TRS - 80 HACKER'S HANDBOOK FOR NEWDOS/80

BY KEVIN O'HARE

TRADEMARK ACKNOWLEDGEMENTS :-

TRS-80 is the Trademark of the Tandy Corp.
NEWDOS/80 is the Trademark of Apparat Inc.
LDOS is the Trademark of Logical Systems Corp.
DOSPLUS is the Trademark of Micro-Systems Software Inc.

All trademarks, trade names and copyrights not listed above are unknown to the Author.

First Edition
First Printing August 1984

Published by
COMPUTEREASE
55 Elizabeth Street,
Brisbane 4000
AUSTRALIA

Printed by
Graphic Service Co.
21 Gipps Street,
Fortitude Valley
Brisbane 4006

FOREWORD

This is an explanation of the different parts of Memory in the TRS-80. There is a wealth of information about the Home Microcomputer that has been documented, particularly about the Tandy Models. The author has tried to collate this from various sources and construct an easy reference guide to the most useful aspects for the programmer.

The DOS Addresses are those for NEWDOS/80 Version 2. There are instances of addresses given which might vary according to the date of the DOS purchase but these are exceptional and should not worry most users. However, the overall usefulness of this information will not be impaired by these occurrences. The old ROM was used for the Model I as the quantity sold gives it predominance over the later version.

Where the text refers to Model I only, the same information applies to the computers PMC80, Video Genie and System 80. Where there is no mention of Model I or Model III, the code or address is common to all the computers.

There is a presumption because of the conciseness of the text that the user has some knowledge of the facets of information given. There are over 10 Books published by the IJG Company which explain nearly all functions and much more in great and lucid detail.

This work started originally as a Memory Reference but commonly sought information about other aspects of the computer were eventually included to give the book a wider application.

I must express my indebtedness to Len Browne and Alf West, without whose encouragement and assistance this work would never have eventuated. Thank you Len and Alf, and may life be kind to you.

