NAME  1:RETURN FILE NAME)
    ON EXIT
      (A): RETURN CODE
           ¥00:NORMAL      ¥A0:NOT PROM CASSETTE
           ¥A1:NOT FOUND  ¥A2:OPEN ERROR
      (C): I/O ERROR FLAG   0:NORMAL   1:ERROR
      (Z): DEPEND ON VALUE OF (A)
    REGISTER PRESERVE   NONE
    PACKET
      BYTE0 - BYTE 7:FILE NAME                                         %
      BYTE8 - BYTE 15:FILE TYPE                                        %
      BYTE16 - BYTE23:FOUND FILE NAME (SET AFTER CALL, READ MODE (A:1))
      BYTE24 - BYTE31:FOUND FILE TYPE (SET AFTER CALL, READ MODE (A:1))

    NOTE.
      IF THERE IS '*' CHARACTER IN 'FILE NAME', 'FILE TYPE' (16 BYTES  %
      LENGTH), CHARACTER MATCHING IS TERMINATED.                       %

 6.2 READ ONE CHARACTER
   SUBROUTINE NAME
      REDPRM
   ON ENTRY
      PARAMETER NONE
   ON EXIT
      (A):CHARACTER CODE
      (B):STATUS       00:NORMAL    ¥01:END OF FILE   OTHERS:ERROR
      (C):I/O ERROR FLAG  0:NORMAL    1:ERROR
      (Z):DEPEND ON VALUE OF (B)
   REGISTER PRESERVE  (X)

 6.3 CLOSE ROM CASSETTE
    SUBROUTINE NAME
      CLSPRM
    ON ENTRY
      PARAMETER NONE
    ON EXIT
      (C): I/O ERROR FLAG   0:NORMAL  1:ERROR
    REGISTER PRESERVE
      NONE
```

```
6.4 READ DIRECTORY
  SUBROUTINE NAME
     DIRPRM
  ON ENTRY
     (X): MEMORY ADDRESS TO WRITE DIRECTORY INFORMATION                  %
     (A):DIRECTRY NUMBER  (0 - 63)
  ON EXIT
     (C): I/O ERROR STATUS     0:NORMAL   1:ERROR
     (A): ERROR CODE        0:NORMAL   1:ERROR
     (Z): DEPEND ON VALUE OF (A)
  REGISTER PRESERVE    NONE
  ANSWERED DIRECTORY INFORMATION
     BYTE 0 - BYTE 7: FILE NAME
     BYTE 8 - BYTE 15: FILE TYPE
     BYTE 16 - BYTE 19: START ADDRESS
     BYTE 20 - BYTE 23: END ADDRESS + 1
     BYTE 24 - BYTE 29: DATE
     BYTE 30 - BYTE 31: NOT DECIDED
  NOTE. IF DIRECTORY NO. = 0(FIRST DIRECTORY), DIRECTORY INFORMATION
        ARE CONTENTS OF PROM ¥0000 - ¥001F. IF 1, ¥0020 - ¥003F.

6.5 CHECK PLUG-IN OPTIONS
     CHECK PLUG-IN AND SET STATUS TO ¥79 (PLGSTS) AND (A) REGISTER    %
     SUBROUTINE NAME
       CHKPLG
     PARAMETER
      ON ENTRY  NONE
      ON EXIT
         (A): BIT 0 - BIT 2 : PLUG-IN OPTIONS SELECT
                        BIT 2,1,0 = 0 0 0 : PROM CASSETTE
                                    0 0 1 : SPARE
                                    0 1 0 : NOT PLUG-IN
                                    0 1 1 : SPARE
                                    1 X X : MICRO CASSETTE
              BIT 3 - BIT 7 :0
          (Z) DEPEND ON VALUE OF (A)
      REGISTER PRESERVE  B,X
```

```
(7) LOAD OR DUMP MEMORY (COMMON FOR DEVICE)
  7.1  OPEN TO DUMP CONTENTS OF MEMORY
       NAME
         OPNDMP
       PARAMETER
        ON ENTRY
         (X): PACKET ADDRESS
         (B): DEVICE NUMBER
       PACKET: BYTE 0: WRITE MODE
                       00:STOP EACH BLOCK
                       01:NON STOP
               BYTE 1: BUFFER ADDRESS (HIGH)
               BYTE 2: BUFFER ADDRESS (LOW)
                          BUFFER SIZE = 256 BYTES + 4 BYTES
               BYTE 3 - BYTE 10: FILE NAME
               BYTE 11- BYTE 18: FILE TYPE
               BYTE 19- BYTE 20: DUMP START ADDRESS
               BYTE 21- BYTE 22: DUMP LAST ADDRESS
               BYTE 23- BYTE 24: DUMP OFFSET
               BYTE 25- BYTE 26: START ADDRESS
       ON EXIT
          (A): RETURN CODE
               00: NORMAL    OTHERS:ERROR
          (Z): DEPEND ON VALUE OF (A)
          (C): I/O ERROR     0:NORMAL   1:ERROR

    NOTE.  DEVICE NUMBER
      '0' : RS232C 110 BPS
      '1' : RS232C 150 BPS
      '2' : RS232C 300 BPS
      '3' : RS232C 600 BPS
      '4' : RS232C 1200 BPS
      '5' : RS232C 2400 BPS
      '6' : RS232C 4800 BPS

      'M' : MICRO CASSETTE
      'C' : EXTERNAL CASSETTE
          DEVICE NUMBER '0' - '6' ARE NOT SUPPORTED IN JAPAN AND
          EUROPE VERSION

  7.2  DUMP CONTENTS OF MEMORY TO OPENED FILE AND AUTOMATICALY CLOSE
       NAME
         DMPDVS
       PARAMETER
          (B): DEVICE NUMBER
       RESULT
          (A): RETURN CODE
               00: NORMAL    OTHERS:ERROR
          (Z): DEPEND ON VALUE OF (A)
          (C): I/O ERROR     0:NORMAL   1:ERROR
```

```
  7.3  OPEN TO LOAD MEMORY FROM DESTINATED DEVICE
       NAME
         OPNLOD
       PARAMETER
         (A): REQUEST TO RETURN FILE NAME FLAG (1:RETURN FILE NAME)
                      (0:NOT RETURN FILE NAME)
         (B): DEVICE NUMBER
         (X): PACKET ADDRESS
       PACKET: BYTE 0: 00:
               BYTE 1: BUFFER ADDRESS (HIGH)
               BYTE 2: BUFFER ADDRESS (LOW)
                          BUFFER SIZE = 256 BYTES + 4 BYTES
               BYTE 3 - BYTE 10: FILE NAME (TERMINATE = '*')
               BYTE 11- BYTE 18: FILE TYPE (TERMINATE = '*')
               BYTE 19- BYTE 20: FOUND FILE NAME (RETURN NAME MODE)
               BYTE 21- BYTE 22: FOUND FILE TYPE (RETURN NAME MODE)

       RESULT
         (A): RETURN CODE
              00: NORMAL    ELSE ERROR
         (Z): SET DEPEND ON VALUE OF (A)
         (C): I/O ERROR FLAG   0:NORMAL   1:ERROR

  7.4  LOAD MEMORY AND AUTOMATICALLY CLOSE
       NAME
         LODDVS
       PARAMETER
        ON ENTRY
         (A): LOAD MODE    00:LOAD TO MEMORY WITH CRC VERIFY
                           01:CRC VERIFY BUT NO LOAD
                              (OK IF ONE OF SAME BLOCK IS COMPLETED)
         (B): DEVICE NUMBER
         (X): OFFSET VALUE
        ON EXIT
         (A): RETURN CODE
              00: NORMAL    ELSE ERROR
         (Z): SET DEPEND ON VALUE OF (A)
         (C): I/O ERROR FLAG   0:NORMAL   1:ERROR
         (X): START ADDRESS (ADD OFFSET VALUE, IF 'LOAD' PROCESS)

PROCEDURE
   (A) DUMP
      1: OPEN TO DUMP
      2: DUMP
   (B) LOAD
      1: OPEN TO LOAD
      2: LOAD

NOTE. IF CALL 'OPEN TO DUMP' ROUTINE, DESTINATE DEVICE WILL FORCE
  CLOSED. AFTER COMLETED TO DUMP/LOAD, AUTOMATICALLY CLOSED
 DEVICE NUMBER
      '0' : RS232C 110 BPS
      '1' : RS232C 150 BPS
      '2' : RS232C 300 BPS
      '3' : RS232C 600 BPS
      '4' : RS232C 1200 BPS
      '5' : RS232C 2400 BPS
      '6' : RS232C 4800 BPS
      'M' : MICRO CASSETTE
      'C' : EXTERNAL CASSETTE
      'P' : PROM CASSETTE
```

```
(8) SERIAL PORT ROUTINES
 8.1 DRIVER ON/OFF
   SUBROUTINE NAME   SERONF
    ON ENTRY  (A):  0:OFF    1:ON
    ON EXIT   (C):  0:NORMAL    1:I/O ERROR
              (A):  0:NORMAL    OTHERS:ERROR
              (Z):  DEPEND ON VALUE OF (A)
    REGISTER PRESERVE  B,X


 8.2 OUT TO SERIAL
   SUBROUTINE NAME    OUTSRL
     ON ENTRY
              (X): PACKET ADDRESS
              (A): LSB: 1:AFTER SEND FUNCTION, ENTER RECIVE FUNCTION  %
                        0:NOT CONTINUE TO RECEIVE FUNCTION            %
     ON EXIT  (C):  0:NORMAL    1:I/O ERROR                           %
              (A):  0:NORMAL    ¥B1:DEVICE ERROR
     REGISTER PRESERVE  X
     PACKET
         1. FORMAT
         2. DESTINATION DEVICE NUMBER  (1 BYTE)
         3. SOURCE DEVICE NUMBER
         4. FUNCTION
         5. CHARACTER LENGTH (1BYTE) (DATA STRING -1)
         .
         .
         N. LAST CHARACTER

     EXAMPLE
       LDA A   #¥0
       LDX     #PACKET
       JSR     OUTSRL
         .
         .
   OUTSRL EQU    ¥FFD0-96
   PACKET FCB    ¥0,¥31,¥30,¥34,¥0D    * SEND CHARACTER 'CR'TO CRT
         .
         .
```

```
8.3 RECEIVE FROM SERIAL PORT
  SUBROUTINE NAME    INSRL
    ON ENTRY (X): ADDRESS OF RECEIVED DATA STRING                    %
    ON EXIT  (C):  0:NORMAL    1:I/O ERROR                           %
             (A):  0:NORMAL    ¥80:TIME OUT  ¥B1:DEVICE ERROR        %
             (B):  RECEIVED BLOCK STATUS
                   0:RECEIVED WITH HEADER
                   1:RECEIVED WITHOUT HEADER
    REGISTER PRESERVE  X
    RECEIVED DATA (RECEIVED WITH HEADER)
        1. FORMAT
        2. DESTINATION DEVICE NUMBER  (1 BYTE)                        %
        3. SOURCE DEVICE NUMBER (1 BYTE)
        4. FUNCTION(1 BYTE)  (SET AFTER RECEIVED)
        5. PACKET CHARACTER LENGTH (1BYTE) (DATA STRING -1)
           (SET AFTER RECIVED)
        .(STORE AFTER CALLED)
        .
        N. LAST CHARACTER

    RECEIVED DATA (RECEIVED WITHOUT HEADER)
        1. NOT USED
        2. NOT USED
        3. NOT USED
        4. NOT USED
        5. PACKET CHARACTER LENGTH (1BYTE) (DATA STRING -1)
           (SET AFTER RECIVED)
        .(STORE AFTER CALLED)
        .
        N. LAST CHARACTER

      LDX     #RCVADR
      JSR     INSRL
        .
        .
  OUTSRL EQU    ¥FFD0-99
  RCVADR RMB    1           * FORMAT
         RMB    1           * DESTINATION DEVICE NUMBER
         RMB    1           * SOURCE DEVICE NUMBER
         RMB    1           * FUNCTION
         RMB    5           * RECIVED DATA

;8.4 WAIT TO BE SELCTED  (DELETED)                                    %
;SUBROUTINE NAME  SERSLC                                              %
;  ON ENTRY                                                           %
;          (A):DESTINATION DEVICE NUMBER                              %
;          (B):SOURCE DEVICE NUMBER                                   %
;          (X):TIME OVER LIMIT (1=0.1 SEC    0:WITHOUT LIMIT)         %
;  ON EXIT                                                            %
;          (A):RETURN CODE   00:OK   ¥80:TIME OVER                    %
;          (Z):DEPEND ON VALUE OF (A)                                 %
;                                                                     %
                                                                      %
```

```
9.1 GET CURRENT TIME AND DATE                                        *
  SUBROUTINE NAME   GETCLK                                           *
    ON ENTRY (X): ADDRESS WHERE DATE AND TIME IS STORED              *
                  WORK AREA IS NEEDED 6 BYTES                        *
    ON EXIT  DESCRIVED ADDRESS: DATE AND TIME                        *
                          (MM DD YY HH MM SS) (BCD CODE)             *
    REGISTER PRESERVE  X                                             *
                                                                     *
9.2 SET CURRENT TIME AND DATE                                        *
SUBROUTINE NAME  SETCLK                                              *
  ON ENTRY                                                           *
          (X): ADDRESS WHERE DATE AND TIME IS STORED               *
              WORK AREA IS NEEDED 6 BYTES                         *
  ON EXIT                                                            *
    PARAMETER NONE                                                   *
    REGISTER PRESERVE  X                                             *
                                                                     *
                                                                     *
```

```
2. RAM MEMORY MAP
 2.1 ZERO PAGE RAM (CLOCK RAM)

  #4E      : POWER ON/OFF STATUS                                              *)
   4F      : DATA OF ADDRESS ¥26 (OUT PORT)

     REGISTER USED BY MAIN I/O
   50      : REGISTER (R0H)  R0:(R0H,R0L)
   51      : REGISTER (R0L)
   52      : REGISTER (R1H)  R1:(R1H,R1L)
   53      : REGISTER (R1L)
   54      : REGISTER (R2H)  R2:(R2H,R2L)
   55      : REGISTER (R2L)
   56      : REGISTER (R3H)  R3:(R3H,R3L)
   57      : REGISTER (R3L)
   58      : REGISTER (R4H)  R4:(R4H,R4L)
   59      : REGISTER (R4L)
   5A      : REGISTER (R5H)  R5:(R5H,R5L)
   5B      : REGISTER (R5L)
   5C      : REGISTER (R6H)  R6:(R6H,R6L)
   5D      : REGISTER (R6L)
   5E      : REGISTER (R7H)  R7:(R7H,R7L)
   5F      : REGISTER (R7L)

    REGISTER USED BY MONITOR (INTERRUPT)
   60      : REGISTER (M0H)  M0:(M0H,M0L)
   61      : REGISTER (M0L)
   62      : REGISTER (M1H)  M1:(M1H,M1L)
   63      : REGISTER (M1L)
   64      : REGISTER (M2H)  M2:(M2H,M2L)
   65      : REGISTER (M2L)
   66      : REGISTER (M3H)  M3:(M3H,M3L)
   67      : REGISTER (M3L)
   68      : REGISTER (M4H)  M4:(M4H,M4L)
   69      : REGISTER (M4L)
   6A      : REGISTER (M5H)  M5:(M5H,M5L)
   6B      : REGISTER (M5L)
   6C      : REGISTER (M6H)  M6:(M6H,M6L)
   6D      : REGISTER (M6L)
   6E      : REGISTER (M7H)  M7:(M7H,M7L)
   6F      : REGISTER (M7L)

   70      : KEY ROUTINE REGISTER (INTERRUPT)  K0H   K0(K0H,K0L)
   71      : KEY ROUTINE REGISTER (INTERRUPT)  K0L
   72      : KEY ROUTINE REGISTER (INTERRUPT)  K1H   K1(K1H,K1L)
   73      : KEY ROUTINE REGISTER (INTERRUPT)  K1L
   74      : SERIAL INTERRUPT REGISTER         S0H   S0(S0H,S0L)
   75      : SERIAL INTERRUPT REGISTER         S0L
   76      : SERIAL INTERRUPT REGISTER         S1H   S1(S1H,S1L)
   77      : SERIAL INTERRUPT REGISTER         S1L
   78      : INITIALIZED FLAG 1 (0 - 7)
   79      : PLUG-IN OPTIONS SELECT
   7A      : RS232 SLAVE READ MODE
   7B      : RUN MODE
   7C      : SLAVE I/O STATUS
   7D      : MAIN I/O STATUS
   7E      : SOFTWARE SWITCH 1
   7F      : SOFTWARE SWITCH 2
```

```
NOTE. THE CONTENTS OF RAM WITH '#' MARK ARE INITIALIZED BY POWER ON
(SET TO 0). WITH '&' MARK ARE INITIALIZED BY SYTEM RESET.

 #¥4E. POWER ON STATUS                                                    %
   #BIT0-BIT3:START MODE (POWER ON BY CLOCK)                              %
              INITIALIZED TO ¥00                                          %
              ¥01:POWER ON IN APPLICATION MODE                            %
              ¥02:POWER ON IN BASIC MODE                                  %
     NOTE. WHEN POWER IS ON BY CLOCK, IF STATUS IS SET ¥01 OR ¥02,        %
           AFTER INITIALIZE I/O, CALL POWER ON PROCEDURE(POINTED BY       %
           ¥130-131), THEN JUMP TO MENU ROUTINE.                          %
   #BIT4-BIT7:POWER OFF MODE (TURN OFF BY SWITCH)                         %
              INITIALIZED TO ¥00                                          %
              ¥01:POWER OFF IN APPLICATION MODE                           %
              ¥02:POWER OFF IN BASIC MODE                                 %
     NOTE. WHEN POWER SWITCH IS TURN OFF, IF STATUS IS SET ¥01 OR         %
           ¥02, BEFORE POWER OFF, CALL POWER OFF PROCEDURE (POINTED       %
           BY ¥132-133), THEN JUMP TO POWER OFF ROUTINE.                  %
 ¥78. INITIALIZED FLAG 1 (0:REQUEST INITIALIZE  1:COMPLETED)              %
    &BIT 0:MENU
    &BIT 1:CLOCK
    &BIT 2:CALCLATOR
    &BIT 3:NOT USED
    &BIT 4:NOT USED
    &BIT 5:NOT USED
    &BIT 6:BASIC APLLICATION
    &BIT 7:BASIC
 ¥79. PULG-IN OPTIONS SELECT
    #BIT 0: SLAVE P46      BIT (2,1,0) = 0 0 0  :PROM CASSETTE            %
    #BIT 1: SLAVE P20                    0 0 1  :SPARE                    %
    #BIT 2: MAIN P17                     0 1 0  :NOT PULG-IN              %
    #BIT 3: ALWAYS 0                     0 1 1  :SPARE                    %
                                         1 X X  :MICRO CASSETTE           %
    &BIT 4:NOT USED
    &BIT 5:NOT USED
    &BIT 6:NOT USED
    &BIT 7:      WHEN HIT BREAK KEY, POWER OFF RS232 DRIVER               %
                      (1:POWER OFF   0:NOT)                               %
 ¥7A. SERIAL PORT STATUS
    #BIT  0,1: RS232 MODE (00:STOP RS232  01:READING BY INTERRUPT MODE
                              10:READING BY READ ONE CHARACTER MODE)
    #BIT  2:   RS232 ON EXECUTE/PAUSE (0:ON EXECUTE(BIT0,1 NOT 0), STOP
                                         1:PAUSE)
    #BIT  3:   ON RS232 DRIVER (0:OFF  1:DRIVER ON)
    #BIT  4:   ON SERIAL DRIVER (0:OFF  1:DRIVER ON)
    #BIT  5,6,7  SERIAL PORT INTERRUPT MODE
               000:READ EXTERNAL CASSETTE   001:READ MICRO CASSETTE
               010:READ RS232   011:NOT USED (FOR READ)
               100:WRITE EXTERNAL CASSETTE    101:WRITE MICRO CASSETTE
               110,111:NOT USED (FOR WRITE)
 ¥7B. RUN MODE
    #BIT  0,1,2,3  RUN NAME (0:BASIC,  1:
    #BIT  4,5,: NOT USED
    #BIT  6: SCREEN STATUS (0:VIRTUAL SCREEN  1:PHISICAL SCREEN)          %
    #BIT  7: RUNNING MODE (1:INTERRPRETER MODE  2:MACHINE LANGAGE)        %
```

```
 ¥7C. SLAVE I/O STATUS   (ON WHEN 1)
   #BIT  0:     PRINTER
   #BIT  1:     EXTERNAL CASSETTE
   #BIT  2:     INTERNAL CASSETTE
   #BIT  3:     RS232 ON (READ)
   #BIT  4:     SPEAKER
   #BIT  5:     PROM CASSETTE POWER
   #BIT  6:     ON BARCODE READER
   #BIT  7:     BROKEN SLAVE CPU BY BREAK KEY (0:NOT  1:BROKEN)
 ¥7D. MAIN I/O STATUS    (0:OFF  1:ON)
   #BIT  0:     LCD ON READ/WRITE CHARACTER
   #BIT  1:     ON CONTINE TO TRANSMIT TO SLAVE CPU                       %
   #BIT  2:     ON CONTINE TO TRANSMIT TO SERAIL LINE                     %
   #BIT  3:     ON CLOCK INTERRUPT                                        %
   #BIT  4:     (POWER FAIL)
   #BIT  5:     (OFF POWER SWITCH)
   #BIT  6:     ON PAUSE KEY
   #BIT  7:     ON BREAK KEY
 ¥7E. SOFTWARE SWITCH 1
   &BIT  0:     CASSEETE PULSE MODE (0:NORMAL  1:REVERSE)
   &BIT  1:     CASSETTE PULSE MODE (1:DEPEND ON BIT 0, 0:AUTO SELECT)%
   &BIT  2:     MICRO CASSETTE PULSE MODE (0:NORMAL  1:REVERSE)           %
   &BIT  3:     MICRO CASSETTE PULSE MODE (0:DEPEND ON BIT 2              %
                                    (1:DEPEND ON BIT 2  0:REVERSE)        %
   &BIT  4,5:   SELECTED BANK (0:BANK 0  1:BANK 1)                        %
   &BIT  6:     BASIC OPTION SELECT                                       %
   #BIT  7:     ADDRESS 00-4D ACCESS MASK (0:DISABLE  1:ENABLE)
 ¥7E. SOFTWARE SWITCH 2  (VALUE OF DIP SWITCH AND PRINTER SWITCH)
   &BIT  0:     DIP SWITCH 1
   &BIT  1:     DIP SWITCH 2
   &BIT  2:     DIP SWITCH 3
   &BIT  3:     DIP SWITCH 4
   #BIT  4:     ENABLE BIT0 - BIT3 (0:DISABLE  1:ENABLE)                  %
   #BIT  5:     ENABLE BIT 7 (0:DISABLE  1:ENABLE)                        %
   &BIT  6:     NOT USED
   &BIT  7:     PRINTER ON/OFF SWITCH
 RAM MEMORY MAP 2

(A) INTERRUPT JUMP ADDRESS(¥100 - ¥11D)
#¥100 - 102: CLOCK INTERRUPT JUMP
#¥103 - 105: EXTERNAL PORT INTERRUPT (IRQ1)
#¥106 - 108: TRAP
#¥109 - 10B: SCI INTERRUPT
#¥10C - 10E: TOF INTERRUPT
#¥10F - 111: OCF INTERRUPT
#¥112 - 114: ICF INTERRUPT
#¥115 - 117: IRQ1 INTERRUPT
&¥118 - 11A: SWI
&¥11B - 11D: NMI
```

```
(B) VECTOR IN RAM. (¥11E - ¥120)
&¥11E - 11F: ADDRESS OF ¥E0 - ¥FF CHRACTER FONT TABLE
#¥120 - 121: WHEN PUSHED BREAK KEY, JUMP ADDRESS(NOT IN BASIC MODE)
#¥122 - 123: WHEN PUSHED MENU KEY, JUMP ADDRESS (NOT IN BASIC MODE)
#¥124 - 125: WHEN PUSHED PAUSE KEY, JUMP ADDRESS(NOT IN BASIC MODE)
#¥126 - 127: WHEN PUSHED CTRL/PF3 KEY, JUMP ADDRESS                    %
#¥128 - 129: WHEN PUSHED CTRL/PF4 KEY, JUMP ADDRESS                    %
#¥12A - 12B: WHEN PUSHED CTRL/PF5 KEY, JUMP ADDRESS                    %
&¥12C - 12D: RAM BOTTOM ADDRESS                                        %

(C) ROM CASSETTE DATA COUNTER                                          %
 ¥12E - 12F: NUMBER OF BYTES IN THE CURRENT ROM FILE.                  @

(D) VECTOR BY WAKE UP  (¥130 - ¥133)                                   %
#¥130 - 131: POWER ON BY WAKE-UP, CALLED PROCEDURE ADDRESS             %
#¥132 - 133: POWER OFF, CALLED PROCEDURE ADRESS                        %

(E) VETOR BY BASIC
#¥134 - 135: TOP ADDRESS OF BASIC PROGRAM                              %
#¥136 - 137: LAST ADDRESS OF BASIC PROGRAM                             %
#¥138 - 139: ENTRY POINT TO GARBAGE COLLECTOR                          %
                                                                       %
(F) HEADER BY MENU                                                     %
#¥13A - 13B:EXIST HEADER FLAG IN EACH ROM                              %
#¥13C - 13F:HEADING BY USER'S PROGRAM (USED BY MENU)                   %

(G) OTHER WORK                                                         @
 ¥140 - ¥18F(188): KEY BOARD ROUTINE WORK AREA                         %
 ¥190 - ¥1AE: MICRO PRINTER WORK AREA                                  @
 ¥1AF - ¥1C3: RS232C WORK AREA                                         @
 ¥1C4 - ¥1D4: SERIAL COMMUNICATION WORK AREA                           @
 ¥1D5 - ¥1EB: EXTERNAL CASSETTE WORK AREA                              @
 ¥1EC - ¥207: INTERNAL MICRO CASSETTE WORK AREA                        @
 ¥208 - ¥20E: ROM CARTRIDGE WORK AREA                                  @
 ¥20F - ¥21A: BINARY MEMORY DUMP/LOAD WORK AREA                        @
 ¥21B       : FOR MICRO CASSETTE                                       @
 ¥21C - ¥21F: SPARE (NOT USED)                                         @
 ¥220 - ¥29F: SCREEN WORK AREA                                         @
 ¥2A0 - ¥2CF: MONITOR WORK AREA                                        @
 ¥2D0 - ¥323: EXTERNAL CASSETTE HEADER                                 @
 ¥324 - ¥377: MICRO CASSETTE HEADER                                    @
 ¥380 - ¥47C: BUFFER USED BY MONITOR R/W ROUTINE                       @
 ¥47D - ¥4AF: STACK (DEFAULT)                                          @
```

```
KEY BOARD ROUTINE

REVISION     B-1
CREATED      12.17.1981
             K. AKAHANE
UPDATE       06.07.1982
UPDATE       06.24.1982
UPDATE       11.16.1982
FILE NAME    KEY HX1D

UPDATE MARK = 72 COLUMN '%' TO REVISION A
UPDATE MARK = 72 COLUMN '@' TO REVISION B

 KEY INPUT PROCEDURE
  AFTER 'OPEN KEY', PUSHED KEY CODES ARE AUTOMATICALY PUSHED INTO KEY
 STACK BY PUSHED KEY INTERRUPT OR INTERVAL TIMER INTERRUPT. THE KEY
 STACK HAS EIGHT BYTES MAX SIZE.

KEY AUTO REPEAT TIME
 SAMPLING TIME 20 M SEC
 FIRST         800 M SEC (INTERRUPT COUNT = 40 TIMES)
 AFTER SECOND  120 M SEC (INTERRUPT COUNT = 6 TIMES)

USE FREE RUNNING COUNTER
USE OCF INTERRUPT (20 M SEC)
KEY STACK = 3 BYTES MAX

AUTO REPEAT CODE
 1. ALPHA NUMERIC KEY
 2. CURSOR LEFT/RIGHT, CURSOR UP/DOWN
 3. SCROLL UP/DOWN
 4. DEL/INS
 5. CLEAR/HOME
 6. HTAB
 7. PAPER FEED (IS NOT ACCEPTED BY KEYIN ROUTINE)
NOT REPEAT KEY
 1. FUNCTION KEY
 2. BREAK
 3. HELP
 4. PAUSE

SPECIAL FUNCTION
 1. CTRL/F1  MICRO CASSETTE MANUAL FUNCTION
 2. CTRL/F2  SCREEN COPY

  MICRO CASSETTE MANUAL FUNCTION COMMAND
   1. COUNTER RESET
   2. REWIND
   3. PLAY
   4. FAST FEED
   5. STOP
   6. QUIT

DIP SWITCH
 LOW 3 BITS (US VERSION)                LOW 3 BITS (EUROPE VERSION)

   BIT2 BIT1 BIT0                         BIT2 BIT1 BIT0
    1    1    1      USA                   1    1    1     NORWAY
    1    1    0      FRANCE                1    1    0     FRANCE
    1    0    1      GERMANY               1    0    1     GERMANY
    1    0    0      ENGLAND               1    0    0     SWEDEN
    0    1    1      DENMARK               0    1    1     DENMARK
    0    1    0      SWEDEN                0    1    0     FRANCE (ASCII CODE)
    0    0    1      ITALY                 0    0    1     GERMANY(ASCII CODE)
```

30

```
 0      0      0      SPAIN                0      0      0      SWEDEN (ASCII CODE)
```

LIST OF GENERATED CODE FROM KEYBOARD

```
NOTE.  G/ = 'GRAPH'
       C/ = 'CONTROL'
       S. = 'NOT CAPITAL LETTER'
       (| = LEFT SQUARE BRACKET
       )| = RIGHT SQUARE BRACKET
       (" = LEFT CURLY BRACKET
       )" = RIGHT CURLY BRACKET
       ¥  = DOLLER SIGN
       ¬  = REVERSE /    (IN KANA VERSION, CHANGED TO '¥' CHARACTER)
       || = UP ARROW HEAD
       |' = EXCLAMATION MARK
       S/ = KANA KOMOJI
```

KEY INPUT CODES (NORMAL OR SHIFT)

	0	1	2	3	4	5	6	7
0			SP	0	@	P		S.P
1			|'	1	A	Q	S.A	S.Q
2			"	2	B	R	S.B	S.R
3			#	3	C	S	S.C	S.S
4			¥	4	D	T	S.D	S.T
5			%	5	E	U	S.E	S.U
6			&	6	F	V	S.F	S.V
7			'	7	G	W	S.G	S.W
8			(	8	H	X	S.H	S.X
9			)	9	I	Y	S.I	S.Y
A			*	:	J	Z	S.J	S.Z
B			+	;	K	(|	S.K	('
C			,	<	L	¬	S.L	|
D			-	=	M	)|	S.M	)'
E			.	>	N	||	S.N	
F			/	?	O	_	S.O	

KEY INPUT CODES (CONTROL CODES) (CONTROL KEY + KEY CODE)

	0	1	2	3	4	5	6	7
0	C/@	C/P						
1	C/A	C/Q						
2	C/B	C/R						
3	C/C	C/S						
4	C/D	C/T						
5	C/E	C/U						
6	C/F	C/V						
7	C/G	C/W						
8	C/H	C/X						
9	C/I	C/Y						
A	C/J	C/Z						
B	C/K	C/(|						
C	C/L	C/¬						
D	C/M	C/)|						
E	C/N	C/||		C/G/)|				
F	C/O	C/_		C/G/¬				

	8	9	A	B	C	D	E	F
0								C/0
1							C/!	C/1
2							C/"	C/2
3							C/#	C/3
4							C/¥	C/4
5							C/%	C/5
6							C/&	C/6
7							C/'	C/7
8							C/(	C/8
9							C/)	C/9
A							C/*	C/:
B							C/+	C/;
C							C/,	C/<
D							C/-	C/=
E							C/.	C/>
F							C//	C/?

NOTE. CONTROL KEY IS EFFECTIVE TO ¥20 - ¥5F CODE, THESE CODES ARE SUBTRACTED ¥40.

GRAPH MODE CODE

	0	1	2	3	4	5	6	7
0							G/(| (US)	
1								
2								
3								
4								
5								
6								
7								
8								
9								
A								
B								
C								
D								
E								G/)| (US)
F								G/¬

	8	9	A	B	C	D	E	F
0	G/S	G/U	G/-				G/0	
1	G/X	G/I					G/1	
2	G/W	G/O					G/2	
3	G/D	G/P					G/3	
4	G/A	G/@					G/4	
5	G/T	G/K					G/5	
6	G/R	G/V					G/6	
7	G/Q	G/,					G/7	
8	G/E	G/M					G/8	
9	G/Z	G/N					G/9	
A	G/C	G/B					G/(| (EU)	
B	G/J	G/;					G/)| (EU)	
C	G/F	G/.						
D	G/G	G/:						
E	G/H	G//						
F	G/Y	G/L						

2. RAM MEMORY MAP

2.1 ZERO PAGE RAM (CLOCK RAM)

```
 #4E      : POWER ON/OFF STATUS                                          *1
  4F      : DATA OF ADDRESS ¥26 (OUT PORT)

    REGISTER USED BY MAIN I/O
  50      : REGISTER (R0H)  R0:(R0H,R0L)
  51      : REGISTER (R0L)
  52      : REGISTER (R1H)  R1:(R1H,R1L)
  53      : REGISTER (R1L)
  54      : REGISTER (R2H)  R2:(R2H,R2L)
  55      : REGISTER (R2L)
  56      : REGISTER (R3H)  R3:(R3H,R3L)
  57      : REGISTER (R3L)
  58      : REGISTER (R4H)  R4:(R4H,R4L)
  59      : REGISTER (R4L)
  5A      : REGISTER (R5H)  R5:(R5H,R5L)
  5B      : REGISTER (R5L)
  5C      : REGISTER (R6H)  R6:(R6H,R6L)
  5D      : REGISTER (R6L)
  5E      : REGISTER (R7H)  R7:(R7H,R7L)
  5F      : REGISTER (R7L)

   REGISTER USED BY MONITOR (INTERRUPT)
  60      : REGISTER (M0H)  M0:(M0H,M0L)
  61      : REGISTER (M0L)
  62      : REGISTER (M1H)  M1:(M1H,M1L)
  63      : REGISTER (M1L)
  64      : REGISTER (M2H)  M2:(M2H,M2L)
  65      : REGISTER (M2L)
  66      : REGISTER (M3H)  M3:(M3H,M3L)
  67      : REGISTER (M3L)
  68      : REGISTER (M4H)  M4:(M4H,M4L)
  69      : REGISTER (M4L)
  6A      : REGISTER (M5H)  M5:(M5H,M5L)
  6B      : REGISTER (M5L)
  6C      : REGISTER (M6H)  M6:(M6H,M6L)
  6D      : REGISTER (M6L)
  6E      : REGISTER (M7H)  M7:(M7H,M7L)
  6F      : REGISTER (M7L)

  70      : KEY ROUTINE REGISTER (INTERRUPT)  K0H   K0(K0H,K0L)
  71      : KEY ROUTINE REGISTER (INTERRUPT)  K0L
  72      : KEY ROUTINE REGISTER (INTERRUPT)  K1H   K1(K1H,K1L)
  73      : KEY ROUTINE REGISTER (INTERRUPT)  K1L
  74      : SERIAL INTERRUPT REGISTER         S0H   S0(S0H,S0L)
  75      : SERIAL INTERRUPT REGISTER         S0L
  76      : SERIAL INTERRUPT REGISTER         S1H   S1(S1H,S1L)
  77      : SERIAL INTERRUPT REGISTER         S1L
  78      : INITIALIZED FLAG 1 (0 - 7)
  79      : PLUG-IN OPTIONS SELECT
  7A      : RS232 SLAVE READ MODE
  7B      : RUN MODE
  7C      : SLAVE I/O STATUS
  7D      : MAIN I/O STATUS
  7E      : SOFTWARE SWITCH 1
  7F      : SOFTWARE SWITCH 2
```

```
NOTE. THE CONTENTS OF RAM WITH '#' MARK ARE INITIALIZED BY POWER ON
(SET TO 0). WITH '&' MARK ARE INITIALIZED BY SYTEM RESET.

#¥4F. POWER ON STATUS                                                  %
  #BIT0-BIT3:START MODE (POWER ON BY CLOCK)                            %
             INITIALIZED TO ¥00                                        %
             ¥01:POWER ON IN APPLICATION MODE                          %
             ¥02:POWER ON IN BASIC MODE                                %
    NOTE. WHEN POWER IS ON BY CLOCK, IF STATUS IS SET ¥01 OR ¥02,      %
          AFTER INITIALIZE I/O, CALL POWER ON PROCEDURE(POINTED BY     %
          ¥130-131), THEN JUMP TO MENU ROUTINE.                        %
  #BIT4-BIT7:POWER OFF MODE (TURN OFF BY SWITCH)                       %
             INITIALIZED TO ¥00                                        %
             ¥01:POWER OFF IN APPLICATION MODE                         %
             ¥02:POWER OFF IN BASIC MODE                               %
    NOTE. WHEN POWER SWITCH IS TURN OFF, IF STATUS IS SET ¥01 OR       %
          ¥02, BEFORE POWER OFF, CALL POWER OFF PROCEDURE (POINTED     %
          BY ¥132-133), THEN JUMP TO POWER OFF ROUTINE.                %
 ¥78. INITIALIZED FLAG 1 (0:REQUEST INITIALIZE  1:COMPLETED)           %
   &BIT 0:MENU
   &BIT 1:CLOCK
   &BIT 2:CALCLATOR
   &BIT 3:NOT USED
   &BIT 4:NOT USED
   &BIT 5:NOT USED
   &BIT 6:BASIC APLLICATION
   &BIT 7:BASIC
 ¥79. PULG-IN OPTIONS SELECT
   #BIT 0: SLAVE P46     BIT (2,1,0) = 0 0 0  :PROM CASSETTE           %
   #BIT 1: SLAVE P20                   0 0 1  :SPARE                   %
   #BIT 2: MAIN P17                    0 1 0  :NOT PULG-IN             %
   #BIT 3: ALWAYS 0                    0 1 1  :SPARE                   %
                                       1 X X  :MICRO CASSETTE          %
   &BIT 4:NOT USED
   &BIT 5:NOT USED
   &BIT 6:NOT USED
   &BIT 7:     WHEN HIT BREAK KEY, POWER OFF RS232 DRIVER              %
                    (1:POWER OFF   0:NOT)                              %
 ¥7A. SERIAL PORT STATUS
   #BIT  0,1: RS232 MODE (00:STOP RS232  01:READING BY INTERRUPT MODE
                          10:READING BY READ ONE CHARACTER MODE)
   #BIT  2:   RS232 ON EXECUTE/PAUSE (0:ON EXECUTE(BIT0,1 NOT 0), STOP
                                            1:PAUSE)
   #BIT  3:   ON RS232 DRIVER (0:OFF  1:DRIVER ON)
   #BIT  4:   ON SERIAL DRIVER (0:OFF  1:DRIVER ON)
   #BIT  5,6,7  SERIAL PORT INTERRUPT MODE
              000:READ EXTERNAL CASSETTE   001:READ MICRO CASSETTE
              010:READ RS232   011:NOT USED (FOR READ)
              100:WRITE EXTERNAL CASSETTE    101:WRITE MICRO CASSETTE
              110,111:NOT USED (FOR WRITE)
 ¥7B. RUN MODE
   #BIT  0,1,2,3  RUN NAME (0:BASIC,  1:
   #BIT  4,5,: NOT USED
   #BIT  6: SCREEN STATUS (0:VIRTUAL SCREEN  1:PHISICAL SCREEN)        %
   #BIT  7: RUNNING MODE (1:INTERRPRETER MODE  2:MACHINE LANGAGE)      %
```

```
¥7C. SLAVE I/O STATUS   (ON WHEN 1)
  #BIT  0:    PRINTER
  #BIT  1:    EXTERNAL CASSETTE
  #BIT  2:    INTERNAL CASSETTE
  #BIT  3:    RS232 ON (READ)
  #BIT  4:    SPEAKER
  #BIT  5:    PROM CASSETTE POWER
  #BIT  6:    ON BARCODE READER
  #BIT  7:    BROKEN SLAVE CPU BY BREAK KEY (0:NOT  1:BROKEN)
¥7D. MAIN I/O STATUS    (0:OFF  1:ON)
  #BIT  0:    LCD ON READ/WRITE CHARACTER
  #BIT  1:    ON CONTINE TO TRANSMIT TO SLAVE CPU                        %
  #BIT  2:    ON CONTINE TO TRANSMIT TO SERAIL LINE                      %
  #BIT  3:    ON CLOCK INTERRUPT                                         %
  #BIT  4:    (POWER FAIL)
  #BIT  5:    (OFF POWER SWITCH)
  #BIT  6:    ON PAUSE KEY
  #BIT  7:    ON BREAK KEY
¥7E. SOFTWARE SWITCH 1
  &BIT  0:    CASSEETE PULSE MODE (0:NORMAL  1:REVERSE)
  &BIT  1:    CASSETTE PULSE MODE (1:DEPEND ON BIT 0, 0:AUTO SELECT)%
  &BIT  2:    MICRO CASSETTE PULSE MODE (0:NORMAL  1:REVERSE)            %
  &BIT  3:    MICRO CASSETTE PULSE MODE (0:DEPEND ON BIT 2               %
                                 (1:DEPEND ON BIT 2  0:REVERSE)          %
  &BIT  4,5:  SELECTED BANK (0:BANK 0  1:BANK 1)                         %
  &BIT  6:    BASIC OPTION SELECT                                        %
  #BIT  7:    ADDRESS 00-4D ACCESS MASK (0:DISABLE  1:ENABLE)
¥7F. SOFTWARE SWITCH 2  (VALUE OF DIP SWITCH AND PRINTER SWITCH)
  &BIT  0:    DIP SWITCH 1
  &BIT  1:    DIP SWITCH 2
  &BIT  2:    DIP SWITCH 3
  &BIT  3:    DIP SWITCH 4
  #BIT  4:    ENABLE BIT0 - BIT3 (0:DISABLE  1:ENABLE)                   %
  #BIT  5:    ENABLE BIT 7 (0:DISABLE  1:ENABLE)                         %
  &BIT  6:    NOT USED
  &BIT  7:    PRINTER ON/OFF SWITCH
RAM MEMORY MAP 2

(A) INTERRUPT JUMP ADDRESS(¥100 - ¥11D)
#¥100 - 102: CLOCK INTERRUPT JUMP
#¥103 - 105: EXTERNAL PORT INTERRUPT (IRQ1)
#¥106 - 108: TRAP
#¥109 - 10B: SCI INTERRUPT
#¥10C - 10E: TOF INTERRUPT
#¥10F - 111: OCF INTERRUPT
#¥112 - 114: ICF INTERRUPT
#¥115 - 117: IRQ1 INTERRUPT
&¥118 - 11A: SWI
&¥11B - 11D: NMI
```

```
(B) VECTOR IN RAM. (¥11E - ¥12D)
&¥11E - 11F: ADDRESS OF ¥E0 - ¥FF CHRACTER FONT TABLE
#¥120 - 121: WHEN PUSHED BREAK KEY, JUMP ADDRESS(NOT IN BASIC MODE)
#¥122 - 123: WHEN PUSHED MENU KEY, JUMP ADDRESS (NOT IN BASIC MODE)
#¥124 - 125: WHEN PUSHED PAUSE KEY, JUMP ADDRESS(NOT IN BASIC MODE)
#¥126 - 127: WHEN PUSHED CTRL/PF3 KEY, JUMP ADDRESS                    %
#¥128 - 129: WHEN PUSHED CTRL/PF4 KEY, JUMP ADDRESS                    %
#¥12A - 12B: WHEN PUSHED CTRL/PF5 KEY, JUMP ADDRESS                    %
&¥12C - 12D: RAM BOTTOM ADDRESS                                        %

(C) ROM CASSETTE DATA COUNTER                                          %
 ¥12E - 12F: NUMBER OF BYTES IN THE CURRENT ROM FILE.                  @

(D) VECTOR BY WAKE UP  (¥130 - ¥133)                                   %
#¥130 - 131: POWER ON BY WAKE-UP, CALLED PROCEDURE ADDRESS              %
#¥132 - 133: POWER OFF, CALLED PROCEDURE ADRESS                        %

(E) VETOR BY BASIC
#¥134 - 135: TOP ADDRESS OF BASIC PROGRAM                              %
#¥136 - 137: LAST ADDRESS OF BASIC PROGRAM                             %
#¥138 - 139: ENTRY POINT TO GARBAGE COLLECTOR                          %
                                                                       %
(F) HEADER BY MENU                                                     %
#¥13A - 13B:EXIST HEADER FLAG IN EACH ROM                              %
#¥13C - 13F:HEADING BY USER'S PROGRAM (USED BY MENU)                   %

(G) OTHER WORK                                                         @
 ¥140 - ¥18F(188): KEY BOARD ROUTINE WORK AREA                         %
 ¥190 - ¥1AE: MICRO PRINTER WORK AREA                                  @
 ¥1AF - ¥1C3: RS232C WORK AREA                                         @
 ¥1C4 - ¥1D4: SERIAL COMMUNICATION WORK AREA                           @
 ¥1D5 - ¥1EB: EXTERNAL CASSETTE WORK AREA                              @
 ¥1EC - ¥207: INTERNAL MICRO CASSETTE WORK AREA                        @
 ¥208 - ¥20E: ROM CARTRIDGE WORK AREA                                  @
 ¥20F - ¥21A: BINARY MEMORY DUMP/LOAD WORK AREA                        @
 ¥21B       : FOR MICRO CASSETTE                                       @
 ¥21C - ¥21F: SPARE (NOT USED)                                         @
 ¥220 - ¥29F: SCREEN WORK AREA                                         @
 ¥2A0 - ¥2CF: MONITOR WORK AREA                                        @
 ¥2D0 - ¥323: EXTERNAL CASSETTE HEADER                                 @
 ¥324 - ¥377: MICRO CASSETTE HEADER                                    @
 ¥380 - ¥47C: BUFFER USED BY MONITOR R/W ROUTINE                       @
 ¥47D - ¥4AF: STACK (DEFAULT)                                          @
```

```
KEY BOARD ROUTINE

REVISION     B-1
CREATED      12.17.1981
             K. AKAHANE
UPDATE       06.07.1982
UPDATE       06.24.1982
UPDATE       11.16.1982
FILE NAME    KEY HX1D

UPDATE MARK = 72 COLUMN '%' TO REVISION A
UPDATE MARK = 72 COLUMN '@' TO REVISION B

 KEY INPUT PROCEDURE
  AFTER 'OPEN KEY', PUSHED KEY CODES ARE AUTOMATICALY PUSHED INTO KEY
 STACK BY PUSHED KEY INTERRUPT OR INTERVAL TIMER INTERRUPT. THE KEY
 STACK HAS EIGHT BYTES MAX SIZE.

KEY AUTO REPEAT TIME
 SAMPLING TIME 20 M SEC
 FIRST          800 M SEC (INTERRUPT COUNT = 40 TIMES)
 AFTER SECOND   120 M SEC (INTERRUPT COUNT = 6 TIMES)

USE FREE RUNNING COUNTER
USE OCF INTERRUPT (20 M SEC)
KEY STACK = 8 BYTES MAX

AUTO REPEAT CODE
 1. ALPHA NUMERIC KEY
 2. CURSOR LEFT/RIGHT, CURSOR UP/DOWN
 3. SCROLL UP/DOWN
 4. DEL/INS
 5. CLEAR/HOME
 6. HTAB
 7. PAPER FEED (IS NOT ACCEPTED BY KEYIN ROUTINE)
NOT REPEAT KEY
 1. FUNCTION KEY
 2. BREAK
 3. HELP
 4. PAUSE

SPECIAL FUNCTION
 1. CTRL/F1  MICRO CASSETTE MANUAL FUNCTION
 2. CTRL/F2  SCREEN COPY

  MICRO CASSETTE MANUAL FUNCTION COMMAND
   1. COUNTER RESET
   2. REWIND
   3. PLAY
   4. FAST FEED
   5. STOP
   6. QUIT

DIP SWITCH
 LOW 3 BITS (US VERSION)                 LOW 3 BITS (EUROPE VERSION)
```

BIT2	BIT1	BIT0	
1	1	1	USA
1	1	0	FRANCE
1	0	1	GERMANY
1	0	0	ENGLAND
0	1	1	DENMARK
0	1	0	SWEDEN
0	0	1	ITALY

BIT2	BIT1	BIT0	
1	1	1	NORWAY
1	1	0	FRANCE
1	0	1	GERMANY
1	0	0	SWEDEN
0	1	1	DENMARK
0	1	0	FRANCE (ASCII CODE)
0	0	1	GERMANY(ASCII CODE)

0 0 0 SPAIN 0 0 0 SWEDEN (ASCII CODE)

LIST OF GENERATED CODE FROM KEYBOARD

```
NOTE.  G/ = 'GRAPH'
       C/ = 'CONTROL'
       S. = 'NOT CAPITAL LETTER'
       (| = LEFT SQUARE BRACKET
       )| = RIGHT SQUARE BRACKET
       (" = LEFT CURLY BRACKET
       )" = RIGHT CURLY BRACKET
       ¥  = DOLLER SIGN
       ¬  = REVERSE /    (IN KANA VERSION, CHANGED TO '¥' CHARACTER)
       || = UP ARROW HEAD
       |' = EXCLAMATION MARK
       S/ = KANA KOMOJI
```

KEY INPUT CODES (NORMAL OR SHIFT)

	0	1	2	3	4	5	6	7
0			SP	0	@	P		S.P
1			|'	1	A	Q	S.A	S.Q
2			"	2	B	R	S.B	S.R
3			#	3	C	S	S.C	S.S
4			¥	4	D	T	S.D	S.T
5			%	5	E	U	S.E	S.U
6			&	6	F	V	S.F	S.V
7			'	7	G	W	S.G	S.W
8			(	8	H	X	S.H	S.X
9			)	9	I	Y	S.I	S.Y
A			*	:	J	Z	S.J	S.Z
B			+	;	K	(|	S.K	('
C			,	<	L	¬	S.L	|
D			-	=	M	)|	S.M	)'
E			.	>	N	||	S.N	
F			/	?	O	_	S.O	

KEY INPUT CODES (CONTROL CODES) (CONTROL KEY + KEY CODE)

	0	1	2	3	4	5	6	7	
0	C/@	C/P							%
1	C/A	C/Q							%
2	C/B	C/R							%
3	C/C	C/S							%
4	C/D	C/T							%
5	C/E	C/U							%
6	C/F	C/V							%
7	C/G	C/W							%
8	C/H	C/X							%
9	C/I	C/Y							%
A	C/J	C/Z							%
B	C/K	C/(|							%
C	C/L	C/¬							%
D	C/M	C/)|							%
E	C/N	C/||		C/G/)|					%
F	C/O	C/_		C/G/¬					%
									%

	8	9	A	B	C	D	F	F
0								C/0
1							C/|'	C/1
2							C/"	C/2
3							C/#	C/3
4							C/¥	C/4
5							C/%	C/5
6							C/&	C/6
7							C/'	C/7
8							C/(	C/8
9							C/)	C/9
A							C/*	C/:
B							C/+	C/;
C							C/,	C/<
D							C/-	C/=
E							C/.	C/>
F							C//	C/?

NOTE. CONTROL KEY IS EFFECTIVE TO ¥20 - ¥5F CODE, THESE CODES ARE SUBTRACTED ¥40.

GRAPH MODE CODE

	0	1	2	3	4	5	6	7
0							G/(| (US)	
1								
2								
3								
4								
5								
6								
7								
8								
9								
A								
B								
C								
D								
E								G/)| (US)
F								G/¬

	8	9	A	B	C	D	E	F
0	G/S	G/U	G/-				G/0	
1	G/X	G/I					G/1	
2	G/W	G/O					G/2	
3	G/D	G/P					G/3	
4	G/A	G/@					G/4	
5	G/T	G/K					G/5	
6	G/R	G/V					G/6	
7	G/Q	G/,					G/7	
8	G/E	G/M					G/8	
9	G/Z	G/N					G/9	
A	G/C	G/B					G/(| (EU)	
B	G/J	G/;					G/)| (EU)	
C	G/F	G/.						
D	G/G	G/:						
E	G/H	G//						
F	G/Y	G/L						

KANA MODE.

	8	9	A	B	C	D	E	F
0				CHUON	TA	MI		
1			MARU	A	CHI	MU		
2			SHIKAK	I	TSU	ME		
3			SHUKAK	U	TE	MO		
4			TEN	E	TO	YA		
5			DOT	O	NA	YU		
6			WO	KA	NI	YO		
7			S/A	KI	NU	RA		
8			S/I	KU	NE	RI		
9			S/U	KE	NO	RU		
A			S/E	KO	HA	RE		
B			S/O	SA	HI	RO		
C			S/YA	SHI	FU	WA		
D			S/YU	SU	HE	N		
E			S/YO	SE	HO	DAKU		
F			S/TSU	SO	MA	HANDAKU		

NUMERIC MODE

	0	1	2	3	4	5	6	7
0				0,(M)				
1				1,(J)				
2				2,(K)				
3				3,(L)				
4				4,(U)				
5				5,(I)				
6				6,(O)				
7				7				
8				8				
9				9				
A			*					
B			+					
C			,					
D			-					
E			.					
F			/	¬				

NOTE. * + , - . / CODES ARE NOT SHIFT.

SPECIAL KEY CODE

	0	1	2
0		SCREEN UP	SPACE
1		SCRREEN DOWN	
2		INS	
3		LEFT SCROLL(CTRL/CUR LEFT)	
4	RIGHT SCROLL(CTRL/CUR RIGHT)		
5			
6			
7			
8	DEL		
9	HTAB		
A			
B	HOME	ESC(SHIFT/PAUSE)	
C	CLEAR	CURSOR RIGHT	
D	CR	CURSOR LEFT	
E		CURSOR UP	
F		CURSOR DOWN	

NOTE. ALL CONTROL KEYS ARE EFFECTIVE IN ANY MODE.

```
OTHER CODES
1. FUNCTION KEY CODE
  F1: ¥FE, ¥F1  (2 BYTES CODES)
  F2: ¥FE, ¥F2
     :
     :
  F10: ¥FE, ¥FA
 NOTE.  AS ¥FE IS CODE OF FUNCTION KEY, ¥FE AND ONE ANY CODE(UNDEFINED)
       ARE PUSED INTO KEYSTACK WHEN WE PRESS CTRL/>.