C O N T E N T S

CONTENTS

CONTENTS BY MEMORY

MODEL I

IN LEVEL II :-

UNDER NEWDOS/80 :-

CONTENTS BY MEMORY

MODEL III

IN LEVEL II :-

UNDER NEWDOS/80 :-

PART 1

THE MEMORY REFERENCE

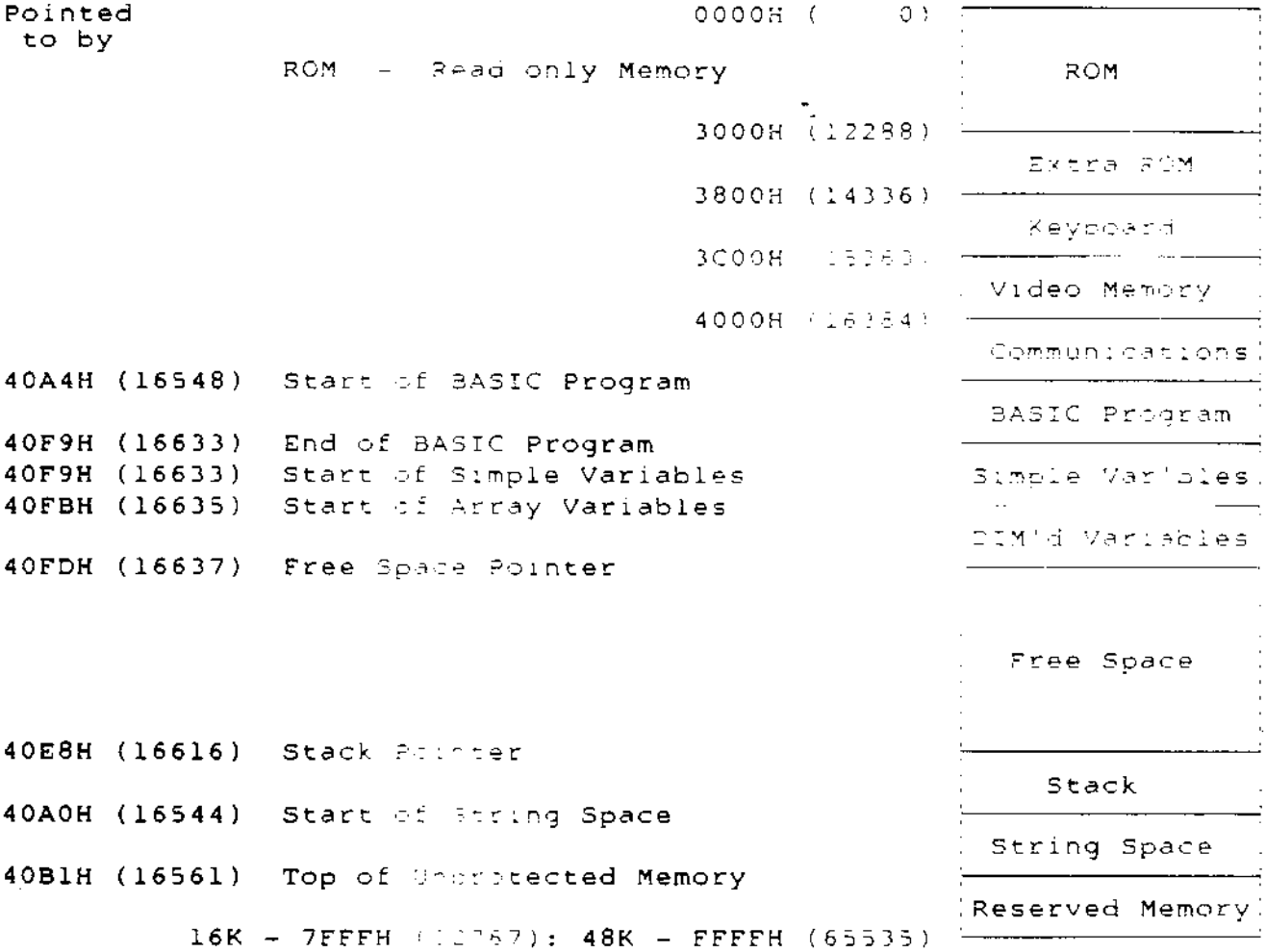
CHAPTER 1

THE MEMORY REFERENCE

ROM

MACROVIEW

The lay-out of Memory is obviously varied but some idea of the areas used is given by the chart below. The Basic pointers on the left contain the addresses whether it be Level II Basic or Disk BASIC.



R O M

0000H - 2FFFH = 0 - 12287 :- 12288 Bytes

This Area contains the READ ONLY Memory (ROM).
The following is an attempt to classify areas :-

0000H to 0700H (1800) :- Device Handling
0700H to 1600H (5600) :- Mathematics
1600H to 1A00H (6700) :- Tables and References
1A00H to 1C00H (7100) :- Monitoring
1C00H to 2B00H (11000) :- BASIC Interpreter
2B00H to 2FFFH (12000) :- Utilities

MODEL I ONLY

3000H - 37DBH = 12288 to 14299 :- 2012 Bytes

The Model I Computer has no memory for these addresses.
This area has been set aside by the makers of the TRS-80 for use by devices which need direct Memory access. e.g.

3000H - 377FH = 12288 to 14207 :- 1920 Bytes.
This area is used by the EXATRON STRINGY FLOPPY for it's Utilities, or the extra ROM for some models of PMC-80 Video Genie and System 80. As with the ROM any memory installed here is generally READ ONLY Memory, i.e. it cannot be changed by the user.

37DCH - 37FFH = 14300 to 14335 :- 36 Bytes

This area is used by the INPUT/OUTPUT Devices and can be classed as Communication Addresses.

37DCH - 37DFH = 14300 to 14303 :- DOS Status Addresses
37E0H = 14304 :- Interrupt Latch Address
37E1H = 14305 :- DOS - Drive Select & Drive 0 Latch Address
37E2H = 14306 :- Cassette Latch Address
37E3H = 14307 :- DOS - Drive 1 Latch Address
37E4H = 14308 :- Cassette Select Address
37E5H = 14309 :- DOS - Drive 2 Latch Address
37E7H = 14311 :- DOS - Drive 3 Latch Address
37E8H = 14312 :- Line Printer
37EBH = 14315 :- Addresses.
37ECH = 14316 :- DOS - Command/Status Register
37EDH = 14317 :- DOS - Track Register
37EEH = 14318 :- DOS - Sector Register
37EFH = 14319 :- DOS - Data Register

MODEL III - EXTRA ROM ROUTINES

3000H - 37FFH = 12288 to 14335 :- 2048 Bytes

This area in the Model III is taken up by extra Routines generally handling Cassette and RS-232C functions.