2. MENU KEY
  ¥FC  (ONLY IN BASIC MODE)

  MODE TRANSFER
          (KANA VERSION)
                                         SHIFT/KANA
 |---------------------------|  |------------------------------|
 |                           |  |          CAPS                |
 |  |------------------------|--|--------------------------|   |
 |  |     SHIFT/KANA         |  V         CAPS             |   |
 |  | |------------------>|---------|--------------------| |   |
 |  | |                   | GRAPH   |                    | |   |
 |  | |   |---------------|---------|<--------------|    | |   |
 |  | |   |    KANA          A  |      SHIFT/KANA   |    | |   |
 |  | |   |                  |  |      KANA         |    | |   |
 |  | |   |  |---------------|--|-----------------| |    | |   |
 |  | |   |  |               |  |                 | |    | |   |
 |  | |   |  |               |  |                 | |    | |   |
 |    |   V  V    KANA       |  V       CAPS      | |    V V   |
 |  |---------|  ----->   |---------|  <------   |---------|   |
 |  | KANA    |           | NORMAL  |            | CAPS L  |   |
 |  |---------|  <----- . |---------| ------>    |---------|   |
 |    |    A      KANA      |     A    CAPS         A  |       |
 |    |    |                |     |                 |  |       |
 |    |    |KANA       NUM  |     |  NUM        CAPS|  | NUM   |
 |    |    |                |     |                 |  |       |
 |    |    |                |     |                 |  |       |
 |    |    |                V     |                 |  |       |
 |    |    |------------  |---------|   -----------/   |       |
 |    |                   | NUMERIC |                  |       |
 |    |---------------->  |---------|   <-------------/        |
 |         NUM              A     |                            |
 |--------------------------|     |----------------------------/
           NUM
```

```
    (ASCII VERSION)        |---------|

                           | GRAPH   |
                           |---------|
                                |
                                | (ON PRESSING GRAPH)
                   NUM          |      CAPS
    |---------|  ----->    |---------|  <------   |---------|
    | NUMERIC |            | NORMAL  |            | CAPS L  |
    |---------|  <-----    |---------| ------>    |---------|
      |    A       NUM                    CAPS        |     A
      |    |                                       |     |
      |    |               NUM                     |     |
      |    |---------------------------------------|     |
      |                                                  |
      |--------------------------------------------------/
                           CAPS
```

```
KEY CALLING SEQUENCE
1. READ KEY-STATUS
   SUBROUTINE NAME   KEYSTS
   ON ENTRY PARAMETER NONE
   ON EXIT      (A):CHARACTER NUMBER WITCH IS STORED IN THE STACK
                (C):I/O ERROR STATUS  0:NORMAL  1:ERROR
   PRESERVE  B,X

2. READ CODE FROM KEY STACK
   SUBROUTINE NAME   KEYIN
   ON ENTRY PARAMETER NONE
   ON EXIT   (A):READ CODE
                  WHEN (A) IS ¥FE, THE CONTENT OF (B) IS FUNCTION CODE.
                       (F1:¥F1  F2:¥F2 ...  F10:¥FA)
             (C): I/O ERROR STATUS   0:NORMAL   1:ERROR
   PRESERVE X, B(WHEN CONTENTS OF A IS NOT ¥FE)
   IF KEY STACK IS EMPTY, MAIN CPU WILL BE SLEEP UNTIL KEY ACCEPTED.

SPECIAL ROUTINE
1. INITIALIZE KEY
   SUBROUTINE NAME  KYINIT
   ON ENTRY  PARAMETER  NONE
   ON EXIT   PARAMETER  NONE
   REGISTER PRESERVE    NONE

2. SET DATA TO INITIAL-READ-KEY-STACK
   SUBROUTINE NAME  KYSTST
   ON ENTRY  (X):DATA CHARACTERS ADDRESS
                   CHARACTERS:KEY CODES. LAST CODE IS ¥FF.
             (B):CHARACTER LENGTH (MAX=16). 0:CLEAR STACK
   ON EXIT
                    (C):  0:NORMAL   1:ERROR
```

```
WORK AREA MAP
 NOTE. ADDRESS WITH # SIGN IS INITIALIZED BY KEY INITIALIZE (POWER ON)

KEY BOARD ROUTINE WORK AREA (¥130 - ¥177)
 #¥140         :KEY STACK MAX SIZE (1 < 8)  (INITIAL 8)                ə
 # 141         :AUTO REPEAT INTERVAL TIME (FIRST)                      ə
 # 142         :AUTO REPEAT INTERVAL TIME (SECOND)                     ə
 # 143 - 144:SAMPLING TIME                                             ə
   145 - 14E:NEW KEY TABLE                                             ə
   14F - 158:OLD KEY TABLE (LAST VALUE)                                ə
   159 - 162:CHECK KEY TABLE (TO FIND PRESSED KEY)                     ə
 # 163 - 164:ADDRESS OF INITIAL KEY STACK                              ə
 # 165         :INITIAL KEY READ MODE FLAG (A:EXISTA DATA B:ON READING) ə
   166         :DATA NUMBER IN THE INITAIL KEY STACK                   ə
 # 167         :READ CHARACTER COUNTER FROM INITIAL KEY STACK          ə
   168         :PUSHED DATA COUNTER IN THE READ KEY STACK              ə
 # 169         :KEY MODE                                               ə
 # 16A         :KEY ACCEPT MODE                                        ə
   16B         :AUTO REPEAT SAMPLING COUNTER                           ə
   16C         :REPEAT KEY MATRIX POSITION                             ə
   16D - 16E:READ KEY CODE                                             ə
   16F - 180:INITIAL KEY STACK (18 BYTES)                              ə
   181 - 188:READ KEY STACK (8 BYTES DEFAULT)                          ə
                                                                       ə
```

```
CHARACTER FONT

REVISION     A-1
CREATED      11.17.1982
             K. AKAHANE
FILE NAME    FONT HX1D

  NOTE.  CHRACTERS WHICH HAVE ¥A0 - ¥DF CODES ARE PROVIDED TO
         NIPPON VERSION.


 COUNTRY CHARACTER FONT (5 * 7 DOTS MATRIX  .=OFF  O=ON)
             XD
           O O O . .     . . . . .     . . . . .     . . . . .     LSB
           O . . . .     . . . . .     . . . . .     . . . . .     BIT 1
 ¥0X       O . . . .     . . . . .     . . . . .     . . . . .
           O . O O O     . . . . .     . . . . .     . . . . .
           O O O . O     . . . . .     . . . . .     . . . . .
           . . O O .     . . . . .     . . . . .     . . . . .
           . . O . O     . . . . .     . . . . .     . . . . .     BIT 6
```

```
            X0          X1          X2          X3
        . . . . .   . . 0 . .   . 0 . 0 .   . 0 . 0 .    LSB
        . . . . .   . . 0 . .   . 0 . 0 .   . 0 . 0 .    BIT 1
 ¥2X    . . . . .   . . 0 . .   . 0 . 0 .   0 0 0 0 0
        . . . . .   . . 0 . .   . . . . .   . 0 . 0 .
        . . . . .   . . . . .   . . . . .   0 0 0 0 0
        . . . . .   . . . . .   . . . . .   . 0 . 0 .
        . . . . .   . . 0 . .   . . . . .   . 0 . 0 .    BIT 6

            X4          X5          X6          X7
        . . 0 . .   0 0 . . .   . 0 0 . .   . 0 0 . .    LSB
        . 0 0 0 0   0 0 . . 0   0 . . 0 .   . . 0 . .    BIT 1
 ¥2X    0 . 0 . .   . . . 0 .   0 . 0 . .   . 0 . . .
        . 0 0 0 .   . . 0 . .   . 0 . . .   . . . . .
        . . 0 . 0   . 0 . . .   0 . 0 . 0   . . . . .
        0 0 0 0 .   0 . . 0 0   0 . . 0 .   . . . . .
        . . 0 . .   . . . 0 0   . 0 0 . 0   . . . . .    BIT 6

            X8          X9          XA          XB
        . . . 0 .   . 0 . . .   . . 0 . .   . . . . .    LSB
        . . 0 . .   . . 0 . .   0 . 0 . 0   . . 0 . .    BIT 1
 ¥2X    . 0 . . .   . . . 0 .   . 0 0 0 .   . . 0 . .
        . 0 . . .   . . . 0 .   . 0 0 0 .   0 0 0 0 0
        . 0 . . .   . . . 0 .   0 . 0 . 0   . . 0 . .
        . . 0 . .   . . 0 . .   . . 0 . .   . . 0 . .
        . . . 0 .   . 0 . . .   . . . . .   . . . . .    BIT 6

            XC          XD          XE          XF
        . . . . .   . . . . .   . . . . .   . . . . .    LSB
        . . . . .   . . . . .   . . . . .   . . . . 0    BIT 1
 ¥2X    . . . . .   . . . . .   . . . . .   . . . 0 .
        . . . . .   0 0 0 0 0   . . . . .   . . 0 . .
        . 0 0 . .   . . . . .   . . . . .   . 0 . . .
        . . 0 . .   . . . . .   0 0 . . .   0 . . . .
        . 0 . . .   . . . . .   0 0 . . .   . . . . .    BIT 6

            X0          X1          X2          X3
        . 0 0 0 .   . . 0 . .   . 0 0 0 .   0 0 0 0 0    LSB
        0 . . . 0   . 0 0 . .   0 . . . 0   . . . 0 .    BIT 1
 ¥3X    0 . . 0 0   . . 0 . .   . . . . 0   . . 0 . .
        0 . 0 . 0   . . 0 . .   . . . 0 .   . . . 0 .
        0 0 . . 0   . . 0 . .   . . 0 . .   . . . . 0
        0 . . . 0   . . 0 . .   . 0 . . .   0 . . . 0
        . 0 0 0 .   . 0 0 0 .   0 0 0 0 0   . 0 0 0 .    BIT 6

            X4          X5          X6          X7
        . . . 0 .   0 0 0 0 0   . . 0 0 .   0 0 0 0 0    LSB
        . . 0 0 .   0 . . . .   . 0 . . .   . . . . 0    BIT 1
 ¥3X    . 0 . 0 .   0 0 0 0 .   0 . . . .   . . . 0 .
        0 . . 0 .   . . . . 0   0 0 0 0 .   . . 0 . .
        0 0 0 0 0   . . . . 0   0 . . . 0   . 0 . . .
        . . . 0 .   0 . . . 0   0 . . . 0   . 0 . . .
        . . . 0 .   . 0 0 0 .   . 0 0 0 .   . 0 . . .    BIT 6
```

```
            X8          X9          XA          XB
        . 0 0 0 .   . 0 0 0 .   . . . . .   . . . . .    LSB
        0 . . . 0   0 . . . 0   . 0 0 . .   . 0 0 . .    BIT 1
 ¥3X    0 . . . 0   0 . . . 0   . 0 0 . .   . 0 0 . .
        . 0 0 0 .   . 0 0 0 0   . . . . .   . . . . .
        0 . . . 0   . . . . 0   . 0 0 . .   . 0 0 . .
        0 . . . 0   . . . 0 .   . 0 0 . .   . . 0 . .
        . 0 0 0 .   . 0 0 . .   . . . . .   . 0 . . .    BIT 6

            XC          XD          XE          XF
        . . . 0 .   . . . . .   . 0 . . .   . 0 0 0 .    LSB
        . . 0 . .   . . . . .   . . 0 . .   0 . . . 0    BIT 1
 ¥3X    . 0 . . .   0 0 0 0 0   . . . 0 .   . . . . 0
        0 . . . .   . . . . .   . . . . 0   . . . 0 .
        . 0 . . .   0 0 0 0 0   . . . 0 .   . . 0 . .
        . . 0 . .   . . . . .   . . 0 . .   . . . . .
        . . . 0 .   . . . . .   . 0 . . .   . . 0 . .    BIT 6

            X0          X1          X2          X3
        . 0 0 0 .   . . 0 . .   0 0 0 0 .   . 0 0 0 .    LSB
        0 . . . 0   . 0 . 0 .   0 . . . 0   0 . . . 0    BIT 1
 ¥4X    0 . 0 0 0   0 . . . 0   0 . . . 0   0 . . . .
        0 . 0 . 0   0 . . . 0   0 0 0 0 .   0 . . . .
        0 . 0 0 0   0 0 0 0 0   0 . . . 0   0 . . . .
        0 . . . .   0 . . . 0   0 . . . 0   0 . . . 0
        . 0 0 0 .   0 . . . 0   0 0 0 0 .   . 0 0 0 .    BIT 6

            X4          X5          X6          X7
        0 0 0 0 .   0 0 0 0 0   0 0 0 0 0   . 0 0 0 .    LSB
        0 . . . 0   0 . . . .   0 . . . .   0 . . . 0    BIT 1
 ¥4X    0 . . . 0   0 . . . .   0 . . . .   0 . . . .
        0 . . . 0   0 0 0 0 .   0 0 0 0 .   0 . 0 0 0
        0 . . . 0   0 . . . .   0 . . . .   0 . . . 0
        0 . . . 0   0 . . . .   0 . . . .   0 . . . 0
        0 0 0 0 .   0 0 0 0 0   0 . . . .   . 0 0 0 0    BIT 6

            X8          X9          XA          XB
        0 . . . 0   . 0 0 0 .   . . 0 0 0   0 . . . 0    LSB
        0 . . . 0   . . 0 . .   . . . 0 .   0 . . 0 .    BIT 1
 ¥4X    0 . . . 0   . . 0 . .   . . . 0 .   0 . 0 . .
        0 0 0 0 0   . . 0 . .   . . . 0 .   0 0 . . .
        0 . . . 0   . . 0 . .   . . . 0 .   0 . 0 . .
        0 . . . 0   . . 0 . .   0 . . 0 .   0 . . 0 .
        0 . . . 0   . 0 0 0 .   . 0 0 . .   0 . . . 0    BIT 6

            XC          XD          XE          XF
        0 . . . .   0 . . . 0   0 . . . 0   . 0 0 0 .    LSB
        0 . . . .   0 0 . 0 0   0 . . . 0   0 . . . 0    BIT 1
 ¥4X    0 . . . .   0 . 0 . 0   0 0 . . 0   0 . . . 0
        0 . . . .   0 . 0 . 0   0 . 0 . 0   0 . . . 0
        0 . . . .   0 . . . 0   0 . . 0 0   0 . . . 0
        0 . . . .   0 . . . 0   0 . . . 0   0 . . . 0
        0 0 0 0 0   0 . . . 0   0 . . . 0   . 0 0 0 .    BIT 6
```

```
            X0           X1           X2           X3
        0 0 0 0 .    . 0 0 0 .    0 0 0 0 .    . 0 0 0 .    LSB
        0 . . . 0    0 . . . 0    0 . . . 0    0 . . . 0    BIT 1
 ¥5X    0 . . . 0    0 . . . 0    0 . . . 0    0 . . . .
        0 0 0 0 .    0 . . . 0    0 0 0 0 .    . 0 0 0 .
        0 . . . .    0 . 0 . 0    0 . 0 . .    . . . . 0
        0 . . . .    0 . . 0 .    0 . . 0 .    0 . . . 0
        0 . . . .    . 0 0 . 0    0 . . . 0    . 0 0 0 .    BIT 6

            X4           X5           X6           X7
        0 0 0 0 0    0 . . . 0    0 . . . 0    0 . . . 0    LSB
        . . 0 . .    0 . . . 0    0 . . . 0    0 . . . 0    BIT 1
 ¥5X    . . 0 . .    0 . . . 0    0 . . . 0    0 . . . 0
        . . 0 . .    0 . . . 0    0 . . . 0    0 . 0 . 0
        . . 0 . .    0 . . . 0    0 . . . 0    0 . 0 . 0
        . . 0 . .    0 . . . 0    . 0 . 0 .    0 . 0 . 0
        . . 0 . .    . 0 0 0 .    . . 0 . .    . 0 . 0 .    BIT 6

            X8           X9           XA           XB
        0 . . . 0    0 . . . 0    0 0 0 0 0    . 0 0 0 .    LSB
        0 . . . 0    0 . . . 0    . . . . 0    . 0 . . .    BIT 1
 ¥5X    . 0 . 0 .    0 . . . 0    . . . 0 .    . 0 . . .
        . . 0 . .    . 0 . 0 .    . . 0 . .    . 0 . . .
        . 0 . 0 .    . . 0 . .    . 0 . . .    . 0 . . .
        0 . . . 0    . . 0 . .    0 . . . .    . 0 . . .
        0 . . . 0    . . 0 . .    0 0 0 0 0    . 0 0 0 .    BIT 6

            XC           XD           XE           XF
        0 . . . 0    . 0 0 0 .    . . 0 . .    . . . . .    LSB
        . 0 . 0 .    . . . 0 .    . 0 . 0 .    . . . . .    BIT 1
 ¥5X    0 0 0 0 0    . . . 0 .    0 . . . 0    . . . . .
        . . 0 . .    . . . 0 .    . . . . .    . . . . .
        0 0 0 0 0    . . . 0 .    . . . . .    . . . . .
        . . 0 . .    . . . 0 .    . . . . .    . . . . .
        . . 0 . .    . 0 0 0 .    . . . . .    0 0 0 0 0    BIT 6

            X0           X1           X2           X3
        . 0 0 . .    . . . . .    0 . . . .    . . . . .    LSB
        . 0 . . .    . . . . .    0 . . . .    . . . . .    BIT 1
 ¥6X    . . 0 . .    0 0 0 0 .    0 0 0 0 .    . 0 0 0 0
        . . . . .    . . . . 0    0 . . . 0    0 . . . .
        . . . . .    . 0 0 0 0    0 . . . 0    0 . . . .
        . . . . .    0 . . . 0    0 . . . 0    0 . . . .
        . . . . .    . 0 0 0 0    0 0 0 0 .    . 0 0 0 0    BIT 6

            X4           X5           X6           X7
        . . . . 0    . . . . .    . . . 0 0    . . . . .    LSB
        . . . . 0    . . . . .    . . 0 . .    . . . . .    BIT 1
 ¥6X    . 0 0 0 0    . 0 0 0 .    0 0 0 0 0    . 0 0 0 0
        0 . . . 0    0 . . . 0    . . 0 . .    0 . . . 0
        0 . . . 0    0 0 0 0 0    . . 0 . .    . 0 0 0 0
        0 . . . 0    0 . . . .    . . 0 . .    . . . . 0
        . 0 0 0 0    . 0 0 0 0    . . 0 . .    . 0 0 0 .    BIT 6
```

```
            X8           X9           XA           XB
        0 . . . .    . . 0 . .    . . . 0 .    0 . . . .    LSB
        0 . . . .    . . . . .    . . . . .    0 . . . .    BIT 1
 ¥6X    0 0 0 0 .    . 0 0 . .    . . 0 0 .    0 . . 0 .
        0 . . . 0    . . 0 . .    . . . 0 .    0 . 0 . .
        0 . . . 0    . . 0 . .    . . . 0 .    0 0 . . .
        0 . . . 0    . . 0 . .    0 . . 0 .    0 . 0 . .
        0 . . . 0    . 0 0 0 .    . 0 0 . .    0 . . 0 .    BIT 6

            XC           XD           XE           XF
        . 0 0 . .    . . . . .    . . . . .    . . . . .    LSB
        . . 0 . .    . . . . .    . . . . .    . . . . .    BIT 1
 ¥6X    . . 0 . .    0 0 0 0 .    0 0 0 0 .    . 0 0 0 .
        . . 0 . .    0 . 0 . 0    0 . . . 0    0 . . . 0
        . . 0 . .    0 . 0 . 0    0 . . . 0    0 . . . 0
        . . 0 . .    0 . 0 . 0    0 . . . 0    0 . . . 0
        . 0 0 0 .    0 . 0 . 0    0 . . . 0    . 0 0 0 .    BIT 6

            X0           X1           X2           X3
        . . . . .    . . . . .    . . . . .    . . . . .    LSB
        . . . . .    . . . . .    . . . . .    . . . . .    BIT 1
 ¥7X    0 0 0 0 .    . 0 0 0 0    0 . 0 0 .    . 0 0 0 0
        0 . . . 0    0 . . . 0    0 0 . . .    0 . . . .
        0 0 0 0 .    . 0 0 0 0    0 . . . .    . 0 0 0 .
        0 . . . .    . . . . 0    0 . . . .    . . . . 0
        0 . . . .    . . . . 0    0 . . . .    0 0 0 0 .    BIT 6

            X4           X5           X6           X7
        . . 0 . .    . . . . .    . . . . .    . . . . .    LSB
        . . 0 . .    . . . . .    . . . . .    . . . . .    BIT 1
 ¥7X    0 0 0 0 0    0 . . . 0    0 . . . 0    0 . . . 0
        . . 0 . .    0 . . . 0    0 . . . 0    0 . . . 0
        . . 0 . .    0 . . . 0    0 . . . 0    0 . 0 . 0
        . . 0 . .    0 . . . 0    . 0 . 0 .    0 . 0 . 0
        . . . 0 0    . 0 0 0 0    . . 0 . .    . 0 . 0 .    BIT 6

            X8           X9           XA           XB
        . . . . .    . . . . .    . . . . .    . . 0 0 .    LSB
        . . . . .    . . . . .    . . . . .    . 0 . . .    BIT 1
 ¥7X    0 . . . 0    0 . . . 0    0 0 0 0 0    . 0 . . .
        . 0 . 0 .    0 . . . 0    . . . 0 .    0 . . . .
        . . 0 . .    . 0 0 0 0    . . 0 . .    . 0 . . .
        . 0 . 0 .    . . . . 0    . 0 . . .    . 0 . . .
        0 . . . 0    . 0 0 0 .    0 0 0 0 0    . . 0 0 .    BIT 6

            XC           XD           XE           XF
        . . 0 . .    . 0 0 . .    . 0 . . .    . . . . .    LSB
        . . 0 . .    . . . 0 .    0 . 0 . 0    . . . . .    BIT 1
 ¥7X    . . 0 . .    . . . 0 .    . . . 0 .    . . . . .
        . . . . .    . . . . 0    . . . . .    . . . . .
        . . 0 . .    . . . 0 .    . . . . .    . . . . .
        . . 0 . .    . . . 0 .    . . . . .    . . . . .
        . . 0 . .    . 0 0 . .    . . . . .    . . . . .    BIT 6
```

GRAPHIC CHARACTER (6 * 8 COTS MATRIX, .=OFF 0=ON)

```
            X0             X1             X2             X3
        . . 0 . . .    . . 0 . . .    . . . . . .    . . 0 . . .    LSB
        . . 0 . . .    . . 0 . . .    . . . . . .    . . 0 . . .    BIT 1
¥8X     . . 0 . . .    . . 0 . . .    . . . . . .    . . 0 . . .
        0 0 0 0 0 0    0 0 0 0 0 0    0 0 0 0 0 0    0 0 0 . . .
        . . 0 . . .    . . . . . .    . . 0 . . .    . . 0 . . .
        . . 0 . . .    . . . . . .    . . 0 . . .    . . 0 . . .
        . . 0 . . .    . . . . . .    . . 0 . . .    . . 0 . . .    BIT 6
        . . 0 . . .    . . . . . .    . . 0 . . .    . . 0 . . .    MSB

            X4             X5             X6             X7
        . . 0 . . .    . . . . . .    . . 0 . . .    . . 0 . . .
        . . 0 . . .    . . . . . .    . . 0 . . .    . . 0 . . .
¥8X     . . 0 . . .    . . . . . .    . . 0 . . .    . . 0 . . .
        . . 0 0 0 0    0 0 0 0 0 0    . . 0 . . .    . . 0 0 0 0
        . . 0 . . .    . . . . . .    . . 0 . . .    . . 0 . . .
        . . 0 . . .    . . . . . .    . . 0 . . .    . . 0 . . .
        . . 0 . . .    . . . . . .    . . 0 . . .    . . 0 . . .
        . . 0 . . .    . . . . . .    . . 0 . . .    . . 0 . . .

            X8             X9             XA             XB
        . . 0 . . .    . . 0 . . .    . . 0 . . .    . 0 . 0 . 0
        . . 0 . . .    . . 0 . . .    . . 0 . . .    0 . 0 . 0 .
¥8X     . . 0 . . .    . . 0 . . .    . . 0 . . .    . 0 . 0 . 0
        0 0 0 . . .    . . 0 0 0 0    0 0 0 . . .    0 . 0 . 0 .
        . . 0 . . .    . . 0 . . .    . . 0 . . .    . 0 . 0 . 0
        . . 0 . . .    . . 0 . . .    . . 0 . . .    0 . 0 . 0 .
        . . 0 . . .    . . 0 . . .    . . 0 . . .    . 0 . 0 . 0
        . . 0 . . .    . . 0 . . .    . . 0 . . .    0 . 0 . 0 .

            XC             XD             XE             XF
        0 0 0 0 0 0    . . . . . .    0 0 0 . . .    . . . . . .
        0 0 0 0 0 0    . . . . . .    0 0 0 . . .    . 0 0 0 . .
¥8X     0 0 0 0 0 0    . . . . . .    0 0 0 . . .    0 0 0 0 0 .
        0 0 0 0 0 0    . . . . . .    0 0 0 . . .    0 0 0 0 0 .
        0 0 0 0 0 0    0 0 0 0 0 0    0 0 0 . . .    0 0 0 0 0 .
        0 0 0 0 0 0    0 0 0 0 0 0    0 0 0 . . .    . 0 0 0 . .
        0 0 0 0 0 0    0 0 0 0 0 0    0 0 0 . . .    . . . . . .
        0 0 0 0 0 0    0 0 0 0 0 0    0 0 0 . . .    . . . . . .

            X0             X1             X2             X3
        . . . . . .    . . 0 . . .    . . . . . .    . . . . . .
        . 0 0 0 . .    . 0 0 0 . .    . 0 . . 0 .    . . 0 . . .
¥9X     0 . . . 0 .    0 0 0 0 0 .    0 0 0 0 0 0    . 0 0 0 . .
        0 . . . 0 .    0 0 0 0 0 .    0 0 0 0 0 0    0 0 0 0 0 .
        0 . . . 0 .    0 0 0 0 0 .    0 0 0 0 0 0    0 0 0 0 0 .
        . 0 0 0 . .    . . 0 . . .    . 0 0 0 0 .    . 0 0 0 . .
        . . . . . .    . 0 0 0 . .    . . 0 0 . .    . . 0 . . .
        . . . . . .    . . . . . .    . . . . . .    . . . . . .

            X4             X5             X6             X7
        . . . . . .    . . 0 . . .    . . . . . .    . . . . . .
        . 0 0 0 . .    . . 0 0 . .    . . . . . .    . . 0 . . .
¥9X     . 0 0 0 . .    . . 0 . 0 .    . 0 0 0 . .    . . 0 . . .
        0 0 0 0 0 .    . . 0 . 0 .    0 . . . 0 .    . 0 0 0 . .
        0 0 0 0 0 .    . 0 0 . 0 .    . 0 0 0 . .    0 0 0 0 0 .
        . . 0 . . .    0 0 0 . . .    0 0 . 0 0 .    . . 0 . . .
        . 0 0 0 . .    0 0 . . . .    0 0 0 0 0 .    . 0 0 0 . .
        . . . . . .    . . . . . .    . . . . . .    . . . . . .
```

```
            X8             X9             XA             XB
        . . . . . .    . . . . . .    . . 0 . . .    . . 0 . . .
        . . . . . .    0 . . . 0 .    0 0 0 0 . .    . 0 0 0 . .
¥9X     . 0 0 0 0 .    0 . . . 0 .    . . 0 . 0 .    0 . 0 . 0 .
        . 0 . 0 0 .    0 . . . 0 .    . . 0 . . .    . . 0 . . .
        0 0 0 0 0 .    . 0 0 0 . .    . . 0 0 . .    . . 0 . . .
        0 0 0 0 0 .    . . 0 . . .    . 0 . 0 0 .    . . 0 . . .
        . 0 . 0 . .    . 0 0 0 . .    0 0 . . 0 .    . . 0 . . .
        . . . . . .    . . . . . .    . . . . . .    . . . . . .

            XC             XD             XE             XF
        . . 0 . . .    . . . . . .    . . . . . .    . . 0 . . .
        . . 0 . . .    0 . . . 0 .    . . 0 . . .    . . 0 . . .
¥9X     . . 0 . . .    . 0 . 0 . .    . . . . . .    0 0 0 0 0 .
        . . 0 . . .    . . 0 . . .    0 0 0 0 0 .    . . 0 . . .
        0 . 0 . 0 .    . 0 . 0 . .    . . . . . .    . . 0 . . .
        . 0 0 0 . .    0 . . . 0 .    . . 0 . . .    . . . . . .
        . . 0 . . .    . . . . . .    . . . . . .    0 0 0 0 0 .
        . . . . . .    . . . . . .    . . . . . .    . . . . . .