Addr.	Contents	Routine	Function
3000H	C3 5E 32	325EH	Cass - 500 Baud - Write leader &sync byte
3003H	C3 9B 32	329BH	Cass - 1500 Baud - Write leader &sync byte
3006H	C3 74 32	3274H	Cass - 500 Baud - Read leader till sync
3009H	C3 DA 32	32DAH	Cass - 1500 Baud - Read leader till sync
300CH	C3 C0 31	31C0H	Cass - Turn Cassette off
300FH	C3 D1 31	31D1H	Cass - Turn Cassette on & wait full speed
3012H	C3 AB 34	34ABH	Disk - Test if Drive ready, then Reset
3015H	C3 55 34	3455H	Initialisation - (IPL) Reset entry
3018H	C3 C2 35	35C2H	Interrupt - Service Routine
301BH	C3 FB 35	35FBH	RS-232C - Initialise
301EH	C3 5A 36	365AH	RS-232C - Input Routine
3021H	C3 80 36	3680H	RS-232C - Output Routine
3024H	C3 8E 33	338EH	Keyboard - Input Routine
3027H	C3 39 37	3739H	I/O - Change Device Routing
302AH	C3 F7 31	31F7H	BASIC - "#" processing (e.g.Print#1)
302DH	C3 7B 37	377BH	BASIC line print processing
3030H	C3 99 37	3799H	BASIC line print processing
3033H	C3 BB 35	35BBH	Get Today's Date
3036H	C3 A0 35	35A0H	Get current Time
3039H	DB E4 CB 6F C3 1C 35		- Reset processing (is Reset pressed ?)
3040H	18 D3 =	JR 3015H	- Reset processing
3042H	C3 B5 37	37B5H	Cassette - Set Baud rate
3045H - 31A4H	7 lots of ASCII codes (some code between each)		
31A5H - 37FFH	Extra Model III routines		

The extra ROM involved in the listing of BASIC program lines allows the listing of the extra Model III symbols and Characters.
The PRINT# extra ROM allows different uses for the '#' symbol.

KEYBOARD MEMORY

3800H - 3BFFH = 14336 to 15359 :- 1024 Bytes

The Keyboard uses a buffer (see Page 18) at 4036H to 403CH in both Mods I & III. Each Row on the Keyboard Diagram uses a Byte of this Buffer for storage excepting the last Row (the shift key row). In the Diagram the location of the Buffer for each Row is listed on the right and the address of each Row on the left.

The Keyboard is Memory mapped and uses this area as follows :-

BITS	7	6	5	4	3	2	1	0	BUFFER
1 3801H 14337	G g	F f	E e	D d	C c	B b	A a	@ `	4036H 16438
2 3802H 14338	O o	N n	M m	L l	K k	J j	I i	H h	4037H 16439
3 3804H 14340	W w	V v	U u	T t	S s	R r	Q q	P p	4038H 16440
4 3808H 14344						Z z	Y y	X x	4039H 16441
5 3810H 14352	7 '	6 &	5 %	4 $	3 #	2 "	1 !	0	403AH 16442
6 3820H 14368	/ ?	. >	- =	, <	; +	: *	9)	8 (	403BH 16443
7 3840H 14390	Space	RghtA	LftA	DownA	UpA	Brk	Cls	Enter	403CH 16444
8 3880H 14454							*Rght Shift	Left Shift	
VALUE	80H=128	40H=64	20H=32	10H=16	8H=8	4H=4	2H=2	1H=1	

*Model III only

The pattern in 3800H to 38FFH is echoed in each of the other 256 Byte areas making up the keyboard memory i.e. 3900H to 39FFH, 3A00H to 3AFFH, and 3B00H to 3BFFH.