           X0             X1             X2             X3
        . . . . .      . . . . .      . . 0 0 0      . . . . .      LSB
        . . . . .      . . . . .      . . 0 . .      . . . . .      BIT 1
¥AX     . . . . .      . . . . .      . . 0 . .      . . . . .
        . . . . .      . . . . .      . . 0 . .      . . 0 . .
        . . . . .      0 0 0 . .      . . . . .      . . 0 . .
        . . . . .      0 . 0 . .      . . . . .      . . 0 . .
        . . . . .      0 0 0 . .      . . . . .      0 0 0 . .      BIT 6
```

```
             X4           X5           X6           X7
          . . . . .    . . . . .    . . . . .    . . . . .        LSB
          . . . . .    . . . . .    0 0 0 0 0    . . . . .        BIT 1
¥AX       . . . . .    . . . . .    . . . . 0    0 0 0 0 0
          . . . . .    . 0 0 . .    0 0 0 0 0    . . . . 0
          . . . . .    . 0 0 . .    . . . 0 .    . . 0 0 .
          . 0 . . .    . . . . .    . . 0 . .    . . 0 . .
          . . 0 . .    . . . . .    . 0 . . .    . 0 . . .        BIT 6

             X8           X9           XA           XB
          . . . . .    . . . . .    . . . . .    . . . . .        LSB
          . . . . .    . . . . .    . . . . .    . . . . .        BIT 1
¥AX       . . . 0 .    . . 0 . .    . . . . .    . . . 0 .
          . . 0 . .    0 0 0 0 0    0 0 0 0 0    0 0 0 0 0
          . 0 0 . .    0 . . . 0    . . 0 . .    . . 0 0 .
          0 . 0 . .    . . . . 0    . . 0 . .    . 0 . 0 .
          . . 0 . .    . . 0 0 .    0 0 0 0 0    0 . . 0 .        BIT 6

             XC           XD           XE           XF
          . . . . .    . . . . .    . . . . .    . . . . .        LSB
          . . . . .    . . . . .    . . . . .    . . . . .        BIT 1
¥AX       . 0 . . .    . . . . .    0 0 0 0 .    . . . . .
          0 0 0 0 0    . 0 0 0 .    . . . 0 .    0 . 0 . 0
          . 0 . . 0    . . . 0 .    0 0 0 0 .    0 . 0 . 0
          . 0 . 0 .    . . . 0 .    . . . 0 .    . . . . 0
          . 0 . . .    0 0 0 0 0    0 0 0 0 .    . . 0 0 .        BIT 6
```

```
             X0           X1           X2           X3
          . . . . .    0 0 0 0 0    . . . . 0    . . 0 . .        LSB
          . . . . .    . . . . 0    . . . 0 .    0 0 0 0 0        BIT 1
¥BX       . . . . .    . . 0 . 0    . . 0 . .    0 . . . 0
          0 0 0 0 0    . . 0 0 .    . 0 0 . .    0 . . . 0
          . . . . .    . . 0 . .    0 . 0 . .    . . . . 0
          . . . . .    . . 0 . .    . . 0 . .    . . . 0 .
          . . . . .    . 0 . . .    . . 0 . .    . . 0 . .        BIT 6

             X4           X5           X6           X7
          . . . . .    . . . 0 .    . 0 . . .    . . 0 . .        LSB
          0 0 0 0 0    0 0 0 0 0    0 0 0 0 0    0 0 0 0 0        BIT 1
¥BX       . . 0 . .    . . . 0 .    . 0 . . 0    . . 0 . .
          . . 0 . .    . . 0 0 .    . 0 . . 0    0 0 0 0 0
          . . 0 . .    . 0 . 0 .    . 0 . . 0    . . 0 . .
          . . 0 . .    0 . . 0 .    . 0 . . 0    . . 0 . .
          0 0 0 0 0    . . . 0 .    0 . . 0 .    . . 0 . .        BIT 6
```

```
         X8          X9          XA          XB
     . . . . .   . 0 . . .   . . . . .   . 0 . 0 .   LSB
     . 0 0 0 0   . 0 0 0 0   0 0 0 0 0   0 0 0 0 0   BIT 1
¥BX  . 0 . . 0   . 0 . 0 .   . . . . 0   . 0 . 0 .
     0 . . . 0   0 . . 0 .   . . . . 0   . 0 . 0 .
     . . . . 0   . . . 0 .   . . . . 0   . . . 0 .
     . . . 0 .   . . . 0 .   . . . . 0   . . . 0 .
     . 0 0 . .   . . 0 . .   0 0 0 0 0   . . 0 . .   BIT 6

         XC          XD          XE          XF
     . . . . .   . . . . .   . 0 . . .   . . . . .   LSB
     0 0 . . .   0 0 0 0 0   0 0 0 0 0   0 . . . 0   BIT 1
¥BX  . . . . 0   . . . . 0   . 0 . . 0   0 . . . 0
     0 0 . . 0   . . . 0 .   . 0 . 0 .   . 0 . . 0
     . . . . 0   . . 0 . .   . 0 . . .   . . . . 0
     . . . 0 .   . 0 . 0 .   . 0 . . .   . . . 0 .
     0 0 0 . .   0 . . . 0   . . 0 0 0   . 0 0 . .   BIT 6

         X0          X1          X2          X3
     . . . . .   . . . 0 .   . . . . .   . 0 0 0 .   LSB
     . 0 0 0 0   0 0 0 . .   0 . 0 . 0   . . . . .   BIT 1
¥CX  . 0 . . 0   . . 0 . .   0 . 0 . 0   0 0 0 0 0
     0 . 0 . 0   0 0 0 0 0   0 . 0 . 0   . . 0 . .
     . . . 0 0   . . 0 . .   . . . . 0   . . 0 . .
     . . . 0 .   . . 0 . .   . . . 0 .   . . 0 . .
     . 0 0 . .   . 0 . . .   . . 0 . .   . 0 . . .   BIT 6

         X4          X5          X6          X7
     . 0 . . .   . . 0 . .   . . . . .   . . . . .   LSB
     . 0 . . .   . . 0 . .   . 0 0 0 .   0 0 0 0 0   BIT 1
¥CX  . 0 . . .   0 0 0 0 0   . . . . .   . . . . 0
     . 0 0 . .   . . 0 . .   . . . . .   . 0 . 0 .
     . 0 . 0 .   . . 0 . .   . . . . .   . . 0 . .
     . 0 . . .   . 0 . . .   . . . . .   . 0 . 0 .
     . 0 . . .   0 . . . .   0 0 0 0 0   0 . . . .   BIT 6

         X8          X9          XA          XB
     . . 0 . .   . . . 0 .   . . . . .   0 . . . .   LSB
     0 0 0 0 0   . . . 0 .   . . 0 . .   0 . . . .   BIT 1
¥CX  . . . 0 .   . . . 0 .   . . . 0 .   0 0 0 0 0
     . . 0 0 .   . . . 0 .   0 . . . 0   0 . . . .
     . 0 0 . 0   . . . 0 .   0 . . . 0   0 . . . .
     0 . 0 . .   . . 0 . .   0 . . . 0   0 . . . .
     . . 0 . .   . 0 . . .   0 . . . 0   . 0 0 0 0   BIT 6

         XC          XD          XE          XF
     . . . . .   . . . . .   . . 0 . .   . . . . .   LSB
     0 0 0 0 0   . 0 . . .   0 0 0 0 0   0 0 0 0 0   BIT 1
¥CX  . . . . 0   0 . 0 . .   . . 0 . .   . . . . 0
     . . . . 0   . . . 0 .   . . 0 . .   . . . . 0
     . . . . 0   . . . . 0   0 . 0 . 0   . 0 . 0 .
     . . . 0 .   . . . . 0   0 . 0 . 0   . . 0 . .
     . 0 0 . .   . . . . .   . . 0 . .   . . . 0 .   BIT 6
```

```
         X0          X1          X2          X3
     . . . . .   . . . . .   . . . . .   . . . . .   LSB
     0 0 0 0 .   . . 0 . .   . . . . 0   0 0 0 0 0   BIT 1
¥DX  . . . . .   . 0 . . .   . . . . 0   . 0 . . .
     0 0 0 0 .   0 . . . .   . 0 . 0 .   0 0 0 0 0
     . . . . .   0 . . . 0   . . 0 . .   . 0 . . .
     0 0 0 0 .   0 0 0 0 0   . 0 . 0 .   . 0 . . .
     . . . . 0   . . . . 0   0 . . . .   . . 0 0 0   BIT 6

         X4          X5          X6          X7
     . 0 . . .   . . . . .   . . . . .   . 0 0 0 .   LSB
     . 0 . . .   . 0 0 0 .   0 0 0 0 0   . . . . .   BIT 1
¥DX  0 0 0 0 0   . . . 0 .   . . . . 0   0 0 0 0 0
     . 0 . . 0   . . . 0 .   0 0 0 0 0   . . . . 0
     . 0 . 0 .   . . . 0 .   . . . . 0   . . . . 0
     . 0 . . .   . . . 0 .   . . . . 0   . . . 0 .
     . 0 . . .   0 0 0 0 0   0 0 0 0 0   . . 0 . .   BIT 6

         X8          X9          XA          XB
     . . . . .   . . . . .   . . . . .   . . . . .   LSB
     0 . . 0 .   . . 0 . .   0 . . . .   0 0 0 0 0   BIT 1
¥DX  0 . . 0 .   0 . 0 . .   0 . . . .   0 . . . 0
     0 . . 0 .   0 . 0 . 0   0 . . . 0   0 . . . 0
     . . . 0 .   0 . 0 . 0   0 . . 0 .   0 . . . 0
     . . 0 . .   0 . 0 . 0   0 . 0 . .   0 . . . 0
     . 0 . . .   0 . 0 0 .   0 0 . . .   0 0 0 0 0   BIT 6

         XC          XD          XE          XE
     . . . . .   . . . . .   . . 0 . .   0 0 0 . .   LSB
     0 0 0 0 0   0 0 . . .   0 . . 0 .   0 . 0 . .   BIT 1
¥DX  0 . . . 0   . . . . .   . 0 . . .   0 0 0 . .
     0 . . . 0   . . . . 0   . . . . .   . . . . .
     . . . . 0   . . . . 0   . . . . .   . . . . .
     . . . 0 .   . . . 0 .   . . . . .   . . . . .
     . . 0 . .   0 0 0 . .   . . . . .   . . . . .   BIT 6
```

```
            USA          FRENCH       GERMANY      ENGLAND

         . 0 . 0 .    . 0 . 0 .    . 0 . 0 .    . . 0 0 .    LSB
         . 0 . 0 .    . 0 . 0 .    . 0 . 0 .    . 0 . . 0    BIT 1
¥23      0 0 0 0 0    0 0 0 0 0    0 0 0 0 0    . 0 . . .
         . 0 . 0 .    . 0 . 0 .    . 0 . 0 .    0 0 0 . .
         0 0 0 0 0    0 0 0 0 0    0 0 0 0 0    . 0 . . .
         . 0 . 0 .    . 0 . 0 .    . 0 . 0 .    . 0 . . .
         . 0 . 0 .    . 0 . 0 .    . 0 . 0 .    0 0 0 0 0    BIT 6

         . . 0 . .    . . 0 . .    . . 0 . .    . . 0 . .
         . 0 0 0 0    . 0 0 0 0    . 0 0 0 0    . 0 0 0 0
¥24      0 . 0 . .    0 . 0 . .    0 . 0 . .    0 . 0 . .
         . 0 0 0 .    . 0 0 0 .    . 0 0 0 .    . 0 0 0 .
         . . 0 . 0    . . 0 . 0    . . 0 . 0    . . 0 . 0
         0 0 0 0 .    0 0 0 0 .    0 0 0 0 .    0 0 0 0 .
         . . 0 . .    . . 0 . .    . . 0 . .    . . 0 . .

         . 0 0 0 .    . 0 . . .    . 0 0 0 .    . 0 0 0 .
         0 . . . 0    . . 0 . .    0 . . . .    0 . . . 0
¥40      0 . 0 0 0    0 0 0 0 .    . 0 0 0 .    0 . 0 0 0
         0 . 0 . 0    . . . . 0    0 . . . 0    0 . 0 . 0
         0 . 0 0 0    . 0 0 0 0    . 0 0 0 .    0 . 0 0 0
         0 . . . .    0 . . . 0    . . . . 0    0 . . . .
         . 0 0 0 .    . 0 0 0 0    . 0 0 0 .    . 0 0 0 .

         . 0 0 0 .    . . 0 . .    . 0 . 0 .    . 0 0 0 .
         . 0 . . .    . 0 . 0 .    . . . . .    . 0 . . .
¥5B      . 0 . . .    . . 0 . .    . . 0 . .    . 0 . . .
         . 0 . . .    . . . . .    . 0 . 0 .    . 0 . . .
         . 0 . . .    . . . . .    0 . . . 0    . 0 . . .
         . 0 . . .    . . . . .    0 0 0 0 0    . 0 . . .
         . 0 0 0 .    . . . . .    0 . . . 0    . 0 0 0 .

         . . . . .    . . . . .    . 0 . 0 .    . . . . .
         0 . . . .    . 0 0 0 0    . . . . .    0 . . . .
¥5C      . 0 . . .    0 . . . .    . 0 0 0 .    . 0 . . .
         . . 0 . .    0 . . . .    0 . . . 0    . . 0 . .
         . . . 0 .    . 0 0 0 0    0 . . . 0    . . . 0 .
         . . . . 0    . . . 0 .    0 . . . 0    . . . . 0
         . . . . .    . 0 0 . .    . 0 0 0 .    . . . . .

         . 0 0 0 .    . 0 0 0 .    . 0 . 0 .    . 0 0 0 .
         . . . 0 .    0 . . . .    . . . . .    . . . 0 .
¥5D      . . . 0 .    . 0 0 0 .    0 . . . 0    . . . 0 .
         . . . 0 .    0 . . . 0    0 . . . 0    . . . 0 .
         . . . 0 .    . 0 0 0 .    0 . . . 0    . . . 0 .
         . . . 0 .    . . . . 0    0 . . . 0    . . . 0 .
         . 0 0 0 .    . 0 0 0 .    . 0 0 0 .    . 0 0 0 .
```

```
            USA          FRENCH       GERMAN       ENGLANC

         . . 0 . .    . . 0 . .    . . 0 . .    . . 0 . .
         . 0 . 0 .    . 0 . 0 .    . 0 . 0 .    . 0 . 0 .
¥5E      0 . . . 0    0 . . . 0    0 . . . 0    0 . . . 0
         . . . . .    . . . . .    . . . . .    . . . . .
         . . . . .    . . . . .    . . . . .    . . . . .
         . . . . .    . . . . .    . . . . .    . . . . .
         . . . . .    . . . . .    . . . . .    . . . . .

         . 0 0 . .    . 0 0 . .    . 0 0 . .    . 0 0 . .
         . 0 . . .    . 0 . . .    . 0 . . .    . 0 . . .
¥60      . . 0 . .    . . 0 . .    . . 0 . .    . . 0 . .
         . . . . .    . . . . .    . . . . .    . . . . .
         . . . . .    . . . . .    . . . . .    . . . . .
         . . . . .    . . . . .    . . . . .    . . . . .
         . . . . .    . . . . .    . . . . .    . . . . .

         . . 0 0 .    . . . 0 .    . 0 . 0 .    . . . . .
         . 0 . . .    . . 0 . .    . . . . .    . . . . .
¥7B      . 0 . . .    . 0 0 0 .    0 0 0 0 .    . . . . .
         0 . . . .    0 . . . 0    . . . . 0    . . . . .
         . 0 . . .    0 0 0 0 0    . 0 0 0 0    . . . . .
         . 0 . . .    0 . . . .    0 . . . 0    . . . . .
         . . 0 0 .    . 0 0 0 0    . 0 0 0 0    . . . . .

         . . 0 . .    . 0 . . .    . . . . .    . . . . .
         . . 0 . .    . . 0 . .    . 0 . 0 .    . . . . .
¥7C      . . 0 . .    . . . . .    . . . . .    . . . . .
         . . . . .    0 . . . 0    . 0 0 0 .    . . . . .
         . . 0 . .    0 . . . 0    0 . . . 0    . . . . .
         . . 0 . .    0 . . . 0    0 . . . 0    . . . . .
         . . 0 . .    . 0 0 0 0    . 0 0 0 .    . . . . .

         . 0 0 . .    . 0 . . .    . 0 . 0 .    . . . . .
         . . . 0 .    . . 0 . .    . . . . .    . . . . .
¥7D      . . . 0 .    . 0 0 0 .    . . . . .    . . . . .
         . . . . 0    0 . . . 0    0 . . . 0    . . . . .
         . . . 0 .    0 0 0 0 0    0 . . . 0    . . . . .
         . . . 0 .    0 . . . .    0 . . . 0    . . . . .
         . 0 0 . .    . 0 0 0 0    . 0 0 0 0    . . . . .

         . 0 . . .    . 0 . 0 .    . 0 0 . .    . . . . .
         0 . 0 . 0    . . . . .    0 . . 0 .    . . . . .
¥7E      . . . 0 .    . . . . .    0 . . 0 .    . . . . .
         . . . . .    . . . . .    0 . 0 . .    . . . . .
         . . . . .    . . . . .    0 . . 0 .    . . . . .
         . . . . .    . . . . .    0 . . 0 .    . . . . .
         . . . . .    . . . . .    0 . 0 . .    . . . . .
```

```
           DENMARK          SWEDEN           ITALY           SPAIN
        (US VERSION)     (US VERSION)
         . 0 . 0 .        . 0 . 0 .        . 0 . 0 .        0 0 0 . .
         . 0 . 0 .        . 0 . 0 .        . 0 . 0 .        0 . . 0 .
  ¥23    0 0 0 0 0        0 0 0 0 0        0 0 0 0 0        0 0 0 . .
         . 0 . 0 .        . 0 . 0 .        . 0 . 0 .        0 . . 0 .
         0 0 0 0 0        0 0 0 0 0        0 0 0 0 0        0 . 0 0 0
         . 0 . 0 .        . 0 . 0 .        . 0 . 0 .        0 . . 0 .
         . 0 . 0 .        . 0 . 0 .        . 0 . 0 .        0 . . 0 0

         . . 0 . .        . . . . .        . . 0 . .        . . 0 . .
         . 0 0 0 0        0 . . . 0        . 0 0 0 0        . 0 0 0 0
  ¥24    0 . 0 . .        . 0 0 0 .        0 . 0 . .        0 . 0 . .
         . 0 0 0 .        0 . . . 0        . 0 0 0 .        . 0 0 0 .
         . . 0 . 0        0 . . . 0        . . 0 . 0        . . 0 . 0
         0 0 0 0 .        . 0 0 0 .        0 0 0 0 .        0 0 0 0 .
         . . 0 . .        0 . . . 0        . . 0 . .        . . 0 . .

         . 0 0 0 .        . . . 0 .        . 0 0 0 .        . 0 0 0 .
         0 . . . 0        . . 0 . .        0 . . . 0        0 . . . 0
  ¥40    0 . 0 0 0        0 0 0 0 0        0 . 0 0 0        0 . 0 0 0
         0 . 0 . 0        0 . . . .        0 . 0 . 0        0 . 0 . 0
         0 . 0 0 0        0 0 0 0 .        0 . 0 0 0        0 . 0 0 0
         0 . . . .        0 . . . .        0 . . . .        0 . . . .
         . 0 0 0 .        0 0 0 0 0        . 0 0 0 .        . 0 0 0 .

         . 0 . 0 0        . 0 . 0 .        . . 0 . .        . . 0 . .
         0 . 0 . .        . . . . .        . 0 . 0 .        . . . . .
  ¥5B    0 . 0 . .        . . 0 . .        . . 0 . .        . . . . .
         0 0 0 0 .        . 0 . 0 .        . . . . .        . . 0 . .
         0 . 0 . .        0 . . . 0        . . . . .        . . 0 . .
         0 . 0 . .        0 0 0 0 0        . . . . .        . . 0 . .
         0 . 0 0 0        0 . . . 0        . . . . .        . . 0 . .

         . . . . 0        . 0 . 0 .        . . . . .        . 0 . . 0
         . 0 0 0 .        . . . . .        0 . . . .        0 . 0 0 .
  ¥5C    0 . . 0 0        . 0 0 0 .        . 0 . . .        . . . . .
         0 . 0 . 0        0 . . . 0        . . 0 . .        0 0 . . 0
         0 0 . . 0        0 . . . 0        . . . 0 .        0 . 0 . 0
         . 0 0 0 .        0 . . . 0        . . . . 0        0 . . 0 0
         0 . . . .        . 0 0 0 .        . . . . .        0 . . . 0

         . . 0 . .        . . 0 . .        . . . 0 .        . . 0 . .
         . 0 . 0 .        . 0 . 0 .        . . 0 . .        . . . . .
  ¥5D    . . 0 . .        . . 0 . .        . 0 0 0 .        . . 0 . .
         . 0 . 0 .        . 0 . 0 .        0 . . . 0        . 0 . . .
         0 . . . 0        0 . . . 0        0 0 0 0 0        0 . . . .
         0 0 0 0 0        0 0 0 0 0        0 . . . .        0 . . . 0
         0 . . . 0        0 . . . 0        . 0 0 0 0        . 0 0 0 .

         . . 0 . .        . 0 . 0 .        . . 0 . .        . . 0 . .
         . 0 . 0 .        . . . . .        . 0 . 0 .        . 0 . 0 .
  ¥5E    0 . . . 0        0 . . . 0        0 . . . 0        0 . . . 0
         . . . . .        0 . . . 0        . . . . .        . . . . .
         . . . . .        0 . . . 0        . . . . .        . . . . .
         . . . . .        0 . . . 0        . . . . .        . . . . .
         . . . . .        . 0 0 0 .        . . . . .        . . . . .
```

```
           DENMARK          SWEDEN           ITALY           SPAIN

         . 0 0 . .        . . . 0 .        . 0 . . .        . 0 0 . .
         . 0 . . .        . . 0 . .        . . 0 . .        . 0 . . .
  ¥60    . . 0 . .        . 0 0 0 .        . . . . .        . . 0 . .
         . . . . .        0 . . . 0        0 . . . 0        . . . . .
         . . . . .        0 0 0 0 0        0 . . . 0        . . . . .
         . . . . .        0 . . . .        0 . . . 0        . . . . .
         . . . . .        . 0 0 0 0        . 0 0 0 0        . . . . .

         . . . . .        . 0 . 0 .        . 0 . . .        . 0 . 0 .
         . . . . .        . . . . .        . . 0 . .        . . . . .
  ¥7B    0 0 . 0 .        0 0 0 0 .        0 0 0 0 .        . . . . .
         . . 0 . 0        . . . . 0        . . . . 0        . . . . .
         . 0 0 0 0        . 0 0 0 0        . 0 0 0 0        . . . . .
         0 . 0 . .        0 . . . 0        0 . . . 0        . . . . .
         . 0 0 0 0        . 0 0 0 0        . 0 0 0 0        . . . . .

         . . . . .        . 0 . 0 .        . 0 . . .        . 0 . . 0
         . . . 0 .        . . . . .        . . 0 . .        0 . 0 0 .
  ¥7C    . 0 0 0 .        . . . . .        . . . . .        . . . . .
         0 . 0 . 0        . 0 0 0 .        . 0 0 0 .        0 0 0 0 .
         0 . 0 . 0        0 . . . 0        0 . . . 0        0 . . . 0
         . 0 0 0 .        0 . . . 0        0 . . . 0        0 . . . 0
         . 0 . . .        . 0 0 0 .        . 0 0 0 .        0 . . . 0

         . . 0 . .        . . 0 . .        . 0 . . .        . 0 0 . .
         . . . . .        . 0 . 0 .        . . 0 . .        . . . 0 .
  ¥7D    0 0 0 0 .        0 0 0 0 .        . 0 0 0 .        . . . 0 .
         . . . . 0        . . . . 0        0 . . . 0        . . . . 0
         . 0 0 0 0        . 0 0 0 0        0 0 0 0 0        . . . 0 .
         0 . . . 0        0 . . . 0        0 . . . .        . . . 0 .
         . 0 0 0 0        . 0 0 0 0        . 0 0 0 0        . 0 0 . .

         . 0 . . .        . 0 . 0 .        . 0 . . .        . 0 . . .
         0 . 0 . 0        . . . . .        . . 0 . .        0 . 0 . 0
  ¥7E    . . . 0 .        . . . . .        . . . . .        . . . 0 .
         . . . . .        0 . . . 0        . . 0 . .        . . . . .
         . . . . .        0 . . . 0        . 0 0 . .        . . . . .
         . . . . .        0 . . . 0        . . 0 . .        . . . . .
         . . . . .        . 0 0 0 0        . 0 0 0 .        . . . . .
```

```
          DENMARK        SWEDEN         NORWAY
       (EU VERSION)   (EU VERSION)
        . 0 . 0 .      . 0 . 0 .      . 0 . 0 .
        . 0 . 0 .      . 0 . 0 .      . 0 . 0 .
¥23     0 0 0 0 0      0 0 0 0 0      0 0 0 0 0
        . 0 . 0 .      . 0 . 0 .      . 0 . 0 .
        0 0 0 0 0      0 0 0 0 0      0 0 0 0 0
        . 0 . 0 .      . 0 . 0 .      . 0 . 0 .
        . 0 . 0 .      . 0 . 0 .      . 0 . 0 .

        . . 0 . .      . . . . .      . . . . .
        . 0 0 0 0      0 . . . 0      0 . . . 0
¥24     0 ..0 . .      . 0 0 0 .      . 0 0 0 .
        . 0 0 0 .      0 . . . 0      0 . . . 0
        . . 0 . 0      0 . . . 0      0 . . . 0
        0 0 0 0 .      . 0 0 0 .      . 0 0 0 .
        . . 0 . .      0 . . . 0      0 . . . 0

        . . . 0 .      . . . 0 .      . . . 0 .
        . . 0 . .      . . 0 . .      . . 0 . .
¥40     0 0 0 0 0      0 0 0 0 0      0 0 0 0 0
        0 . . . .      0 . . . .      0 . . . .
        0 0 0 0 .      0 0 0 0 .      0 0 0 0 .
        0 . . . .      0 . . . .      0 . . . .
        0 0 0 0.0      0 0 0 0 0      0 0 0 0 0

        . 0 . 0 0      . 0 . 0 .      . 0 . 0 0
        0 . 0 . .      . . . . .      0 . 0 . .
¥5B     0 . 0 . .      . . 0 . .      0 . 0 . .
        0 0 0 0 .      . 0 . 0 .      0 0 0 0 .
        0 . 0 . .      0 . . . 0      0 . 0 . .
        0 . 0 . .      0 0 0 0 0      0 . 0 . .
        0 . 0 0 0      0 . . . 0      0 . 0 0 0

        . . . . 0      . 0 . 0 .      . . . . 0
        . 0 0 0 .      . . . . .      . 0 0 0 .
¥5C     0 . . 0 0      . 0 0 0 .      0 . . 0 0
        0 . 0 . 0      0 . . . 0      0 . 0 . 0
        0 0 . . 0      0 . . . 0      0 0 . . 0
        . 0 0 0 .      0 . . . 0      . 0 0 0 .
        0 . . . .      . 0 0 0 .      0 . . . .

        . . 0 . .      . . 0 . .      . . 0 . .
        . 0 . 0 .      . 0 . 0 .      . 0 . 0 .
¥5D     . . 0 . .      . . 0 . .      . . 0 . .
        . 0 . 0 .      . 0 . 0 .      . 0 . 0 .
        0 . . . 0      0 . . . 0      0 . . . 0
        0 0 0 0 0      0 0 0 0 0      0 0 0 0 0
        0 . . . 0      0 . . . 0      0 . . . 0

        . 0 . 0 .      . 0 . 0 .      . 0 . 0 .
        . . . . .      . . . . .      . . . . .
¥5E     0 . . . 0      0 . . . 0      0 . . . 0
        0 . . . 0      0 . . . 0      0 . . . 0
        0 . . . 0      0 . . . 0      0 . . . 0
        0 . . . 0      0 . . . 0      0 . . . 0
        . 0 0 0 .      . 0 0 0 .      . 0 0 0 .
```

```
          DENMARK        SWEDEN          ITALY

        . . . 0 .      . . . 0 .      . . . 0 .
        . . 0 . .      . . 0 . .      . . 0 . .
¥60     . 0 0 0 .      . 0 0 0 .      . 0 0 0 .
        0 . . . 0      0 . . . 0      0 . . . 0
        0 0 0 0 0      0 0 0 0 0      0 0 0 0 0
        0 . . . .      0 . . . .      0 . . . .
        . 0 0 0 0      . 0 0 0 0      . 0 0 0 0

        . . . . .      . 0 . 0 .      . . . . .
        . . . . .      . . . . .      . . . . .
¥7B     0 0 . 0 .      0 0 0 0 .      0 0 . 0 .
        . . 0 . 0      . . . . 0      . . 0 . 0
        . 0 0 0 0      . 0 0 0 0      . 0 0 0 0
        0 . 0 . .      0 . . . 0      0 . 0 . .
        . 0 0 0 0      . 0 0 0 0      . 0 0 0 0

        . . . . .      . 0 . 0 .      . . . . .
        . . . 0 .      . . . . .      . . . 0 .
¥7C     . 0 0 0 .      . . . . .      . 0 0 0 .
        0 . 0 . 0      . 0 0 0 .      0 . 0 . 0
        0 . 0 . 0      0 . . . 0      0 . 0 . 0
        . 0 0 0 .      0 . . . 0      . 0 0 0 .
        . 0 . . .      . 0 0 0 .      . 0 . . .

        . . 0 . .      . . 0 . .      . . 0 . .
        . . . . .      . . . . .      . . . . .
¥7D     0 0 0 0 .      0 0 0 0 .      0 0 0 0 .
        . . . . 0      . . . . 0      . . . . 0
        . 0 0 0 0      . 0 0 0 0      . 0 0 0 0
        0 . . . 0      0 . . . 0      0 . . . 0
        . 0 0 0 0      . 0 0 0 0      . 0 0 0 0

        . 0 . 0 .      . 0 . 0 .      . 0 . 0 .
        . . . . .      . . . . .      . . . . .
¥7E     . . . . .      . . . . .      . . . . .
        0 . . . 0      0 . . . 0      0 . . . 0
        0 . . . 0      0 . . . 0      0 . . . 0
        0 . . . 0      0 . . . 0      0 . . . 0
        . 0 0 0 0      . 0 0 0 0      . 0 0 0 0

            X0             X1             X2                X3
        . . . . .      . . . . .      . . . . .        . . . . .        LSB
        . . . . .      . . . . .      . . . . .        . . . . .        BIT 1
¥8X     . . . . .      . . . . .      . . . . .        . . . . .
        . . . . .      . . . . .      . . . . .        . . . . .
        . . . . .      . . . . .      . . . . .        . . . . .
        . . . . .      . . . . .      . . . . .        . . . . .
        . . . . .      . . . . .      . . . . .        . . . . .        BIT 6
```

```
     WRITTEN BY        KENJI AKAHANE
     REVISION          B-1
     FILE              ROMCAS HXID
     DATE              11.28.1981
     UPDATE            11.18.1982

   NOTE.  A LINE WITH 72 COLUMN '%' MARK IS UPDATED.


     1. HEADER
      32 BYTES SIZE

      0 - 7   COLUMN: FILE NAME
                      (COLUMN 0 VALUE   ¥00:DELETED  ¥FF:END OF HEADER)
      8 - 15  COLUMN: FILE TYPE
      16 - 19 COLUMN: TOP ADDRESS (4 BYTES HEXADECIMAL ASCII CODE)
      20 - 23 COLUMN: BOTTOM ADDRESS + 1 (4 BYTES HEXADECIMAL ASCII CODE)
      24 - 29 COLUMN: DATE (MMDDYY ASCII CODE)
      30 - 31 COLUMN: NOT USED

     2. ROM MEMORY LOCATION
      ¥00 - ¥1F : HEADER 0 (HEADER FOR FILE 0)
      ¥20 - ¥3F : HEADER 1
      ¥40 - ¥5F : HEADER 2
      ¥60 - ¥7F : HEADER 3
          :
          :
          :
      ¥E0 - ¥FF : HEADER 7
      ¥100 - ¥11F : HEADER 8
          :
          :
          :
      ¥1C0 - ¥1DF : HEADER 14
      ¥1E0 - ¥1FF : HEADER 15
      ¥200        : END MARK (¥FF)
      ¥201 -      : DATA STRING

    IF ONLY ONE FILE IS REQURED, WE CAN USE MEMORY FROM ¥21. (THE CONTENT
   OF ¥20 MUST BE ¥FF.)

    CALLING SEQUENCE
    SEE 'MAINIO HX1D'


    MAIN MEMORY MAP                                                     @
                                                                        @
     208       :ROM CARTRIDGE STATUS                                    @
     209 - 20A:ADDRESSING COUNTER                                       @
     20B/- 20C:ADDRESS OF THE TOP OF FILE                               @
     20D - 20E:ADDRESS OF THE BOTTOM + 1 OF FILE                        @
```

```
DOCUMENT OF SCREEN COMMAND
FILE NAME     SCREEN HX1D
CREATED       06.10.1982
AUTHOR        K. AKAHANE,  M. HANAOKA
UPDATE        06.23.1982   K. A


 SCREEN COMMAND

 ENTRY POINT : DISCON (¥FF5E)
 PRAMETER    : 1 ACCX : PACKET TOP ADDRESS.
                2 PACKET
                  0 : FUNCTION
                  1 : DATA 1
                  .     .
                  .     .
                  N : DATA N

                3 NOTE
                  IF YOU USE CRT THEN PACKET NEED MORE 4 BYTES AREA
                  FROM FUNCTION-4 TO FUNCTION-1. BUT THEN ACCX POINTS
                  FUNCTION PACKET ADDRESS.

 RETURN       : 1 PACKET
                  DATA IS SET FROM NEXT ADDRESS OF FUNCTION.

 REGISTER     : PRESERVD ACCX AND ACCB
 FMT  DID  SID  FNC  SIZ      MSG / FUNCTION NAME (L:LCD C:CRT)
 ----------------------------------------------------------------------------
                              SCREEN DEVICE SELECT(L.C)
 00   MM   SS   84   00       00-00 = DEVICE NO. (CRT:30 LCD:22)
 01   SS   MM   84   00       00-00 = ERROR CODE.
                                      00 : NON ERROR.
                                      FE : DEVICE NOT READY.
                                      FF : DEVICE NAME IS NOT CORRECT.

                              INITIALIZE SCREEN DEVICE.(C)
 00   MM   SS   85   00       00-00 = XX
 01   SS   MM   85   00       00-00 = ERROR CODE.
                                      00 : NON ERROR.
                                      FE : I/O ERROR.

                              CHECK SCREEN DEVICE
                              AND GET SOME PARAMETERS. (L.C)
 00   MM   SS   86   00       00-00 = XX
 01   SS   MM   86   00       00-00 = SCREEN DEVICE NO.
                              01-02 = SCREEN TOP ADDRESS.(CRT)
                              03-04 = MAX SCREEN SIZE.(CRT)

                              SET SCREEN SIZE (L.C)
 00   MM   SS   87   03       00-00 = SCREEN WIDTH OF VIRTUAL SCREEN
                              01-01 = SCREEN DEPTH OF VIRTUAL SCREEN
                              02-03 = TOP ADDRESS OF VIRTUAL SCREEN
                                      (NOT USED)
 01   SS   MM   87   00       00-00 = ERROR CODE.
                                      00 : NON ERROR.
                                      FF : SCREEN SIZE IS NOT CORRECT.
                                      FE : ADDRESS OF TOP OF SCREEN IS
                                           NOT CORRECT.
                              READ SCREEN SIZE (L.C)
 00   MM   SS   88   00       00-00 = XX
 01   SS   MM   88   01       00-00 = SCREEN WIDTH
                              01-01 = SCREEN DEPTH

                              GET PHYSICAL SCREEN SIZE (L.C)
 00   MM   SS   89   80       00-00 = XX
 01   SS   MM   89   01       00-00 = SCREEN WIDTH
                              01-01 = SCREEN DEPTH

                              SET THE  PYSICAL SCREEN POINTER
                               ON THE VIRTUAL SCREEN (L.C)
 00   MM   SS   C0   01       00-00 = COORDINATE (X)
                              01-01 = COORDINATE (Y)
 01   SS   MM   C0   00       00-00 = XX

                              GET THE PYSICAL SCREEN POINTER
                               ON THE VIRTUAL SCREEN (L.C)
 00   MM   SS   8A ✓ 00       00-00 = XX
 01   SS   MM   8A   01       00-00 = COORDINATE (X)
                              01-01 = COORDINATE (Y)
```

```
                              ?SET READ POINTER (L.C)
                               (ERASED 1982/05/08)
 00   MM   SS   C1   01       00-00 = COORDINATE OF X IN THE VIRTUAL
                                      SCREEN
                              01-01 = COORDINATE OF Y
 01   SS   MM   C1   00       00-00 = XX

                              ?GET READ POINTER (L.C)
                               (ERASED 1982/05/08)
 00   MM   SS   8B   00       00-00 = XX
 01   SS   MM   8B   01       00-00 = COORDINATE OF X IN THE VIRTUAL
                                      SCREEN
                              01-01 = COORDINATE OF Y

                              SET CURSOR POSITION
                               ON THE VIRTUAL SCREEN (L.C)
 00   MM   SS   C2   01       00-00 = COORDINATE (X)
                              01-01 = COORDINATE (Y)
 01   SS   MM   C2   00       00-00 = XX

                              GET CURSOR POSITION
                               ON THE VIRTUAL SCREEN (L.C)
 00   MM   SS   8C   00       00-00 = XX
 01   SS   MM   8C   01       00-00 = COORDINATE (X)
                              01-01 = COORDINATE (Y)

                              SET CURSOR MARGIN (L.C)
 00   MM   SS   C3   00       00-00 = MARGIN
 01   SS   MM   C3   00       00-00 = XX

                              GET CURSOR MARGIN (L.C)
 00   MM   SS   8D   00       00-00 = XX
 01   SS   MM   8D   00       00-00 = MARGIN

                              SET SCROLL STEP (L.C)
 00   MM   SS   C4   01       00-00 = HORIZONTAL SCROLL STEP
                              01-01 = VERTICAL SCROLL STEP
 01   SS   MM   C4   00       00-00 = XX

                              GET SCROLL STEP (L.C)
 00   MM   SS   8E   00       00-00 = XX
 01   SS   MM   8E   01       00-00 = HORIZONTAL SCROLL STEP
                              01-01 = VERTICAL SCROLL STEP

                              SET LIST FLAG (L.C)
 00   MM   SS   C5   00       00-00 = XX
 01   SS   MM   C5   00       00-00 = XX

                              RESET LIST FLAG (L.C)
 00   MM   SS   C6   00       00-00 = XX
 01   SS   MM   C6   00       00-00 = XX
```

```
                                   SET POINT TO DISPLAY (L.C)
00   MM   SS   C7   04             00-01 = COORDINATE (X)
                                   02-03 = COORDINATE (Y)
                                   04-04 = COLOR CODE
01   SS   MM   C7   00             00-00 = XX

                                   GET POINT ON THE DISPLAY (L.C)
00   MM   SS   8F   03             00-01 = COORDINATE (X)
                                   02-03 = COORDINATE (Y)
01   SS   MM   8F   00             00-00 = COLOR CODE

                                   DRAW LINE TO DISPLAY (L.C)
00   MM   SS   C8   08             00-01 = COORDINATE (X) OF START POINT
                                   02-03 = COORDINATE (Y) OF START POINT
                                   04-05 = COORDINATE (Y) OF END POINT
                                   06-07 = COORDINATE (Y) OF END POINT
                                   08-08 = COLOR CODE (C)
01   SS   MM   C8   00             00-00 = XX

                                   ?READ CHARACTER ON THE READ POINTER (L.C)
                                   (ERASED 1982/05/08)
00   MM   SS   90   00             00-00 = XX
01   SS   MM   90   00             00-00 = READ CHARACTER

                                   READ THE EXTENT OF CURRENT LINE. (L.C)
00   MM   SS   91   00             00-00 = XX
01   SS   MM   91   03             00-00 = FIRST LINE NUMBER WHICH HAS
                                           CURRENT LINE.
                                   01-01 = LAST LINE NUMBER WHICH HAS
                                           CURRRENT LINE.
                                   02-02 = LAST COORDINATE (X)
                                   03-03 = LAST COORDINATE (Y)

                                   SET LINE TERMINATE POSITION (L.C)
00   MM   SS   C9   00             00-00 = LINE NUMBER
01   SS   MM   C9   00             00-00 = XX

                                   WRITE ONE CHARACTER
                                    TO VIRTUAL SCREEN (L.C)
00   MM   SS   92   00             00-00 = CHARACTER CODE
01   SS   MM   92   01             00-00 = CURSOR POSITION X
                                   01-01 = CURSOR POSITION Y

                                   CLEAR THE GRAPHIC SCREEN. (L.C)
00   MM   SS   CA   00             00-00 = BACK GROUND COLOR (C)
01   SS   MM   CA   00             00-00 = XX

                                   SET SCROLL SPEED (C)
00   MM   SS   CB   00             00-00 = SCROLL SPEED (0-9)
01   SS   MM   CB   00             00-00 = XX
```

```
                                                                      %
                                  **SET DISPLAY MODE (C)              %
00   MM   SS   93   02              00-00 = CHARACTER MODE.           %
                                            (00:NOT USE 01:ACTIVE)    %
                                    01-01 = GRAPHIC MODE.             %
                                            (00:NOT USE 01:GRP0 02:GRP1)  %
                                    02-02 = BACKGROUND COLOR          %
                                            (00:GREEN                 %
                                             01:YELLOW                %
                                             02:BLUE                  %
                                             03:RED                   %
                                             04:WHITE                 %
                                             05:SYIAN                 %
                                             06:MAGENDA               %
                                             07:ORANGE  )             %
01   SS   MM   93   00              00-00 = ERROR CODE                %
                                            (00:NON ERROR FF:ERROR)   %

                                   ?SET CURSOR POSITION
                                    ON PHISICAL SCREEN (C)
00   MM   SS   CC   01              00-00 = COORDINATE (X)
                                    01-01 = COORDINATE (Y)
01   SS   MM   CC   00              00-00 = XX

                                   ?GET CURSOR POSITION
                                    ON PHISICAL SCREEN (C)
00   MM   SS   94   00              00-00 = XX
01   SS   MM   94   01              00-00 = COORDINATE (X)
                                    01-01 = COORDINATE (Y)

                                    WRITE ONE CHARACTER ON ACC-POSITION
                                    ON PHYSICAL SCREEN.
00   MM   SS   CD   01              00-00 = CHARACTER CODE
                                    01-01 = COLOR CODE.
01   SS   MM   CD   00              00-00 = XX

                                    SET ACC-POSITION ON PHYSICAL SCREEN(C)
00   MM   SS   CE   01              00-00 = COORDINATE (X)
                                    01-01 = COORDINATE (Y)
01   SS   MM   CE   00              00-00 = XX

                                    READ ONE CHARACTER ON ACC-POSITION
                                    ON PHYSICAL SCREEN (C)
00   MM   SS   95   00              00-00 = XX
01   SS   MM   95   01              00-00 = CHRACTER CODE.
                                    01-01 = COLOR CODE.

                                    COLOR SET SELECT (C)                  %
00   MM   SS   CF   00              00-00 = COLOR SET CODE.               %
                                            (0:COLOR SET 0  1:COLOR SET 1) %
01   SS   MM   CF   00              00-00 = XX                            %

                                    SET CURSOR MODE (C)
00   MM   SS   00   00              00-00 = CURSOR MODE
                                            (0:ON 1:OFF)
01   SS   MM   00   00              00-00 = XX

                                   ?READ ONE LINE'S (32 BYTES) CHARACTERS ON
                                    READ POINTER (C)
00   MM   SS   96   00              00-00 = XX
01   SS   MM   96   20              01-20 = CHARACTER CODE
```

```
                                         ?SET GRAPHIC CHARACTER FONT (C)
  00    MM    SS    D1    06              00-00 = CHARACTER CODE
                                          01-06 = CHARACTER FONT PATTERN
  01    SS    MM    D1    01              00-00 = XX

                                         ?DRAW CIRCLE (C)
  00    MM    SS    D2    0?              00-0? =
  01    SS    MM    D2    00              00-00 = XX

                                         ?PAINT (C)
  00    MM    SS    D3    0?              00-0? =
  01    SS    MM    D3    00              00-00 = XX

*** SCREEN NEW COMMAND (1982/05/08)

                                          READ CHARACTERS FROM VS. (L,C)
  00    MM    SS    97    03              00-00 = START X-COORDINATE.
                                          01-01 = START Y-COORDINATE.
                                          02-03 = READ CHARACTERS NO.
  01    SS    MM    97    NN              00-NN = CHARACTERS WHICH ARE RED.

                                          WRITE ONE CHARACTER TO VS AND
                                          GET EXTENT OF NEW CURSOR POSITION.(L,C)
  00    MM    SS    98    00              00-00 = CHARACTER CODE.
  01    SS    MM    98    03              00-00 = CURSOR POSITION X.
                                          01-01 = CURSOR POSITION Y.
                                          02-02 = FIRST LINE NO.
                                          03-03 = LAST LINE NO.
```

```
WORK AREA MEMORY MAP
 MEMORY WITH '*' MARK ARE USED IN 'MAINI03' (LCD DRIVER ROUTINE)

  50 -  51:SAVE SCREEN FUNCTION PACKET ADDRESS
  52 -  53:TEMPORARY
  54      :TEMPORARY
  55      :TEMPORARY
  56      :TEMPORARY
  57      :TEMPORARY
  58 -  59:TEMPORARY
  5A -  5B:TEMPORARY
  5C      :TEMPORARY
  5D      :TEMPORARY
  5E -  5F:TEMPORARY

  60 -  61:NOT USED (USED IN THE MONITOR)

*220 - 26F:REAL SCREEN BUFFER (80 BYTES)
 270 - 271:VIRTUAL SCREEN BUFFER TOP ADDRESS
 272 - 273:VIRTUAL SCREEN BUFFER BOTTOM ADDRESS
 274 - 275:PHISICAL SCREEN HOME POSITION ADDRESS IN THE BUFFER
 276      :VIRTUAL SCREEN WIDTH (X) (SIZE - 1)
 277      :VIRTUAL SCREEN DEPTH (Y) (SIZE - 1)
*278      :CURSOR POSITION IN THE PHISICAL SCREEN (X) (0 - 19)
*279      :CURSOR POSITION IN THE PHISICAL SCREEN (Y) (0 - 3)
 27A      :LEFT/RIGHT SCROLL STEP COUNT
 27B      :UP/DOWN SCROLL STEP COUNT
 27C      :CURSOR MARGIN
*27D      :SCROLL SPEED (0 - 9)
 27E      :POINTER WHERE CHARACTER IS DISPLAYED (X) (0 - 19)
 27F      :POINTER WHERE CHARACTER IS DISPLAYED (Y) (0 - 3)
*280      :DISPLAY STATUS
         BIT 7
          REFRESH SCREEN
          1:REWRITE ALL REAL SCREEN
          0:ONLY WRITE ONE CHARACTER
         BIT 6
          CURSOR FLAG (MODE)
          1:CURSOR ON
          0:CURSOR OFF
         BIT 5
          CURSOR SWITCH TO WRITE ONE CHARACTER
          1:CURSOR (UNDER LINE) ON
          0:CURSOR OFF
         BIT 4
          SCROLL DELAY FLAG
          1:WAIT(SCROLL DELAY) BEFOR WRITING ONE CHACTER
          0:NOT DELAY
         BIT 3
          NOT USED
         BIT 2
          NOT USED
         BIT 1
          NOT USED
         BIT 0
          LIST FLAG (FIX WINDOW)
          1:FIXED
          0:NOT FIXED
 281      :NOT USED  ('SAVEXX' IN NIPPON FIRST VERSION)
 282      :NOT USED  ('SAVEXY' IN FIRST VERSION)
 283      :NOT USED  ('SAVECH' IN FIRST VERSION)
 284 - 285:NOT USED  ('DATPNT' IN FIRST VERSION)
*286 - 28B:CHARACTER FONT BUFFER (6 BYTES)
```

```
HC-20 PROTOCOL
CREATED  BY  KENJI AKAHANE
REVISION     B-2
DATE         03.04.1982
UPDATE       04.20.1982
UPDATE       11.19.1982
FILE         PROTOCOL HX1D

NOTE. UPDATE MARK IS 72 COLUMN '%'. REVISION B
      UPDATE MARK IS 72 COLUMN '@'. REVISION C

 CP/NET LOGICAL MESSAGE SPECIFICATION

NOTES:  MM = MASTER ID
        SS = SLAVE ID
        XX = DON'T CARE BYTE
        NN = VALUE SPECIFIED

        ALL NUMERIC VALUES ARE IN HEXADECIMAL.

 FMT  DID  SID  FNC  SIZ      MSG / FUNCTION NAME
-------------------------------------------------------------------
                              SYSTEM RESET:
 00   MM   SS   00   00       00-00 = XX
 01   SS   MM   00   00       00-00 = 00

                              CONSOLE INPUT:
 00   MM   SS   01   00       00-00 = XX
 01   SS   MM   01   00       00-00 = 00

                              CONSOLE OUTPUT:
 00   MM   SS   02   00       00-00 = XX
 01   SS   MM   02   00       00-00 = CHARACTER INPUT

                              RAW CONSOLE INPUT:
 00   MM   SS   03   00       00-00 = MASTER CONSOLE #
 01   SS   MM   03   00       00-00 = CHARACTER INPUT

                              RAW CONSOLE OUTPUT:
 00   MM   SS   04   00       00-00 = MASTER CONSOLE #
                              00-00 = CHARACTER TO OUTPUT
 01   SS   MM   04   00       00-00 = 00

                              LIST PUTPUT:
 00   MM   SS   05   00       00-00 = MASTER LIST #
                              01-NN = CHARACTERS TO LIST DEVICE
                                      (NN = 01 TO 80)
 01   SS   MM   05   00       00-00 = 00

                              DIRECT CONSOLE I/O:
 00   MM   SS   06   00       00-00 = XX
 01   SS   MM   06   00       00-00 = 00
```

```
                              GET I/O BYTE:
 00   MM   SS   07   00       00-00 = XX
 01   SS   MM   07   00       00-00 = 00

                              SET I/O BYTE:
 00   MM   SS   08   00       00-00 = XX
 01   SS   MM   08   00       00-00 = 00

                              PRINT STRING:
 00   MM   SS   09   00       00-00 = XX
 01   SS   MM   09   00       00-00 = 00

                              READ CONSOLE BUFFER:
 00   MM   SS   0A   00       00-00 = XX
 01   SS   MM   0A   00       00-00 = 00

                              GET CONSOLE STATUS:
 00   MM   SS   0B   00       00-00 = MASTER CONSOLE #
 01   SS   MM   0B   00       00-00 = CONSOLE STATUS BYTE

                              RETURN VERSION NUMBER:
 00   MM   SS   0C   00       00-00 = XX
 01   SS   MM   0C   00       00-00 = 00

                              RESET DISK SYSTEM:
 00   MM   SS   0D   00       00-00 = XX
 01   SS   MM   0D   00       00-00 = 00

                              SELECT DISK:
 00   MM   SS   0E   00       00-00 = SELECTED DISK
 01   SS   MM   0E   00       00-00 = RETURN CODE

                              OPEN FILE:
 00   MM   SS   0F   0E       00-01 = FCB ADDRESS IN SLAVE
                              02-02 = DRIVE CODE
                              03-0A = FILE NAME
                              0B-0D = FILE TYPE
                              0E-0E = EXTENT NUMBER
 01   SS   MM   0F   00       00-00 = DIRECTORY CODE

                              CLOSE FILE:
 00   MM   SS   10   01       00-00 = FCB ADDRESS IN SLAVE
 01   SS   MM   10   00       00-00 = DIRECTORY CODE

                              SEARCH FOR FIRST:
 00   MM   SS   11   0C       00-00 = DRIVE CODE
                              01-08 = FILE NAME
                              09-0B = FILE TYPE
                              0C-0C = EXTENT NUMBER
 01   SS   MM   11   20       00-00 = DIRECTORY CODE
                              01-20 = DIRECTORY FCB ENTRY

                              SEARCH FOR NEXT:
 00   MM   SS   12   01       00-00 = XX
 01   SS   MM   12   00       00-00 = DIRECTORY CODE
                              01-20 = DIRECTORY FCB ENTRY
```

					DELETE FILE:
00	MM	SS	13	0C	00-00 = DRIVE CODE
					01-08 = FILE NAME
					09-0B = FILE TYPE
					0C-0C = EXTENT NUMBER
01	SS	MM	13	00	00-00 = DIRECTORY CODE
					READ SEQUENTIAL:
00	MM	SS	14	03	00-01 = FCB ADDRESS IN SLAVE
					02-02 = EXTENT NUMBER
					03-03 = CURRENT RECORD
01	SS	MM	14	82	00-00 = EXTENT NUMBER
					01-01 = CURRENT RECORD
					02-81 = SECTOR OF DATA READ
					82-82 = RETURN CODE
					WRITE SEQUENTIAL:
00	MM	SS	15	83	00-01 = FCB ADDRESS IN SLAVE
					02-02 = EXTENT NUMBER
					03-03 = CURRENT RECORD
					04-83 = SECTOR OF DATA TO WRITE
01	SS	MM	15	02	00-00 = EXTENT NUMBER
					01-01 = CURRENT RECORD
					02-02 = RETURN CODE
					MAKE FILE:
00	MM	SS	16	0E	00-01 = FCB ADDRESS IN SLAVE
					02-02 = DRIVE CODE
					03-0A = FILE NAME
					0B-0D = FILE TYPE
					0E-0E = EXTENT NUMBER
01	SS	MM	16	00	00-00 = DIRECTORY CODE
					RENAME FILE:
00	MM	SS	17	1F	00-00 = DRIVE CODE
					01-08 = FILE NAME
					09-0B = FILE TYPE
					0C-0C = EXTENT NUMBER
					0D-0D = S1 (NOT USED)
					0E-0E = S2 (NOT USED)
					0F-0F = RECORD COUNT (NOT USED)
					10-10 = DRIVE CODE
					11-~~18~~ = FILE NAME
					19-1B = FILE TYPE
					1C-1C = EXTENT NUMBER
					1D-1D = S1 (NOT USED)
					1E-1E = S2 (NOT USED)
					1F-1F = RECORD COUNT (NOT USED)
01	SS	MM	17	00	00-00 = DIRECTORY CODE

					RETURN LOGIN VECTOR:
00	MM	SS	18	00	00-00 = XX
01	SS	MM	18	00	00-00 = 00
					RETRUN CURRENT DISK:
00	MM	SS	19	0C	00-00 = XX
01	SS	MM	19	00	00-00 = 00
					SET DMA ADDRESS:
00	MM	SS	1A	00	00-00 = XX
01	SS	MM	1A	00	00-00 = 00
					GET ALLOCATION VECTOR ADDRESS
00	MM	SS	1B	00	00-01 = XX
01	SS	MM	1B	00	00-00 = 00
					WRITE PROTECT DISK:
00	MM	SS	1C	00	00-01 = XX
01	SS	MM	1C	00	00-00 = 00
					GET R/O VECTOR:
00	MM	SS	1D	00	00-00 = XX
01	SS	MM	1D	00	00-00 = 00
					SET FILE ATTRIBUTES:
00	MM	SS	1E	0C	00-00 = DRIVE CODES
					01-08 = FILE NAME
					09-0B = FILE TYPE
					0C-0C = EXTENT NUMBER
01	SS	MM	1E	00	00-00 = DIRECTORY CODE
					GET DISK PARAMETER ADDRESS
00	MM	SS	1F	00	00-01 = XX
01	SS	MM	1F	00	00-00 = 00
					SET/GET USER CODE:
00	MM	SS	20	00	00-00 = SET/GET CODE
01	SS	MM	20	00	00-00 = CURRENT CODE (IF GET)
					READ RANDOM
00	MM	SS	21	04	00-01 = FCB ADDRESS IN SLAVE
					02-04 = R0,R1,R2 RANDOM RECORD #
01	SS	MM	21	82	00-00 = EXTENT NUMBER
					01-01 = CURRENT RECORD
					02-81 = SECTOR OF DATA READ
					82-82 = RETURN CODE
					WRITE RONDOM:
00	MM	SS	22	84	00-01 = FCB ADDRESS IN SLAVE
					02-81 = SECTOR OF DATA TO WRITE
					82-84 = R0,R1,R2 RANDOM RECORD #
01	SS	MM	22	02	00-00 = EXTENT NUMBER
					01-01 = CURRENT RECORD
					02-02 = RETURN CODE

```
                                          COMPUTE FILE SIZE:
00    MM    SS    23    01                00-01 = FCB ADDRESS IN SLAVE
01    SS    MM    23    05                00-00 = EXTENT NUMBER
                                          01-01 = CURRENT RECORDE
                                          02-04 = R0,R1,R2 RANDOM RECORD #
                                          05-05 = RETURN CODE

                                          SET RANDOM RECORD
00    MM    SS    24    03                00-01 = FCB ADDRESS IN SLAVE
                                          02-02 = EXTENT NUMBER
                                          03-03 = CURRENT RECORD
01    SS    MM    24    03                00-02 = R0,R1,R2 RANDOM RECORD #
                                          03-03 = 00

                                          RESET DRIVE:
00    MM    SS    25    01                00-01 = DRIVE VECTOR
01    SS    MM    25    00                00-00 = RETURN CODE

                                          ACCESS DRIVE:
00    MM    SS    26    01                00-01 = DRIVE VECTOR
01    SS    MM    26    00                00-00 = 00

                                          FREE DRIVE:
00    MM    SS    27    01                00-01 = DRIVE VECTOR
01    SS    MM    27    00                00-00 = 00

                                          WRITE RANDOM WITH ZERO FILL:
00    MM    SS    28    84                00-01 = FCB ADDRESS IN SLAVE
                                          02-81 = SECTOR OF DATA TO WRITE
                                          82-84 = R0,R1,R2 RANDOM RECORD #
01    SS    MM    28    02                00-00 = EXTENT NUMBER
                                          01-01 = CURRENT RECORD
                                          02-02 = RETURN CODE
```

HC-20 SERIAL NETWORK PROTOCOL

1. MASTER - SLAVE HANDSHAKE

```
  SOURCE          DESTINATION          COMMENT

                                  ENQURE TO DESTINATION DEVICE
 (EOT)     --->
  P1       --->
  DID      --->
  SID      --->
  ENQ      --->
          <---     ACK

  SOH      --->                   SEND HEADER (FUNCTION)
  FMT      --->
  DID      --->
  SID      --->
  FNC      --->
  SIZ      --->
  HCS      --->
          <---     ACK (NAK), (WAK)

  STX      --->
  DB0      --->
  DB1      --->
   .
   .
  DBN      --->
  ETX      --->
  CKS      --->
          <---     ACK, (NAK)

 (EOT)     --->
```

2. NET WORK RS232C 8-BIT STANDARD PROTOCOL

```
| FMT | DID | SID | FNC | SIZ | MSG                   |
```

FMT = MESSAGE FORMAT CODE
DID = MESSAGE DESTINATION PROCESSOR ID
SID = MESSAGE SOURCE PROCESSOR ID
FNC = FUNCTION CODE
SIZ = DATA FIELD LENGTH - 1
MSG = ACTUAL MESSAGE, SIZ + 1 BYTES LONG

MESSAGE FIELD LENGTH TABLE

FMT CODE	FMT	DID	SID	FNC	SIZ	MSG	COMMENT
00	1	1	1	1	1	1-256	PREFERRED FORMAT
01	1	1	1	1	1	1-256	RETURNED RESULT
02	1	1	1	1	2	1-65536	(NOT SUPPORTED)
03	1	1	1	1	2	1-65536	(NOT SUPPORTED)
04	2	2	2	1	1	1-256	(NOT SUPPORTED)
05	2	2	2	1	1	1-256	(NOT SUPPORTED)
06	2	2	2	1	2	1-65536	(NOT SUPPORTED)
07	2	2	2	1	2	1-65536	(NOT SUPPORTED)

CONTROL CODES

SOH	= 01
STX	= 02
ETX	= 03
EOT	= 04
ENQ	= 05
ACK	= 06
DLE	= 10
NAK	= 15
WAK	= DLE ; (1F 3B)

```
ENQ PROCEDURE
(A): FROM MASTER TO SLAVE
0  (EOT) = END TRANSMIT
1  P1 = POLLING/SELECT  FUNCTION  (00:SELECT  80:POLLING)
                                  SUPORT SELECT ONLY
2  DID = SLAVE SELECTED DEVICE ID
3  SID = MASTER DEVICE ID
4  ENQ
(B): FROM SLAVE TO MASTER
1  ACK

HEADING PROCEDURE
(A): FROM MAIN TO SLAVE
1  SOH =
2  FMT = (00)
3  DID = SELECTED SLAVE ID
4  SID = MASTER ID
5  FNC = FUNCTION CODE
6  SIZ = SIZE
7  HCS = CHECKSUM OF HEADER (COMPLEMENT OF 'SOH --- SIZ')
```

```
(B): FROM SLAVE TO MASTER
1  ACK = IF RECIEVED CORRECTLY, AND IS ABLE TO RECIVE TEXT THEN 'ACK'
         IF RECIEVED CORRECTLY, BUT ISN'T ABLE TO RECIVE TEXT THEN 'WAK'
         IF RECIEVED NOT CORRECTLY, THEN 'NAK'

(C): FROM MASTER TO SLAVE
   IF RECIEVED CODE FROM SLAVE IS 'ACK' GOTO 'SEND TEXT'
   IF RECIEVED CODE IS 'WAK', WAIT 100 MILI SEC. THEN GOTO (A),
   IF RECIEVED CODE IS 'NAK', GOTO (A)
   IF RECIEVED CODE IS NOT 'ACK', 'NAK', EITHER 'WAK', OR NOT RECIEVED
  ANSWER, MASTER CPU SEND 'ENQ', THEN SLAVE CPU SEND (ACK:'WAK', 'ACK'
  ), THEN SLAVE SEND (ACK:'WAK', 'NAK', 'ACK') AGAIN.

SEND TEXT PROCEDURE
(A): FROM MASTER TO SLAVE
1  STX
2  DATA BYTE 0
3  DATA BYTE 1
     .
     .
N  DATA BYTE N-2
N+1 ETX
N+2 CKS = CHECKSUM (STX --- ETX)
     (EOT)

(B): FROM SLAVE TO MASTER
1  ACK = IF RECIEVED CORRECTLY, AND IS ABLE TO RECIVE TEXT THEN 'ACK'
         IF RECIEVED CORRECTLY, BUT ISN'T ABLE TO RECIVE TEXT THEN 'WAK'
         IF RECIEVED NOT CORRECTLY, THEN 'NAK'

(C): FROM MASTER TO SLAVE
   IF RECIEVED CODE FROM SLAVE IS 'ACK' GOTO 'SEND TEXT'
   IF RECIEVED CODE IS 'WAK', WAIT 100 MILI SEC, THEN GOTO (A),
   IF RECIEVED CODE IS 'NAK', GOTO (A)
   IF RECIEVED CODE IS NOT 'ACK' EITHER 'NAK', OR NOT RECIEVED ANSWER,
  SEND SLAVE 'ENQ', THEN SLAVE SEND (ACK: 'NAK', 'ACK') TO MASTER
  AGAIN.
```

NOTE.

TIME CUT = 100 MILI SEC.

ONCE MASTER CPU SELECT SLAVE CPU, UNTIL 'EOT' CODE IS SEND TO SLAVE CPU, MASTER-SLAVE CONNECTION IS NOT CUT.

IF CURRENT HEADER IS SAME AS LAST HEADER, CURRENT HEADER IS ABLE TO BE OMIT.

HX-20 ORIGINAL COMMANDS

NOTE.

HX-20 ORIGINAL FNC COMMANDS ARE FROM 80H TO FFH.
IF THE BIT6 OF FNC IS 0, THE COMMAND HAS TO RECEIVE THE ASNSER
FROM DEVICE. IF BIT6 IS 0, DESTINATION DEVICE MAY OMIT THE ANSWER.

(1) DISK

SEE TF FUNCTION.

4 CRT COMMAND

SEE SCREEN DOCUMENT.

```
(A) HX-20 EXTERNAL CASSETTE
   WRITTEN BY     KENJI AKAHANE
   REVISION       B-1
   FILE           CASSETTE HXID
   DATE           11.28.1981
   UPDATE         11.18.1982

 NOTE.  A LINE WITH 72 COLUMN '%' MARK IS UPDATED.
        A LINE WITH 72 COLUMN '@' MARK IS UPDATED.

1. HARD FORMAT
    1. APPLE FORMAT
     (A). MICRO CASSETTE
      0: LOW 250 MICRO SEC. HIGH 250 MICRO SEC.                         %
      1: LOW 500 MICRO SEC. HIGH 500 MICRO SEC.                         %
           ----  ----  --                                               %
           |  |  |  |  |                                                %
       -----  ----  ----                                                %
       --> |  | <--      0 (HIGH 250 MICRO SEC)                         %
       --> |     | <--    ONE BIT                                       %
                                                                        %
           -----        -----                                           %
           |   |        |   |                                           %
       -----   ----------   -------                                     %
       --> |     | <---  1 (HIGH 500 MICRO SEC)                         %
     (B). EXTERNAL CASSETTE
      0: LOW 250 MICRO SEC. HIGH 250 MICRO SEC.
      1: LOW 500 MICRO SEC. HIGH 500 MICRO SEC.
    2. BLOCK
     2.1  MOTOR SYNCLONIZED SECTION.
          1 SEC 1 (= STRING OF ¥FF)  (5 COUNTS BY TAPE COUNTER)
     2.2  TAPE SYNCLONIZED SECTION.
          80 BITS 0
     2.3  PREAMBLE
          1 BYTE ¥FF, 1BYTE ¥AA
     2.4  STRING OF DATA
     2.5  POSTAMBLE
          1 BYTE ¥AA, 1BYTE ¥00
```

```
2. SOFT FORMAT
  (A)  EPSON FORMAT
   (1) HEADER BLOCK
      1.  BLOCK LENGTH
          86 BYTES
      2.  FORMAT
       2.1  BLOCK NUMBER AREA (4 BYTES)
          0     COLUMN:'H'
          1 - 2 COLUMN: 00 (BINARY)
          3     COLUMN: SAME BLOCK COUNT (1 - 2)
       2.2  DATA BLOCK  (80 BYTES)
          0 - 3 COLUMN:IDENTIFY  'HDR1'
          4 - 11      :FILE NAME (ASCII CODE)
          12 - 14     :FILE TYPE (ASCII CODE)
          15 - 19     : FILL WITH SPACE
          20          :RECORD TYPE ('F':FIX 'V':VARIABLE)
                      :             '2':FIX, WRITTEN TWICE)
          21          :BLOCK MODE ('0':STOP EACH BLOCK,
                      :            'S':SHORT GAP (GAP = 0.1 SEC)
                      :            'L':LONG GAP (GAP = 1.0 SEC)
          22 - 26     :BLOCK LENGTH (BYTES/BLOCK ASCII CODE)
                      :              '00001' - '09999'
          27 - 31     :NOT USED
          32 - 37     :FILE CREATED DATE (ASCII CODE MMDDYY)
          38 - 43     :FILE CREATED TIME (ASCII CODE HHMMSS)
          44 - 49     :NOT USED
          50 - 51     :VOLUME NUMBER (ASCII CODE '01' - '99')
          52 - 59     :SYSTEM NAME ('HX-20   ')
          60 - 79     :FILL WITH SPACE CODES(UNDEFINED)
       2.3 TRAIL AREA
          0 - 1 COLUMN:CRC (16 BITS) BY CRC-CCITT
          2 - 3 COLUMN:¥AA,¥00                                          @
   (2) DATA BLOCK
      1.   BLOCK LENGTH
           DEFINED BY HEADER
      2.   FORMAT
       2.1 BLOCK NUMBER AREA
          0     COLUMN:'D'
          1 - 2 COLUMN:BLOCK NUMBER (BINARY)(1 - N)
          3     COLUMN:SAME BLOCK WRITE COUNT('1':1 '2':2)
       2.2 DATA AREA
          0 - N-1 COLUMN:DATA STRING (N IS DEFINED BY HEADER)
       2.3 TRAIL AREA
          0 - 1 COLUMN:CRC (16 BITS) BY CRC-CCITT
          2 - 3 COLUMN:¥AA,¥00                                          @
```

```
5.5  WRITE ONE BLOCK
     ON ENTRY  (X):ADDRESS OF THE BUFFER
               (A):BLOCK START MODE
               (B):BLOCK END MODE
     ON EXIT   (C):ERROR FLAG   1:ERROR  0:NORMAL

5.6  STOP WRITE BY EPSON FORMAT
5.7  SEARCH AND READ HEADER BLOCK
     ON ENTRY  (X):ADDRESS OF THE BUFFER
               (A):BLOCK START MODE
               (B):BLOCK END MODE
     ON EXIT   (C):ERROR FLAG    0:NORMAL
5.8  SEARCH AND READ EOF BLOCK
     ON ENTRY  (X):ADDRESS OF THE BUFFER
               (A):BLOCK START MODE
               (B):BLOCK END MODE
     ON EXIT   (C):ERROR FLAG    0:NORMAL
5.9  READ ONE BLOCK
     ON ENTRY  (X):ADDRESS OF THE BUFFER
               (A):BLOCK START MODE
               (B):BLOCK END MODE
     ON EXIT   (A,B):BLOCK NUMBER     (A):¥F0:HEADER  ¥F1:EOF
               (X):VALUE OF BCC
               (C):ERROR FLAG    0:NORMAL   1:ERROR
5.10 STOP READ BY EPSON FORMAT
5.11 REWIND TO TOP OF FILE
5.12 RESET COUNTER
5.13 FEED UNTIL COUNTER VALUE N

BLOCK START MODE
 00:READ/WRITE START FROM STOP.
 01:READ/WRITE START AFTER DEFINED GAP
 -1:READ/WRITE START FROM LEAD TAPE
BLOCK END MODE
 00:STOP AFTER READ/WRITE THIS BLOCK
 01:NON STOP READ/WRITE
 -1:STOP WITH TRAILER TAPE

6. CASSETTE SUBROUTINE CALLED BY APPLICATION PROGRAM
   SEE MAINIO DOCUMENT   ('MAINIO HX1D')


MAIN WORK AREA MEMORY MAP
(FOR EXTERNAL CASSETTE)
 1D5 -     :CURRENT MODE
 1D6 - 1D7:CURRENT BLOCK NUMBER
 1D8 - 1D9:VALUE OF BCC REGISTER (SLAVE MCU)
 1DA - 1DB:NOT USED
 1DC      :BLOCK GAP MODE
 1DD      :ERROR FLAG
 1DE - 1DF:BUFFER TOP ADDRESS
 1E0 - 1E1:BUFFER BOTTOM ADRESS + 1
 1E2 - 1E3:BUFFER SIZE (BYTES)
 1E4 - 1E5:POINTER FOR WRITING DATA TO THE BUFFER
 1E6 - 1E7:POINTER FOR READING DATA FROM THE BUFFER
 1E8 - 1E9:DATA NUMBER IN THE BUFFER
 1EA      :READ TRY LIMIT COUNT
 1EB      :READ TRIED COUNT
```

```
(B): FROM SLAVE TO MASTER
1  ACK = IF RECIEVED CORRECTLY, AND IS ABLE TO RECIVE TEXT THEN 'ACK'
         IF RECIEVED CORRECTLY, BUT ISN'T ABLE TO RECIVE TEXT THEN 'WAK'
         IF RECIEVED NOT CORRECTLY, THEN 'NAK'

(C): FROM MASTER TO SLAVE
   IF RECIEVED CODE FROM SLAVE IS 'ACK' GOTO 'SEND TEXT'
   IF RECIEVED CODE IS 'WAK', WAIT 100 MILI SEC, THEN GOTO (A),
   IF RECIEVED CODE IS 'NAK', GOTO (A)
   IF RECIEVED CODE IS NOT 'ACK', 'NAK', EITHER 'WAK', OR NOT RECIEVED
  ANSWER, MASTER CPU SEND 'ENQ', THEN SLAVE CPU SEND (ACK:'WAK', 'ACK'
  ), THEN SLAVE SEND (ACK:'WAK', 'NAK', 'ACK') AGAIN.

SEND TEXT PROCEDURE
(A): FROM MASTER TO SLAVE
1  STX
2  DATA BYTE 0
3  DATA BYTE 1
   .
   .
N  DATA BYTE N-2
N+1 ETX
N+2 CKS = CHECKSUM (STX --- ETX)
     (EOT)

(B): FROM SLAVE TO MASTER
1  ACK = IF RECIEVED CORRECTLY, AND IS ABLE TO RECIVE TEXT THEN 'ACK'
         IF RECIEVED CORRECTLY, BUT ISN'T ABLE TO RECIVE TEXT THEN 'WAK'
         IF RECIEVED NOT CORRECTLY, THEN 'NAK'

(C): FROM MASTER TO SLAVE
   IF RECIEVED CODE FROM SLAVE IS 'ACK' GOTO 'SEND TEXT'
   IF RECIEVED CODE IS 'WAK', WAIT 100 MILI SEC, THEN GOTO (A),
   IF RECIEVED CODE IS 'NAK', GOTO (A)
   IF RECIEVED CODE IS NOT 'ACK' EITHER 'NAK', OR NOT RECIEVED ANSWER,
  SEND SLAVE 'ENQ', THEN SLAVE SEND (ACK: 'NAK', 'ACK') TO MASTER
  AGAIN.


 NOTE.
  TIME CUT = 100 MILI SEC.
 ONCE MASTER CPU SELECT SLAVE CPU, UNTIL 'EOT' CODE IS SEND TO SLAVE
 CPU, MASTER-SLAVE CONNECTION IS NOT CUT.
 IF CURRENT HEADER IS SAME AS LAST HEADER, CURRENT HEADER IS ABLE TO
 BE OMIT.
```

HX-20 ORIGINAL COMMANDS

NOTE.

HX-20 ORIGINAL FNC COMMANDS ARE FROM 80H TO FFH.
IF THE BIT6 OF FNC IS 0, THE COMMAND HAS TO RECEIVE THE ASNSER
FROM DEVICE. IF BIT6 IS 0, DESTINATION DEVICE MAY OMIT THE ANSWER.

(1) DISK

SEE TF FUNCTION.

4 CRT COMMAND

SEE SCREEN DOCUMENT.

HX-20 ORIGINAL COMMANDS

NOTE.

HX-20 ORIGINAL FNC COMMANDS ARE FROM 80H TO FFH.
IF THE BIT6 OF FNC IS 0, THE COMMAND HAS TO RECEIVE THE ASNSER FROM DEVICE. IF BIT6 IS 0, DESTINATION DEVICE MAY OMIT THE ANSWER.

(1) DISK

SEE TF FUNCTION.

4 CRT COMMAND

SEE SCREEN DOCUMENT.

```
(A) HX-20 EXTERNAL CASSETTE
    WRITTEN BY        KENJI AKAHANE
    REVISION          B-1
    FILE              CASSETTE HXID
    DATE              11.28.1981
    UPDATE            11.18.1982

 NOTE.  A LINE WITH 72 COLUMN '%' MARK IS UPDATED.
        A LINE WITH 72 COLUMN '@' MARK IS UPDATED.

1. HARD FORMAT
   1. APPLE FORMAT
    (A). MICRO CASSETTE
     0: LOW 250 MICRO SEC. HIGH 250 MICRO SEC.                       %
     1: LOW 500 MICRO SEC. HIGH 500 MICRO SEC.                       %
            ----  ----  --                                           %
            |  |  |  |  |                                            %
        -----  ----  ----                                            %
        --> |  | <--      0 (HIGH 250 MICRO SEC)                     %
        --> |     | <--     ONE BIT                                  %
                                                                     %
            -----        -----                                       %
            |   |        |   |                                       %
        -----   ----------   -------                                 %
        --> |     | <---  1 (HIGH 500 MICRO SEC)                     %
    (B). EXTERNAL CASSETTE
     0: LOW 250 MICRO SEC. HIGH 250 MICRO SEC.
     1: LOW 500 MICRO SEC. HIGH 500 MICRO SEC.
   2. BLOCK
    2.1  MOTOR SYNCLONIZED SECTION.
         1 SEC 1 (= STRING OF ¥FF)  (5 COUNTS BY TAPE COUNTER)
    2.2  TAPE SYNCLONIZED SECTION.
         80 BITS 0
    2.3  PREAMBLE
         1 BYTE ¥FF, 1BYTE ¥AA
    2.4  STRING OF DATA
    2.5  POSTAMBLE
         1 BYTE ¥AA, 1BYTE ¥00
```

```
2. SOFT FORMAT
 (A)  EPSON FORMAT
  (1) HEADER BLOCK
     1.  BLOCK LENGTH
         86 BYTES
     2.  FORMAT
      2.1  BLOCK NUMBER AREA (4 BYTES)
         0      COLUMN:'H'
         1 - 2 COLUMN: 00 (BINARY)
         3      COLUMN: SAME BLOCK COUNT (1 - 2)
      2.2  DATA BLOCK  (80 BYTES)
         0 - 3 COLUMN:IDENTIFY  'HDR1'
         4 - 11      :FILE NAME (ASCII CODE)
         12 - 14     :FILE TYPE (ASCII CODE)
         15 - 19     : FILL WITH SPACE
         20          :RECORD TYPE ('F':FIX 'V':VARIABLE)
                     :              '2':FIX, WRITTEN TWICE)
         21          :BLOCK MODE ('0':STOP EACH BLOCK,
                     :             'S':SHORT GAP (GAP = 0.1 SEC)
                     :             'L':LONG GAP (GAP = 1.0 SEC)
         22 - 26     :BLOCK LENGTH (BYTES/BLOCK ASCII CODE)
                     :              '00001' - '09999'
         27 - 31     :NOT USED
         32 - 37     :FILE CREATED DATE (ASCII CODE MMDDYY)
         38 - 43     :FILE CREATED TIME (ASCII CODE HHMMSS)
         44 - 49     :NOT USED
         50 - 51     :VOLUME NUMBER (ASCII CODE '01' - '99')
         52 - 59     :SYSTEM NAME ('HX-20   ')
         60 - 79     :FILL WITH SPACE CODES(UNDEFINED)
      2.3 TRAIL AREA
         0 - 1 COLUMN:CRC (16 BITS) BY CRC-CCITT
         2 - 3 COLUMN:¥AA,¥00                                        @
  (2) DATA BLOCK
      1.  BLOCK LENGTH
          DEFINED BY HEADER
      2.  FORMAT
      2.1 BLOCK NUMBER AREA
         0      COLUMN:'D'
         1 - 2 COLUMN:BLOCK NUMBER (BINARY)(1 - N)
         3      COLUMN:SAME BLOCK WRITE COUNT('1':1 '2':2)
      2.2 DATA AREA
         0 - N-1 COLUMN:DATA STRING (N IS DEFINED BY HEADER)
      2.3 TRAIL AREA
         0 - 1 COLUMN:CRC (16 BITS) BY CRC-CCITT
         2 - 3 COLUMN:¥AA,¥00                                        @
 (3) EOF BLOCK
    1.  BLOCK LENGTH
        80 BYTES
    2.  FORMAT
      2.1 BLOCK NUMBER AREA
          0     COLUMN:'E'
          1 - 2 COLUMN: BLOCK NUMBER (LAST DATA RECORD BLOCK + 1)
          3     COLUMN: SAME BLOCK WRITE COUNT (1 - 2)
      2.2 DATA AREA
          0 - 3 COLUMN:IDENTFY (ASCII CODE 'EOF ')
          4 - 79       :FILL WITH SPACE CODES(UNDEFINED)
      2.3 TRAIL AREA
          0 - 1 COLUMN:CRC BY CRC-CCITT
          2 - 3 COLUMN:¥AA,¥00

(B): BINARY DUMP FORMAT
 RECORD
 ----------------------------------------------------    ----------------
 | NUMBER OF    | ADDRESS     | DATA 1 | DATA 2 |        | DATA N | CHECK|
 | DATA IN THE  | OF THE TOP  |        |        |        |        | SUM  |
 | RECORD       | OF DATA     |        |        |        |        |      |
 ----------------------------------------------------    ----------------

 BYTE 0:DATA NUMBER IN THE RECORD: ONE BYTE LENGTH (VALUE N)
 BYTE 1,2:ADDRESS OF THE TOP OF DATA:TWO BYTES LENGTH
 BYTE 3:DATA 1: ONE BYTE
     :
     :
 BYTE N+2: DATA N
 BYTE N+3: CHECKSUM (SUM FROM RECORD NUMBER TO CHECKSUM = 0)
                (ADDITION, BUT NOT EXCLUSIVE OR)
 LAST RECORD
 BYTE 0:DATA LENGTH = 0
 BYTE 1,2:ADDRESS = PROGRAM START ADDRESS (PC)
                  (0000: NONEXISTENT ENTRY POINT)
 BYTE 4:CHECKSUM

EXAMPLE
 CONTENTS OF ADDRESS ¥1000 - ¥1002 = ¥01, ¥02, ¥03.
 DATA RECORDS ARE
 03       FIRST RECORD RECORD SIZE
 10       DATA ADDRESS (HIGH BYTE)
 00       DATA ADDRESS (LOW BYTE)
 01       DATA 1
 02       DATA 2
 03       DATA 3
 E7       CHECKSUM OF FIRST RECORD
 00       LAST RECORD (RECORD SIZE = 0)
 10       ENTRY ADDRESS (HIGH)
 00       ENTRY ADDRESS (LOW)
 F0       CHECKSUM
```

```
3. COMMAND(FROM MAIN CPU TO SLAVE CPU)
 1.  SET COUNTER VALUE (COUNTER=16 BITS HEXADECIMAL CODE
 2.  READ COUNTER VALUE
 3.  SEARCH HEADER BLOCK AND READ (EPSON FORMAT)
 4.  READ NEXT BLOCK (BY EPSON FORMAT).
 5.  SKIP N BYTES.
 6.  WRITE HEADER BLOCK (BY EPSON FORMAT).
 7.  WRITE NEXT BLOCK (BY EPSON FORMAT).
 8.  WRITE TRAILER
 9.  SEEK BY COUNTER VALUE.
 10. REWIND TO TOP OF FILE.
 11. SET BCC REGISTER
 12. READ BCC REGISTER
 13. SET MICRO CASSETTE MODE
 14. READ CASSETTE STATUS REGISTER
 15. CLEAR CASSETTE STATUS REGISTER
 16. OPEN TO WRITE (WRITE ONE CHARACTER MODE)
 17. OPEN TO READ (READ ONE CHARACTER MODE)
 18. STOP TO READ (READ ONE CHARACTER MODE)

4. ANOTHER FUNCTION
  4.1 BY KEY BOARD (MICRO CASSETTE)
   .1 STOP
   .2 REWIND
   .3 FAST FEED
   .4 PLAY (SLOW FEED)
   .5 COUNTER RESET

5. CASSETTE MAIN CPU SUBROUTINE
  5.1  SET CASSETTE PARAMETER
       ON ENTRY  (X):PARAMETER ADDRESS
                      PARAMETER 10 BYTES
  5.2  SET EPSON FORMAT PARAMETER
       ON ENTRY  (A):BLCOK MODE    0:STOP EACH BLOCK    1:SHORT GAP
                                   2:LONG GAP
                 (B):DATA WRITE MODE  0:WRITE ONE TIME  1:WRITE TWICE
                 (X):BLOCK SIZE
  5.3  WRITE HEADER BLOCK
       ON ENTRY  (X):ADDRESS OF BUFFER
                 (A):BLOCK START MODE
                 (B):BLOCK END MODE
       ON EXIT   (C):ERROR FLAG  1:ERROR   0:NORMAL
  5.4  WRITE EOF BLOCK
       ON ENTRY  (X):ADDRESS OF BUFFER
                 (A):BLOCK START MODE
                 (B):BLOCK END MODE
       ON EXIT   (C):ERROR FLAG  1:ERROR   0:NORMAL
```

```
5.5  WRITE ONE BLOCK
     ON ENTRY  (X):ADDRESS OF THE BUFFER
               (A):BLOCK START MODE
               (B):BLOCK END MODE
     ON EXIT   (C):ERROR FLAG   1:ERROR  0:NORMAL

5.6  STOP WRITE BY EPSON FORMAT
5.7  SEARCH AND READ HEADER BLOCK
     ON ENTRY  (X):ADDRESS OF THE BUFFER
               (A):BLOCK START MODE
               (B):BLOCK END MODE
     ON EXIT   (C):ERROR FLAG    0:NORMAL
5.8  SEARCH AND READ EOF BLOCK
     ON ENTRY  (X):ADDRESS OF THE BUFFER
               (A):BLOCK START MODE
               (B):BLOCK END MODE
     ON EXIT   (C):ERROR FLAG    0:NORMAL
5.9  READ ONE BLOCK
     ON ENTRY  (X):ADDRESS OF THE BUFFER
               (A):BLOCK START MODE
               (B):BLOCK END MODE
     ON EXIT   (A,B):BLOCK NUMBER     (A):¥F0:HEADER  ¥F1:EOF
               (X):VALUE OF BCC
               (C):ERROR FLAG    0:NORMAL   1:ERROR
5.10 STOP READ BY EPSON FORMAT
5.11 REWIND TO TOP OF FILE
5.12 RESET COUNTER
5.13 FEED UNTIL COUNTER VALUE N

BLOCK START MODE
 00:READ/WRITE START FROM STOP.
 01:READ/WRITE START AFTER DEFINED GAP
 -1:READ/WRITE START FROM LEAD TAPE
BLOCK END MODE
 00:STOP AFTER READ/WRITE THIS BLOCK
 01:NON STOP READ/WRITE
 -1:STOP WITH TRAILER TAPE

6. CASSETTE SUBROUTINE CALLED BY APPLICATION PROGRAM
   SEE MAINIO DOCUMENT    ('MAINIO HX10')                            ﾗ
                                                                     ﾗ
MAIN WORK AREA MEMORY MAP                                            ﾗ
(FOR EXTERNAL CASSETTE)                                              ﾗ
 1D5 -     :CURRENT MODE                                             ﾗ
 1D6 - 1D7:CURRENT BLOCK NUMBER                                      ﾗ
 1D8 - 1D9:VALUE OF BCC REGISTER (SLAVE MCU)                         ﾗ
 1DA - 1DB:NOT USED                                                  ﾗ
 1DC       :BLOCK GAP MODE                                           ﾗ
 1DD       :ERROR FLAG                                               ﾗ
 1DE - 1DF:BUFFER TOP ADDRESS                                        ﾗ
 1E0 - 1E1:BUFFER BOTTOM ADRESS + 1                                  ﾗ
 1E2 - 1E3:BUFFER SIZE (BYTES)                                       ﾗ
 1E4 - 1E5:POINTER FOR WRITING DATA TO THE BUFFER                    ﾗ
 1E6 - 1E7:POINTER FOR READING DATA FROM THE BUFFER                  ﾗ
 1E8 - 1E9:DATA NUMBER IN THE BUFFER                                 ﾗ
 1EA       :READ TRY LIMIT COUNT                                     ﾗ
 1EB       :READ TRIED COUNT                                         ﾗ
```

```
  THE BIT MAP(8 BITS) FOR BANK-1 IS :

                           -----
THE MSB OF ¥13B              *          ¥E000  OF  BANK-1
                           -----
                             *          ¥C000  OF  BANK-1
                           -----
                             *          ¥A000  OF  BANK-1
                           -----
                             *          ¥8000  OF  BANK-1
                           -----
                             *          ¥6000  OF  BANK-1
                           -----
                             *          ¥4000  OF  BANK-1
                           -----
                             *          ¥2000  OF  BANK-1
                           -----
THE LSB OF ¥13B              *          ¥0000  OF  BANK-1
                           -----

  * = 0  :  IF NO HEADER OF APPLICATION EXISTS ON ONE ROM.
  * = 1  :  IF SOME HEADER OF APPLICATIONS EXIST ON ONE ROM.


  THE LINK TABLE(4 BYTES) IS :

     ¥13C-13F : LINK TO RAM APPLICATION

  IF NO HEADER OF RAM APPLICATION EXISTS, THEN LINK TABLE IS
 "/:/E/¥FF/¥FF/".


.PA

 2.3  HEADER OF BASIC APPLICATION

    HEADER OF BASIC APPLICATION IS DIFFERENT FROM THE
   HEADER OF ROM APPLICATION AND USER APPLICATION(2.1)

 1) LINK OFFSET (2 BYTES SIZE)

     ¥FFFF : NOT EXIST NEXT HEADER
     OTHERS : LINK FOR NEXT NAME

 2) FILE NAME (8 BYTES SIZE)
     FILE NAME


   BASIC APPLICATION HASN'T LINK WITH NEITHER ROM
  APPLICATION NOR USER APPLICATION.
   MENU LISTS BASIC APPLICATION NAME AFTER ROM APPLICATIONS
  AND USER APPLICATIONS.
```

2.4 MAKING BIT MAP AND LINK TABLE

BEFORE HC SYSTEM INITIALIZATION(CTRL/I),THERE ISN'T NEITHER BIT MAP NOR LINK TABLE. BEFORE INITIALIZATION, FIRST MENU LISTS "CTRL/@ INITIALIZE", MONITOR, AND DUMY NAME (19 MAX). USER WANTS TO INITIALIZE HC SYSTEM, TYPE CTRL/@. AFTER HC SYSTEM INITIALIZATION, MENU MAKES BIT MAP AND A LINK TABLE(2.2). LINK STARTS FROM ¥D000(MONITOR). NEXT, MENU LOOKS ¥A000 (BOTH BANK-0 AND BANK-1).NEXT, MENU LOOKS ¥8000.NEXT ¥6000. NEXT ¥4000. MENU SETS BIT MAP ACCORDING TO EXISTING A HEADER OF APPLICATION AND WRITES "/:/E/¥FF/¥FF" IN LINK TABLE.

IF USER WANTS TO MAKE PROGRAMS OR RUN PROGRAM IN RAM, THEN USER WRITES THE HEADER OF USER PROGRAM AND REWRITES LINK TABLE. SO USER REWRITES LINK TABLE AND LINKS TO THE HEADER OF USER PROGRAM. THE HEADER OF USER PROGRAM IS WRITTEN ACCORDING TO APPLICATION ID CONFIGURATION.(2.1) IN THE HEADER OF USER PROGRAM, LINK ADDRESS POINTS TO THE NEXT HEADER OF USER PROGRAM .

FOR EXAMPLE, IF THERE A HEADER OF RAM ON ¥1000,

```
  ¥1000  /:/A/¥FF/¥FF/¥10/¥20/U/S/E/R/-/A/¥00/
  ¥13C   /:/A/¥10/¥00/
```

2.5 LISTING MENU

FIRST MENU LISTS ROM APPLICTIONS ACCORDING TO BIT MAP. IF THERE ARE SOME LINK FOR USER PROGRAM, USER PROGRAM ARE LISTED. IF THERE ARE SOME BASIC APPLICATION, THEN BASIC APPLICATIONS ARE LISTED. (SEE EXAMPLE)

.PA

3 EXAMPLE

```
                 BANK-0                         BANK-1

  0000   -------------------

  1000   : A ¥FF ¥FF ¥10 ¥20
            U S E R - A ¥00
                  :
  1FFF   -------------------

  3FFF   -------------------            -------------------
  4000                                  : A ¥50 ¥00 ¥40 ¥18
                                           A P L C - 5 ¥00
                                                 :
  4FFF   -------------------            -------------------
  5000                                  : A ¥FF ¥FF ¥50 ¥25
                                           A P L C - 4 ¥00
                                                 :
  5FFF   -------------------            -------------------
  6000   : A ¥FF ¥FF ¥60 ¥20
            A P L C - 2 ¥00
                  :
  7FFF   -------------------            -------------------
  8000   : B ¥FF ¥FF ¥80 ¥10            : A ¥FF ¥FF ¥80 ¥33
            B A S I C ¥00                  A P L C - 3 ¥00
                  :
  8FFF   -------------------            -------------------
                  :
  BFFF   -------------------            -------------------
```

```
                  :
  CFFF   -------------------            -------------------
  D000   : A ¥FF ¥FF ¥00 ¥33
            M O N I T O R ¥00
                  :
  DFFF   -------------------
```

```
      AND 2 BASIC APPLICATIONS (APLC-1 & APLC-2).

      BIT MAP IS :

         ¥13A       01011000 (BINARY)
         ¥13B       00010100 (BINARY)

      LINK TABLE IS :

         ¥13C        /:/A/¥10/¥00/

  .PA
       MENU IS BEING SHOWN:

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

```
MONITOR DOCUMENT

DATE          01.11.1982
AUTHOR        K. AKAHANE
UPDATE BY     M.HANAOKA
              02.22.1982
              04.02.1982    (K.A)
              06.06.1982    (M.H)
              06.22.1982    (K.A)
FILE NAME     MONITOR HX1D

1. MONITOR COMMAND

 1.1 SET TO MEMORY
  S      (S 'ADDRESS')
   PARAMETER
   1. ADDRESS (HEXA DECIMAL)

     WHEN WE ENTER 'S', ADDRESS AND 'CR', THE CONTENT OF DESIGNATE IS
    DISPLAYED. THEN WE CAN CHANGE CONTENT TO ENTER 'HEXA DECIMAL CODE'
    AND 'CR'. CONTINUOUSLY THE DISPLAY WILL DISPLAY NEXT ADDRESS
    AND CONTENT. IF WE WANT TO STOP 'S' COMMAND MODE, ENTER '.' AND
    FOLLOWING 'CR'. IF WE NEED NOT CHANGE THE CONTENT, ENTER 'CR' ONLY.

  EXAMPLE
   (SET 00 TO 1000-1003)
    -S1000'CR'             .... SET 'S' MODE COMMAND
    -S1000 0A 00'CR'       .... S1000 0A :DISPLAY BY MONITOR, ENTER 00'CR'
    -S1001 03 00'CR'
    -S1002 0C 00'CR'
    -S1003 0D 00'CR'
    -S1004 0E .'CR'
                           .... LAST ENTRY DATUM IS .'CR'

 1.2 DUMP MEMORY
   D     (D 'ADDRESS')
   PARAMETER
   1. ADDRESS (HEXA DECIMAL)
     IF LCD IS SELECTED AS DISPLAY, FOLLOWING THE ADDRESS THE CONTENTS
    OF FIVE BYTES ARE DISPLAYED PER A LINE. IF 'ADDRESS' IS OMITTED,
    THE CONTENTS OF FOLLOWING ADDRESS IS DISPLAYED.  AFTER EXECUTE      %
    'D' COMMAND, 'D' CHARACTER IS STILL DISPLAYED. IF WE WANT TO        %
    DISPLAY NEXT CONTENTS, WE ENTER ONLY 'CR'.                          %

   EXAMPLE
  ----------------------
  |-D1000'CR'          |      WE ENTER  ' D 1 0 0 0 CR '
  |1000: 00 01 02 03 04|
  |1005: 05 06 07 08 09|
  |100A: 0A 0B 0C 0D 0E|
  ----------------------
             |
             V
  ----------------------
  |-D'CR'              |      WE ENTER  ' CR '
  |100F: 0F 10 11 12 13|
  |1014: 14 15 16 17 18|
  |1019: 19 1A 1B 1C 1D|
  ----------------------

 **** NOTE FOR SET AND DUMP ****
  ORDINARY ADDRESS 0 TO 4D ARE PROTECTED TO ACCESS.
  IF YOU WANT TO ACCESS THESE ADDRESS THEN YOU HAVE TO SET BIT 7 OF 7F.
```

1.3 GO (EXECUTE PROGRAM)

G (G 'ADDRESS' , 'BREAK ADDRESS 1')

PARAMETER

1. ADDRESS (HEXA DECIMAL)
   THE PROGRAM COUNTER IS SET TO DESIGNATE VALUE.
2. BREAK ADDRESS (HEXA DECIMAL)
   WE CAN SET BREAK POINT ONE POINT MAX. WHEN VALUE OF THE PROGRAM COUNTER WILL BE SAME AS 'BREAK ADDRESS 1' THE 'TRAP INTERRUPT' WILL BE CAUSED, THEN DISPLAY CONTENTS OF REGISTERS AND WE CAN ENTER COMMAND AGAIN.

NOTE.

THE CONTENT OF 'BREAK POINT' IS CHANGED TO '00' BY THE MONITOR TO CAUSE 'TRAP'. AFTER PROGRAM COUNTER REACHED BREAK POINT ANDCAUSED 'TRAP', THE ORIGINAL CONTENT IS RECOVERED. THE BREAK POINTES ARE ONLY EFFECTIVE ON THE RAM.

EXAMPLE

WHEN CONTENES OF ADDRESS 1000 - 1003 ARE 01(NOP), 01, 01

```
-------------------------
|-G1000,1002            |
|                       |
|  A=00  B=00   X=0000  |
|  C=00  S=00FF P=E000  |
-------------------------

            |
            V
-------------------------
|-                      |
|BREAK                  |
|  A=00  B=00   X=0000  |
|  C=00  S=00FF P=1002  |
-------------------------
```

**** NOTE OF TRAP ****

WHEN TRAP IS CAUSED, IF THE ADDRESS IS NOT BREAK POINT, THE MONITOR DISPLAY 'TRAP ' ON THE LCD.

1.4 BACK TO BASIC

B

RETURN TO THE ROUTINE WHICH CALLED THE MONITER.

1.5 EXAMINE REGISTERS

X

THE CONTENTS OF REGISTERS ARE DISPLAYED ON THE THIRD LINE AND FORTH LINE. AT FIRST THE CONTENT OF ACCUMULATOR A IS DISPLAYED ON THE FIRST LINE.

IF YOU WANT TO CHANGE ITS VALUE, ENTER HEXA DECIMAL VALUE AND FOLLOWING 'RETURN', OR IF YOU DO NOT WANT TO CHANGE, ENTER 'RETURN' ONLY. AFTER 'RETURN', THE CONTENT OF ACCUMULATOR B IS DISPLAYED ON THE FIRST LINE.

THE CONTENTS OF THE FIRST LINE ARE CHANGED CYCLIC, (ACC A ---> ACC B ---> INDEX REGISTER ---> CONDITION CODES ---> STACK POINTER ---> PROGRAM COUNTER)

EXAMPLE

```
-------------------------
|-X A=20 00             |
|                       |
| A=20 B=00    X=1000   |
| C=C0 S=07FF P=2000    |
-------------------------
              |   (ENTER 00 'CR')
              V
-------------------------
|-X B=00                |
|                       |
| A=00 B=00    X=1000   |
| C=C0 S=07FF P=2000    |
-------------------------
```

1.6 READ FILE

R (R 'DEVICE','FILE NAME',R)

PARAMETER

1. DEVICE 'C': CASSETTE 'M': MICRO CASSETTE 'R': ROM CASSETTEE
   '0' - '9' : SERIAL COMMUNICATION (INCLUDE DISK)
2. FILE NAME FILENAME (EIGHT BYTES MAX) . FILE TYPE (THREE BYTES)
   BINALY DATA ARE LOAD TO MEMORY FROM EXTERNAL STORAGE.
   THE LOADING START ADDRESS IS ADDED TO OFFSET VALUE DEFINED BY 'IO' COMMAND. BUT ENTRY ADDRESS IS NOT ADDED TO.
   IF YOU ADD 'R' OPTION, THE PROGRAM COUNTER IS SET TO 'ENTRY ADDRESS' AFTER LOADING.

EXAMPLE

```
A00000'CR'          .... DEFINE OFFSET
-RC,HCPROG.COM,R
```

1.7 VERIFY FILE

V (V 'DEVICE','FILE NAME')

PARAMETER

1. DEVICE 'C': CASSETTE 'M': MICRO CASSETTE 'R': ROM CASSETTEE
2. FILE NAME FILENAME (EIGHT BYTES MAX) . FILE TYPE (THREE BYTES)

NOTE.

DEVICE NO. '0' - '6' ARE SUPPORTED IN THE FOLLOWING VERSION

US VERSION (VERSION 1, VERSION 2)

NIPPON VERSION (VERSION 1)

1.8 WRITE FILE

W (W 'DEVICE','FILE NAME')
PARAMETER
1. DEVICE 'C': CASSETTE 'M': MICRO CASSETTE
'0' - '9': SERIAL COMMUNICATION
2. FILE NAME FILENAME (EIGHT BYTES MAX) . FILE TYPE (THREE BYTES)

THE CONTENTS OF MEMORIES FROM "TOP ADDRESS" TO "BOTTOM ADDRESS" ARE SAVED TO EXTERNAL STORAGES. "OFFSET VALUE" IS ADDED TO ADDRESS DATA (INCLUDE STARTING ADDRESS).

BEFOR EXECUTE 'W' COMMAND, 'TOP ADDRESS', 'BOTTOM ADDRESS', 'START ADDRESS' AND 'OFFSET VALUE' MUST BE DEFINED.

1.9 SET ANY VALUES THAT IS NEEDED BY R, W AND V COMMANDS.

1.9.1 SET TOP ADDRESS.

PARAMETER
1. ADDRESS (HEXA DECIMAL)

1.9.2 SET BOTTOM ADDRESS.

PARAMETER
1. ADDRESS (HEXA DECIMAL)

THIS COMMAND SET THE BOTTOM ADDRESS.

1.9.3 SET OFFSET

PARAMETER
1. ADDRESS (HEXA DECIMAL)

THIS COMMAND SET THE VALUE OF OFFSET WHICH IS USED BY 'R' OR 'W' COMMAND.

1.9.4 SET START ADDRESS.

PARAMETER
1. ADDRESS (HEX DECIMAL)

THIS COMMAND SET THE START ADDRESS.
WHEN WE EXUCUTE 'R' COMMAND WITH 'R' OPTION, PROGRAM COUNTER IS SET TO THIS VALUE. START ADDRESS IS ADDED OFFSET VALUE.

NOTE. THESE ADDRESS DATA (AT, AB, AS, AO) IS CLEARED AFTER 'R' OR 'W' COMMAND.

IN W COMMAND ADDRESS IS USED AS FOLLOWS.
TOP ADDRESS 'LOWER LIMIT OF DUMP ADDRESS'
LAST ADDRESS 'UPPER LIMIT OF DUMP ADDRESS'
OFFSET ADDRESS 'OFFSET VALUE TO DUMP'
IN R COMMAND, ADDRESS IS USED AS FOLLOWS.
TOP ADDRESS 'LOWER LIMIT ADDRESS WE CAN LOAD'
BOTTOM ADDRESS 'UPPER LIMIT ADDRESS WE CAN LOAD'
OFFSET ADDRESS 'OFFSET VALUE'

IN R COMMAND, IN SYSTEM, CHECK ADDRESS TO AVOID DESTROY CONTENTS OF MEMORY

EXAMPLE
DUMP 1000 - 1FFF THEN LOAD 2000 -2FFF AND EXECUTE FROM 2100

```
--------------------------
|-A                      |
|                        |
|                        |
|                        |
--------------------------
            |
            V  (CR)
--------------------------
|-A T=0000 1000          |     * SET TOP ADDRESS.
|                        |     * ENTER '1000'
| A=00 B=12    X=1234    |
| C=D8 S=0166 P=D000     |
--------------------------
            |
            V  (CR)
--------------------------
|-A L=0000 1FFF          |     * SET LAST ADDRESS.
|                        |
| A=00 B=12    X=1234    |
| C=D8 S=0166 P=D000     |
--------------------------
            |
            V  (CR)
--------------------------     * SET OFFSET.
|-A O=0000 1000          |
|                        |
| A=00 B=12    X=1234    |
| C=D8 S=0166 P=D000     |
--------------------------
            |
            V  (CR)
--------------------------
|-A E=0000 2100          |    * SET START (ENTRY) ADDRESS.
|                        |
| A=00 B=12    X=1234    |
| C=D8 S=0166 P=D000     |
--------------------------
```

1.10 SET KEY DATA TO INITIAL STACK

K 'KEY CHARACTER STRING' 'CTRL/@'
WHEN POWER SWITCH IS ON, TO START AUTOMATICALLY, SET KEY DATA TO KEY INITIAL STACK. WHEN POWER SWITCH IS ON, THESE CHARACTERS ARE PUSHED TO KEY STACK AS IF THESE CHRACTERS ARE ENTERED FROM KEYBOARD.

KEY CHARACTER STRING: 16 BYTES MAX
FUNCTION CODE: TWO BYTES
TO CLEAR INITIAL KEY STACK, K'CR' (NULL STRING).
LAST CHRACTER: CTRL/@ (00)

EXAMPLE

```
-K4RUN100'CR''CTRL/@'
          (1:SELECT BASIC BY MENU ROUTINE)
          (2:RUN100'CR'    BY BASIC)
```

NOTE.
WHEN TURN ON, THE WAY TO CANCEL AUTO START PROCESURE IS 'TURN ON POWER SWTCH' ON PRESSING 'BREAK' KEY. IF YOU PRESS 'BREAK' KEY, CANCEL TO READ FROM INITIAL KEY STACK.

```
1.11 CALL 'SUBROUTINE' COMMNAD                                        @
     CALL DESTINETED SUBROUTINE AND RETURN TO THE MONITOR.            @
  C  (C  'SUBROUTINE ENTRY POINT')                                    @

 MEMORY MAP
                                                                      @
  60        :TEMPORARY                                                @
  61        :TEMPORARY                                                @
  62        :TEMPORARY                                                @
  63        :TEMPORARY                                                @
  64        :TEMPORARY                                                @
  65        :TEMPORARY                                                @
  66        :TEMPORARY                                                @
  67        :TEMPORARY                                                @
  68        :TEMPORARY                                                @
  69        :VALUE OF THE CONDITION CODE REGISTER                     @
  6A        :VALUE OF THE ACCUMULATOR B                               @
  6B        :VALUE OF THE ACCUMULATOR A                               @
 6C -  6D :VALUE OF THE INDEX REGIDTER                                @
 6E -  6F :VALUE OF THE PROGRAM COUNTER                               @
                                                                      @
 2A0 -  2A1:BREAK POINT ADDRESS                                       @
 2A2        :BREAK POINT DATA                                         @
 2A3        :SAVE LCD STATUS (SAVE CONTENTS OF ¥280)                  @
 2A4        :BINARY DUMP/LOAD PACKET (MODE)                           @
 2A5 -  2A6:                         (BUFFER ADDRESS)                 @
 2A7 -  2AE:                         (FILE NAME)                      @
 2AF -  2B6:                         (FILE TYPE)                      @
 2B7 -  2B8:                         (TOP ADDRESS)                    @
 2B9 -  2BA:                         (LAST ADDRESS)                   @
 2BB -  2BC:                         (OFFSET VALUE)                   @
 2BD -  2BE:                         (ENTRY POINT)                    @
 2BF -  2C0:VALUE OF THE STACK POINTER                                @
 2C1 -  2C2:RETURN ADDRESS FOR 'B' COMMAND                            @
 2C3 -  2C4:ADDRESS: LAST COLUMN OF FIRST LINE OF LCD (BUFFER)        @
 2C5        :SAVE VALUE OF RUNMOD (SAVE CONTENTS OF ¥7B)              @
```

```
CREATED BY   K.AKAHANE
REVISION     B-1
FILE NAME    SLAVE HXID
DATE         01.07.1982
UPDATE       06.07.1982
UPDATE       11.16.1982

NOTE. A LINE WITH 72 COLUMN '@' IS UPDATED.
NOTE. A LINE WITH 2 COLUMN ';' DELETED


PART 1.
 PROCEDURE OF COMUNICATION BETWEEN MAIN-CPU AND SLAVE-CPU.

1. SYSTEM COMMANDS
  'ACK'=¥01

 ¥00: READY CHECK
     1.M  ¥00               1.S 'ACK'
 ¥01: SET INITIAL STATUS
     1.M  ¥01               1.S 'ACK'
     CONSTANTS WHICH ARE INITIALIZED                                  @
       A:CLEAR BUFFER, SLAVE BUFFER POINTER
       B:RECIVED COMMAND = NONE
       C:GENERATING POLYNOMIAL  ¥8408
       D:BCC REGISTER 0
       E:RS232C BIT RATE 300 BPS                                      @
       F:RS232C BIT LENGTH 7
       G:RS232C PARITY EVEN, STOP BITS 1, CHECK CD
       H:CLEAR RS232 STATUS REGISTER
       I:SUPER VISOR COMMAND (MASK)
       J:EXTERNAL CASSETTE LONG BIAS ¥139 (2000 BAUD)
       K:EXTERNAL CASSETTE SHORT BIAS ¥9D (1000 BAUD)
       L:EXTERNAL CASSETTE BOUNDARY VALUE ¥E3
       M:EXTERNAL CASSETTE GAP BYTES 125
       N:INTERNAL CASSETTE LONG BIAS LOW PULSE TIME  ¥D0
       O:INTERNAL CASSETTE LONG BIAS HIGH PULSE TIME  ¥68
       P:INTERNAL CASSETTE SHORT BIAS ¥4E
       Q:INTERNAL CASSETTE BOUNDARY VALUE ¥EA
       R:INTERNAL CASSETTE GAP BYTES  250
       S:INTERNAL CASSETTE COUNTER VALUE 0
       T:INTERNAL CASSETTE COUNTER PULSE LOW
       U:TIMER OVERFLOW INITIAL COUNTER  10
 ¥02: RESET SLAVE-CPU (I/O HOT START INITIALIZE)                      @
     1.M  ¥02               1.S 'ACK'
 ¥03: OPEN SUPER VISOR MASK
     1.M  ¥03               1.S 'ACK'
     2.M  'CODE' =¥AA       1.S 'ACK'
 ¥04: CLOSE SUPER VISOR MASK
     1.M  ¥04               1.S 'ACK'
```

```
¥05: READ CONTENTS OF MEMORY
     1.M  ¥05                1.S 'ACK'
     2.M ADDRESS (HIGH)      2.S 'ACK'
     3.M ADDRESS (LOW)       3.S DATA
¥06: STORE TO MEMORY
     1.M  ¥06                1.S 'ACK'
     2.M ADDRESS (HIGH)      2.S 'ACK'
     3.M ADDRESS (LOW)       3.S 'ACK'
     4.M DATA                4.S 'ACK'
¥07: OR TO MEMORY
     1.M  ¥07                1.S 'ACK'
     2.M ADDRESS (HIGH)      2.S 'ACK'
     3.M ADDRESS (LOW)       3.S 'ACK'
     4.M DATA                4.S 'ACK'
¥08: AND TO MEMORY
     1.M  ¥08                1.S 'ACK'
     2.M ADDRESS (HIGH)      2.S 'ACK'
     3.M ADDRESS (LOW)       3.S 'ACK'
     4.M DATA                4.S 'ACK'
¥09: BARCODE READER POWER ON
     1.M  ¥09                1.S 'ACK'
¥0A: BARCODE READER POWER OFF
     1.M  ¥0A                1.S 'ACK'
¥0B: JUMP TO LOCATION
     1.M  ¥0B                1.S 'ACK'
     2.M ADDRESS (HIGH)      2.S 'ACK'
     3.M ADDRESS (LOW)       3.S 'ACK'
¥0C: BREAK
     1.M  ¥0C                1.S 'BREAK ACK'  = ¥02
¥0D: POWER OFF
     1.M  ¥0D                1.S 'ACK'
     2.M  ¥AA
¥0E: SELF CHECK  (DELETED)                                       %
     1.M  ¥0E                1.S OR OF BCC REGISTER (0:OK)       %
                                                                 %
NOTE. SUPER VISOR COMMAND = ¥05, ¥06, ¥07, ¥08, ¥0B, ¥0D
NOTE. COMMAND ¥9 AND ¥A DO NOT WORK IN VERSION 1. IN VERSION 2 WORK    a
```

```
2. PRINTER COMMANDS
 ¥10: PRINT BY GRAPHIC PATTERN                                                   a
     1.M  ¥10                1.S 'ACK'
     2.M DATA (LS 6 BITS)  1.S 'PRINTER ACK' =¥11

     DATA TYPE.     7    6    5    4    3    2    1    0
                     <NOT USED> <LAST>                  <FIRST DOT>
   REPEAT 24 TIMES FROM 1 TO 2 TO PRINT ONE DOT LINE.                            a
 ¥11: LINE FEED N DOT LINES
     1.M  ¥11                1.S 'ACK'
     2.M  DATA (LINE N)      2.S 'PRINTER ACK'=¥11

 ¥12: FEED ONE DOT LINE (NON BRAKE)
      (TURN ON MICRO PRINTER'S MOTOR 100 M SEC)                                  a
     1.M  ¥12                1.S 'ACK'

2. EXTERNAL CASSETTE COMMANDS
 ¥20: EXTERNAL CASSETTE READY CHECK
    1.M ¥20                          1.S 'ACK'
 ¥21: SET CASSETTE PARAMETER
    1.M ¥21                          1.S 'ACK'
    2.M LONG BIAS (HIGH)             2.S 'EXTERNAL CASSETTE ACK' =¥21
    3.M LONG BIAS (LOW)              3.S 'EXTERNAL CASSETTE ACK'
    4.M SHORT BIAS (HIGH)            4.S 'EXTERNAL CASSETTE ACK'
    5.M SHORT BIAS (LOW)             5.S 'EXTERNAL CASSETTE ACK'
    6.M BOUNDARY VALUE (H)           6.S 'EXTERNAL CASSETTE ACK'
    7.M BOUNDARY VALUE (L)           7.S 'EXTERNAL CASSETTE ACK'
    8.M GAP BYTES (H)                8.S 'EXTERNAL CASSETTE ACK'
    9.M GAP BYTES (L)                9.S 'EXTERNAL CASSETTE ACK'
 ¥22: REMOTE ON                                                                  a
    1.M ¥22                          1.S 'ACK'
 ¥23: REMOTE OFF                                                                 a
    1.M ¥23                          1.S 'ACK'
 ¥24: WRITE ONE BLOCK BY EPSON FORMAT
    1.M ¥24                          1.S 'ACK'
    2.M BLOCK LEAD TAPE MODE         2.S 'EXTERNAL CASSETTE ACK'
        00:WITH LONG GAP (DEPEND OF VALUE OF 'GAP BYTES',                       a
                             DEFAULT VALUE = 125 BYTES)                         a
        01:WITH SHORT GAP (15 BYTES ¥FF)                                        a
        FF:WITH LEAD TAPE (625 BYTES ¥FF)                                       a
    3.M BLOCK TRAIL MODE             3.S 'EXTERNAL CASSETTE ACK'
        00:WITH LONG GAP (DEFAULT 125 BYTES ¥FF) AND STOP                       a
                             DEFAULT VALUE = 125 BYTES)                         a
        01:WITH SHORT GAP (15 BYTES ¥FF) AND NONSTOP                            a
        FF:WITH TRAIL TAPE (625 BYTES ¥FF) AND STOP                             a
    4.M BLOCK SIZE (H)               4.S 'EXTERNAL CASSETTE ACK'
    5.M BLOCK SIZE (L)               5.S 'EXTERNAL CASSETTE ACK'
    6.M DATA CODE                    6.S 'EXTERNAL CASSETTE ACK'
      .
      .
    N.M LAST CODE                    N.S 'EXTERNAL CASSETTE ACK'
 ¥25: WRITE TRAIL TAPE
    1.M  ¥25                         1.S 'ACK'
    2.M BYTE COUNT (H)               2.S 'EXTERNAL CASSETTE ACK'
    3.M BYTE COUNT (L)               3.S 'EXTERNAL CASSETTE ACK'
```

```
¥26: SEARCH HEADER BLOCK
   1.M  ¥26                     1.S 'ACK'
   2.M BLOCK LEAD TAPE MODE     2.S 'EXTERNAL CASSETTE ACK'
   3.M BLOCK TRAIL MODE         3.S 'EXTERNAL CASSETTE ACK'
   4.M BLOCK SIZE (H)           4.S 'EXTERNAL CASSETTE ACK'
   5.M BLOCK SIZE (L)           5.S 'EXTERNAL CASSETTE ACK'
¥27: SEARCH EOF BLOCK
   1.M  ¥27                     1.S 'ACK'
   2.M BLOCK LEAD TAPE MODE     2.S 'EXTERNAL CASSETTE ACK'
   3.M BLOCK TRAIL MODE         3.S 'EXTERNAL CASSETTE ACK'
   4.M BLOCK SIZE (H)           4.S 'EXTERNAL CASSETTE ACK'
   5.M BLOCK SIZE (L)           5.S 'EXTERNAL CASSETTE ACK'
¥28: READ ONE BLOCK
   1.M  ¥28                     1.S 'ACK'
   2.M BLOCK LEAD TAPE MODE     2.S 'EXTERNAL CASSETTE ACK'
       (DUMMY)                                                  @
   3.M BLOCK TRAIL MODE         3.S 'EXTERNAL CASSETTE ACK'
     00:STOP AFTER READ (REMOTE OFF)                            @
     01:NONSTOP AFTER READ (REMOTE ON)                          @
   4.M BLOCK SIZE (H)           4.S 'EXTERNAL CASSETTE ACK'
   5.M BLOCK SIZE (L)           5.S 'EXTERNAL CASSETTE ACK'
¥29: READ START ONE CHARACTER READ MODE
   1.M  ¥29                     1.S 'ACK'
   2.M BLOCK LEAD TAPE MODE     2.S 'EXTERNAL CASSETTE ACK'
   3.M BLOCK TRAIL MODE         3.S 'EXTERNAL CASSETTE ACK'
   4.M MAX CHARACTER (H)        4.S 'EXTERNAL CASSETTE ACK'
   5.M MAX CHARACTER (L)        5.S 'EXTERNAL CASSETTE ACK'
¥2A: WRITE START ONE CHARACTER MODE
   1.M  ¥2A                     1.S 'ACK'
   2.M BLOCK LEAD TAPE MODE     2.S 'EXTERNAL CASSETTE ACK'
   3.M BLOCK TRAIL MODE         3.S 'EXTERNAL CASSETTE ACK'
   4.M MAX CHARACTER (H)        4.S 'EXTERNAL CASSETTE ACK'     @
   5.M MAX CHARACTER (L)        5.S 'EXTERNAL CASSETTE ACK'
¥2B: SET PULSE MODE (NORMAL/REVERSE)
   1.M  ¥2B                     1.S 'ACK'
   2.M 0(NORMAL) 4(REVERSE)     2.S 'EXTERNAL CASSETTE ACK'
```

```
3. SPEAKER COMMANDS
 ¥30: BEEP BY SCALE AND TIME
   1.M ¥30                      1.S 'ACK'
   2.M MUSICAL SCALE            1.S 'SPEAKER ACK' = ¥31
       SCALE  0:PAUSE
              1:DO   2:RE  3:MI 4:FA -----
       FROM 0 TO 57 (DECIMAL)
   3.M TIME (1=0.1 SEC)         1.S 'SPEAKER ACK'
 ¥31: BEEP BY FREQUENCY AND TIME
   1.M ¥31                      1.S 'ACK'
   2.M FREQUENCY (HIGH)         1.S 'SPEAKER ACK'
   3.M FREQUENCY (LOW)          1.S 'SPEAKER ACK'
      FREQUENCY: TIME OF 1/2 CYCLE (1=1.6 MICRO SEC)
   4.M TIME (HIGH)              1.S 'SPEAKER ACK'
   5.M TIME (LOW)               1.S 'SPEAKER ACK'
      TIME: 1=400 MICRO SEC
 ¥32: BEEP FOR KEY ACCEPT (BEEP 0.03 SEC)                           @
   1.M ¥32                      1.S 'ACK'
 ¥33: BEEP FOR CTRL/G (BEEP 1 SEC)                                  @
   1.M ¥33                      1.S 'ACK'
 ¥34: SET MELODY DATA
   1.M ¥34                      1.S 'ACK'
   2.M SCALE (SAME AS 'BEEP') 1.S 'SPEAKER ACK'
   3.M TIME  (SAME AS 'BEEP') 1.S 'SPEAKER ACK'
   4.M   SAME AS '2'
   5.M   SAME AS '3'
    .
    .
   2N.M    SAME AS '2'
   2N+1.M SAME AS '3'
   LAST. ¥FF                    1.S 'SPEAKER ACK'
     2N < 48
  NOTE. STORED MELEDY DATA WILL BE LOST WHEN MICRO PRINTER COMMAND  @
        ¥10 AND ¥ 11 IS EXECUTED. BECAUSE THESE COMMNADS USE SAME   @
        MEMORY BUFFER.                                              @
¥35: GO MELODY WHICH DATA IS SET BY COMMAND '¥34'
   1.M ¥35                      1.S 'ACK'
```

```
4. RS232C COMMANDS
        'ACK' CODE = ¥41
  ¥40: RS232C POWER ON (DRIVER ON)                                        ə
    1.M  ¥40                      1.S 'ACK'
  ¥41: RS232C OFF (DRIVER OFF)                                            ə
    1.M  ¥41                      1.S 'ACK'
  ¥42: RS232C MODE SET
    1.M  ¥42                      1.S 'ACK'
    2.M BIT RATE (H)              2.S 'RS232 ACK'                         ə
    3.M BIT RATE (L)              3.S 'RS232 ACK'                         ə
    4.M RS232C BIT LENGTH         4.S 'RS232 ACK'
    5.M RS232C MODE               5.S 'RS232 ACK'
      MODE:      7    6    5    4    3     2     1    0
                 <PARITY>  CTS  DSR  RTS   CD   <STOP BITS>
                 PARITY: 00.EVEN  01.ODD  10.NONE
                 CD (CARRIER DETECT)   0:CHECK   1:NO CHECK
                 DSR     0:CHECK     1:IGNORED                            ə
                 CTS     0:CHECK     1:IGNORED                            ə
                 STOP BITS  01:1  10:2  11:3                              ə
                 RTS   0:LOW  1:HIGH
  ¥43: READ RS232 STATUS REGISTER
    1.M  ¥43                      1.S  VALUE OF STATUS REGISTER
  ¥44: CLEAR STATUS REGISTER
    1.M  ¥44                      1.S  'ACK'
  ¥45: START READ (INTERRUPT MODE)
      AFTER ACCEPT THIS COMMAND, SLAVE CPU SENDS RS232C RECEIVED DATA ə
      TO MAIN CPU VIA SCI(SERIAL COMMUNICATION I/F), AND WHEN ACEEPT  ə
      ANOTHER COMMAND, THIS PROCESS WILL CANCEL.                          ə
    1.M  ¥45                      1.S  'ACK'
      ENTER RS232C RECEIVE MODE
      FROM SLAVE-CPU TO MAIN-CPU:: CONTINUOUS OF RECEIVED DATA
  ¥46: STOP READ
    1.M  ¥46                      1.S 'ACK'
; ¥47: START READ (READ ONE CHARACTER MODE) (DELETED)                     ə
;   1.M  ¥47                      1.S 'ACK'                               ə
;     IN READ ONE CHARACTER MODE, RECEIVED CHARACTERS ARE PUSHED INTO ə
;    THE STACK, WHEN ACCEPT READ CHARACTER COMMAND, ONE CHARACTER IS  ə
;    SEND TO MAIN-CPU.                                                    ə
  ¥48: SET GENERATING POLYNOMIAL
    1.M  ¥48                      1.S 'ACK'
    2.M  POLYNOMIAL (H)           2.S 'RS232 ACK' = ¥41
    3.M  POLYNOMIAL (L)           3.S 'RS232 ACK'
```

```
  ¥49: SET BCC REGISTER
    1.M  ¥48                           1.S 'ACK'
    2.M  VALUE (H)                     2.S 'RS232 ACK' = ¥41
    3.M  VALUE (L)                     3.S 'RS232 ACK'
  ¥4A: READ BCC REGISTER  (HIGH)
    1.M  ¥4A                           1.S VALUE OF BCC REGISTER (H)
  ¥4B: READ BCC REGISTER  (LOW)
    1.M  ¥4B                           1.S VALUE OF BCC REGISTER (L)
  ¥4C: SERIAL DRIVER ON
    1.M  ¥4C                           1.S 'ACK'
  ¥4D: SET RTS (REQUEST TO SEND)
    1.M  ¥4D                           1.S 'ACK'
    1.M  0 OR 1 (0:LOW 1:HIGH) 1.S 'RS232 ACK'
; ¥4E: READ RS232 BUFFER STATUS IN READ ONE CHARACTER MODE               ə
;   1.M  ¥4E                           1.S  STATUS                       ə
; ¥4F: READ ONE CHARACTER IN READ ONE CHARACTER MODE                     ə
;   1.M  ¥4F                           1.S RECEIVED CHARACTER            ə
;     WE CAN USE THE COMMAND AT BAUD FROM 110 TO 1200.                   ə

   NOTE. COMMAND ¥47, ¥4E AND ¥4F ARE DELETED (IN VERSION 1, IN          ə
         VERSION 2)                                                      ə

      STATUS REGISTER  (1:ERROR)                                         ə
        BIT 0: CARRIER DETECT ERROR
        PIT 1: PARITY ERROR
        BIT 2: OVER RUN ERROR
        BIT 3:
        BIT 7: RECEIVED CHARACTER

5. PLUG-IN OPTION COMMANDS                                               %
  ¥50: CHECK OPTION DEVICE                                               %
    1.M  ¥50                           1.S OPTIONS CODE                  %
                                         CODE  BIT2-BIT7 :0              %
                                         BIT0:VALUE OF P46               %
                                         BIT1:VALUE OF P20               ə
  ¥51: POWER ON ROM CASSETTE                                             %
    1.M  ¥51                           1.S 'ACK'                         %
  ¥52: POWER OFF ROM CASSETTE                                            %
    1.M  ¥52                           1.S 'ACK'                         %
```

```
6. MICRO CASSETTE COMMANDS
  ¥60: READY CHECK MICRO CASSETTE
   1.M  ¥60                    1.S ACK

  ¥61: SET MICRO CASSETTE PARAMETER
   1.M  ¥61                    1.S ACK
   2.M LONG PULSE BIAS (LOW PULSE) (HIGH BYTE)
                               2.S MICRO CASSETTE ACK (=¥61)
   3.M LONG PULSE BIAS (LOW PULSE) (LOW BYTE)
                               3.S MICRO CASSETTE ACK (=¥61)
   4.M LONG PULSE BIAS (HIGH PULSE) (HIGH BYTE)
                               4.S MICRO CASSETTE ACK (=¥61)
   5.M LONG PULSE BIAS (HIGH PULSE) (LOW BYTE)
                               5.S MICRO CASSETTE ACK (=¥61)
   6.M SHORT PULSE BIAS  (HIGH BYTE)
                               6.S MICRO CASSETTE ACK (=¥61)
   7.M SHORT PULSE BIAS  (LOW BYTE)
                               7.S MICRO CASSETTE ACK (=¥61)
   8.M BOUNDARY VALUE  (HIGH BYTE)
                               8.S MICRO CASSETTE ACK (=¥61)
   9.M BOUNDARY VALUE  (LOW BYTE)
                               9.S MICRO CASSETTE ACK (=¥61)

  ¥62  GAP BYTE LENGTH
   1.M ¥62                     1.S ACK
   2.M GAP BYTE COUNT (HIGH BYTE)
                               2.S MICRO CASSETTE ACK (=¥61)
   3.M GAP BYTE COUNT (LOW BYTE)
    1.M ¥23                     1.S 'ACK'

  ¥63  READ SKIP N BYTES
   1.M ¥63                     1.S ACK
   2.M SKIP BYTE COUNT (HIGH BYTE)
                               2.S MICRO CASSETTE ACK (=¥61)
   3.M SKIP BYTE COUNT (LOW BYTE)
                               3.S MICRO CASSETTE ACK (=¥61)

 ¥64: WRITE ONE BLOCK BY EPSON FORMAT
    1.M ¥64                     1.S 'ACK'
    2.M BLOCK LEAD TAPE MODE    2.S 'MICRO CASSETTE ACK'
        00:WITH LONG GAP (DEPEND OF VALUE OF 'GAP BYTES',              @
                             DEFAULT VALUE = 125 BYTES)                @
        01:WITH SHORT GAP (15 BYTES ¥FF)                               @
        FF:WITH LEAD TAPE (625 BYTES ¥FF)                              @
    3.M BLOCK TRAIL MODE        3.S 'MICRO CASSETTE ACK'
        00:WITH LONG GAP (DEPEND OF VALUE OF 'GAP BYTES',              @
                             DEFAULT VALUE = 125 BYTES) AND STOP       @
        01:WITH SHORT GAP (15 BYTES ¥FF) AND NONSTOP                   @
        FF:WITH TRAIL TAPE (625 BYTES ¥FF) AND STOP                    @
    4.M BLOCK SIZE (H)          4.S 'MICRO CASSETTE ACK'
    5.M BLOCK SIZE (L)          5.S 'MICRO CASSETTE ACK'
    6.M DATA CODE               6.S 'MICRO CASSETTE ACK'
     .
     .
    N.M LAST CODE               N.S 'EXTERNAL CASSETTE ACK'

 ¥65: WRITE TRAIL TAPE
    1.M  ¥65                    1.S 'ACK'
    2.M BYTE COUNT (H)          2.S 'MICRO CASSETTE ACK'
    3.M BYTE COUNT (L)          3.S 'MICRO CASSETTE ACK'
```

```
¥66: SEARCH HEADER BLOCK
   1.M  ¥66                    1.S 'ACK'
   2.M BLOCK LEAD TAPE MODE    2.S 'MICRO CASSETTE ACK'
       (DUMMY)                                                @
   3.M BLOCK TRAIL MODE        3.S 'MICRO CASSETTE ACK'
       00:STOP AFTER READ                                     @
       01:NONSTOP AFTER READ                                  @
   4.M BLOCK SIZE (H)          4.S 'MICRO CASSETTE ACK'
   5.M BLOCK SIZE (L)          5.S 'MICRO CASSETTE ACK'

¥67: SEARCH EOF BLOCK
   1.M  ¥67                    1.S 'ACK'
   2.M BLOCK LEAD TAPE MODE    2.S 'MICRO CASSETTE ACK'
       (DUMMY)                                                @
   3.M BLOCK TRAIL MODE        3.S 'MICRO CASSETTE ACK'
       00:STOP AFTER READ                                     @
       01:NONSTOP AFTER READ                                  @
   4.M BLOCK SIZE (H)          4.S 'MICRO CASSETTE ACK'
   5.M BLOCK SIZE (L)          5.S 'MICRO CASSETTE ACK'

¥68: READ ONE BLOCK
   1.M  ¥68                    1.S 'ACK'
   2.M BLOCK LEAD TAPE MODE    2.S 'MICRO CASSETTE ACK'
       (DUMMY)                                                @
   3.M BLOCK TRAIL MODE        3.S 'MICRO CASSETTE ACK'
       00:STOP AFTER READ                                     @
       01:NON STOP AFTER READ                                 @
   4.M BLOCK SIZE (H)          4.S 'MICRO CASSETTE ACK'
   5.M BLOCK SIZE (L)          5.S 'MICRO CASSETTE ACK'

¥69: READ START ONE CHARACTER READ MODE
   1.M  ¥69                    1.S 'ACK'
   2.M BLOCK LEAD TAPE MODE    2.S 'MICRO CASSETTE ACK'
       (DUMMY)                                                @
   3.M BLOCK TRAIL MODE        3.S 'MICRO CASSETTE ACK'
       00:STOP AFTER READ                                     @
       01:NON STOP AFTER READ                                 @
   4.M MAX CHARACTER (H)       4.S 'MICRO CASSETTE ACK'
   5.M MAX CHARACTER (L)       5.S 'MICRO CASSETTE ACK'

¥6A: WRITE START ONE CHARACTER MODE
   1.M  ¥6A                    1.S 'ACK'
   2.M BLOCK LEAD TAPE MODE    2.S 'MICRO CASSETTE ACK'
   3.M BLOCK TRAIL MODE        3.S 'MICRO CASSETTE ACK'
   4.M MAX CHARACTER (M)       4.S 'MICRO CASSETTE ACK'
   5.M MAX CHARACTER (L)       5.S 'MICRO CASSETTE ACK'

¥6B: STOP EPSON FORMAT WRITE
     (MICRO CASSETTE POWER OFF)                               @
   1.M  ¥6B                    1.S 'ACK'

¥6C: STOP READ ONE CHARACTER MODE
   1.M  ¥6C                    1.S 'ACK'

¥6D: SET COUNTER VALUE
   1.M  ¥6D                    1.S 'ACK'
   2.M COUNTER VALUE (H)       2.S 'MICRO CASSETTE ACK'
   3.M COUNTER VALUE (L)       3.S 'MICRO CASSETTE ACK'

¥6E: READ COUNTR VALUE (H)
   1.M  ¥6E                    1.S COUNTER VALUE (H)

¥6F: READ COUNTER VALUE (L)
   1.M  ¥6F                    1.S COUNTER VALUE (L)
```

```
¥70: CHECK IF WRITE ENABLE
    1.M  ¥70                        1.S 0:ENABLE   ¥FF:DISABLE

¥71: REWIND N COUNT
    1.M  ¥71                        1.S 'MICRO CASSETTE ACK'
    2.M COUNT VALUE (H)             2.S 'MICRO CASSETTE ACK'
    3.M COUNT VALUE (L)             3.S 'MICRO CASSETTE ACK'

¥72: FEED N COUNT
    1.M  ¥72                        1.S 'ACK'
    2.M COUNT VALUE (H)             2.S 'MICRO CASSETTE ACK'
    3.M COUNT VALUE (L)             3.S 'MICRO CASSETTE ACK'

¥73: REWIND TO TOP OF FILE
    1.M  ¥73                        1.S 'ACK'

¥74: READ VALUE OF STATUS REGISTER
    1.M  ¥74                        1.S VALUE OF STATUS REGISTER

¥75: CLAER STATUS REGISTER
    1.M  ¥75                        1.S 'ACK'

¥76: HEAD LOAD
    1.M  ¥76                        1.S 'ACK'

¥77: HEAD UNLOAD
    1.M  ¥77                        1.S 'ACK'

¥78: GO REWIND
    1.M  ¥78                        1.S 'ACK'

¥79: GO FAST FEED
    1.M  ¥79                        1.S 'ACK'

¥7A: GO SLOW FEED
    1.M  ¥7A                        1.S 'ACK'

¥7B: STOP (REWIND, FAST FORWARD, SLOW FORWARD)
    1.M  ¥7B                        1.S 'ACK'

¥7C: MICRO CASSETTE POWER ON AND READ COUNTER PULSE
    1.M  ¥7C                        1.S COUNTER PULSE STATUS
                                        (COUNTER POSITION  ¥00 OR ¥80)

¥7D: MICRO CASSETTE POWER OFF
    1.M  ¥7D                        1.S 'ACK'
```

```
7. PORT COMMAND

¥80: CONNECT SLAVE CPU'S PORT XXXX TO MAIN CPU'S PORT P12 (SFLAG).
      AFTER ACCEPTED THIS COMMAND, VALUE OF DESTINATED ADDRESS OF
     SLAVE CPU IS STORED TO P34 (CONNECTED TO MAIN CPU'S P12)
     CONTINUOUSLY, THEN MAIN CPU CAN SEE SLAVE PORT'S DATA TO SEE
     P12. (SLAVE P34 IS THE OUT PORT, MAIN P12 THE IN PORT)

     1.M  ¥80                         1.S  'ACK'
     2.M  PORT ADDRESS (H)            2.S  'ACK'
     3.M  PORT ADDRESS (L)            3.S  'ACK'
     4.M  PORT BIT                    4.S  'ACK'
           (TARGET BIT = 1)
¥81: CONNECT 'PLUG-1' TO PORT XX
      AFTER ACCEPTED THIS COMMAND, VALUE OF P40 OF SLAVE CPU IS
     STORED TO THE DESTINATED PORT OF SLAVE CPU CONTINUOUSLY.
     P267 (ADDRESS ¥26 BIT 7, OUT PORT) OF MAIN CPU IS CONNECTED TO
     THE P40 (IN PORT) OF SLAVE CPU.

     1.M  ¥81                         1.S  'ACK'
     2.M  PORT ADDRESS (H)            2.S  'ACK'
     3.M  PORT ADDRESS (L)            3.S  'ACK'
     4.M  PORT BIT                    4.S  'ACK'
           (TARGET BIT = 1)

;¥82: CONNECT 'SEND' TO PORT XXX (DELETED)
;    1.M  ¥82                         1.S  'ACK'
;    2.M  PORT ADDRESS (H)            2.S  'ACK'
;    3.M  PORT ADDRESS (L)            3.S  'ACK'
;    4.M  PORT BIT                    4.S  'ACK'
;          (TARGET BIT = 1)
;
;¥83: CONNECT 'PLUG-1' AND 'SEND' TO PORT XXX (DELETED)
;    1.M  ¥83                         1.S  'ACK'
;    2.M  PORT ADDRESS (H) (PL) 2.S  'ACK'
;    3.M  PORT ADDRESS (L) (PL) 3.S  'ACK'
;    4.M  PORT BIT         (PL) 4.S  'ACK'
;          (TARGET BIT = 1)
;    5.M  PORT ADDRESS (H) (SE) 5.S  'ACK'
;    6.M  PORT ADDRESS (L) (SE) 6.S  'ACK'
;    7.M  PORT BIT         (SE) 7.S  'ACK'
           (TARGET BIT = 1)
NOTE. COMMAND ¥82 AND ¥83 ARE ERASED. (IN VERSION 1, IN VERSION 2)
NOTE. AFTER THESE COMMAND, VALUE OF 'PLUG-1' OR 'SEND' DATA IS
      STORED TO DESTINATION ADDRESS. IN THESE MODE, THE COMMAND WILL
      WORK TILL RECEIVE NEXT COMMAND.


BREAK COMMAND
1. SEND BREAK COMMAND
2. SEND BREAK COMMAND, AND CAUSE TO SLAVE CPU OVERRUN FLAMMING ERROR.
```

```
  *****      *****    *********             *********  *      **  ********
 *******    *******   *********             *********  **     **  *********
**     **  **     **  **    **              **         ***    **  **     **
**     **  **     **       **               **         ****   **  **     **
**     **  **     **      **      *******   ******     ** **  **  **     **
**     **  **     **     **       *******   ******     **  ** **  **     **
**     **  **     **    **                  **         **   ****  **     **
**     **  **     **    **                  **         **    ***  **     **
 *******    *******     **                  *********  **     **  *********
  *****      *****      **                  *********  **      *  ********
```

PRINT COMPLETED AT 14:03:23 FOR USER: CMS1   DIST: USER01

```
*********  *      **  ********                *****      *****    *********
*********  **     **  *********              *******    *******   *********
**         ***    **  **     **             **     **  **     **  **    **
**         ****   **  **     **             **     **  **     **       **
******     ** **  **  **     **   *******   **     **  **     **      **
******     **  ** **  **     **   *******   **     **  **     **     **
**         **   ****  **     **             **     **  **     **    **
**         **    ***  **     **             **     **  **     **    **
*********  **     **  *********              *******    *******     **
*********  **      *  ********                *****      *****      **
```