The CPU scans the Keyboard Memory. When a key is pressed the value thus created is XOR'ed with the Row Buffer location so that on the first scan the Buffer receives the value of the key pressed. On the second scan if the key is still pressed, it is rendered ineffective since it is XOR'ed with it's own value.

KEYBOARD MEMORY

The ROM reads each row of the keyboard memory. To obtain the address of the next row in the Keyboard map the LSB of the current address is shifted left (doubled). The Bytes in the area from 3800H - 38FFH are affected in the following way :-

1st row - Keys @ABCDEFG Values 40H(64) to 47H(71)
Shifted `abcdefg 60H(96) to 67H(103)

In the top 1 of every 2 bytes, the bit is set corresponding to the column value in the bottom line of the table on the previous page. i.e. the bit is set in 3801H,3803H,3805H,3807H,3809H, etc right up to 3BFFH.

2nd row - Keys HIJKLMNO Values 48H(72) to 4FH(79)
Shifted hijklmno 68H(104) to 6FH(111)

In this case the bit of the top 2 of every 4 bytes is set according to the column value. i.e. the bit is set in 3802H,3803H,3806H,3807H,etc.

3rd row - Keys PQRSTUVW Values 50H(80) to 57H(87)
Shifted pqrstuvw 70H(112) to 77H(119)

The top 4 of every 8 Bytes for this row are set. Notice the number of bytes affected is the Bit value of the row number. (3804H-3807H,etc).

4th row - Keys XYZ Values 58H(88) to 5AH(90)
Shifted xyz 78H(120) to 7AH(122)

The top 8 of every 16 bytes is affected. There are no keys for values 5BH(91) to 5FH(95) as the four arrow symbols that these values give with PRINT CHR$ are used for control codes. i.e. 3808H-380FH,etc.
N.B. Some models of the PMC-80, Video Genie and System 80 use 4 of these spaces as special function keys.

5th row - Keys 01234567 Values 30H(48) to 37H(55)
Shifted !"#$%&' 20H(32) to 27H(39)

In this case the bit of the top 16 of every 32 bytes is set according to the column value. i.e. 3810H-381FH,3830H-383FH,etc.

6th row - Keys 89:;<=>? Values 38H(56) to 3FH(63)
Keys ()*+,-./ 28H(40) to 2FH(47)

In this case the bit of the top 32 of every 64 bytes is set according to the column value. i.e. 3820H-383FH,3840H-385FH,etc.

7th row - Enter = (0DH) CLS = (1FH) Break = (01H)
Up Arrow =(5BH) Down Arrow =(0AH) Left Arrow =(08H)
Right Arrow = (09H) Space Bar = (20H)

In the top 64 of every 128 Bytes the Bit is set corresponding to the Bit Column on the Keyboard Diagram. i.e. 3840H-387FH,38C0H-38FFH,etc.

8th row - The only value here is the 01H of the Shift Key
In the top 128 of every 256 Bytes the Value 01H is OR'ed. (3880H-38FFH)

VIDEO MEMORY

3C00H - 3FFFH = 15360 to 16383 :- 1024 Bytes

This is the Memory that the Screen uses.

The Video is also Memory mapped and is available to the Programmer through the PRINT, PRINT@, POKE&PEEK, and other Functions.

Each line on the Screen is 64 Bytes long and there are 16 Lines. Total Bytes used on the Screen is 16*64=1024, 1 byte per screen position.

A particular position on the Screen can be accessed using PRINT@ by (Line No.-1) * 64 + TAB(POS).

This means that by POKEing (15360+(Line No.-1)*64+TAB),X or PRINT CHR$(X) the Screen can be filled with :-

A Control Code	Value	00H - 1FH	(0 to 31)
A Character	Value	20H - 7FH	(32 to 127)
A Graphics Symbol	Value	80H - BFH	(128 to 191)
A Space Compression Code	Value	C0H - FFH	(192 to 255)

In actual fact the only way to place a Byte in the bottom right hand corner of the screen is by POKEing as PRINTing causes an automatic line feed and the top line of the screen is overwritten by the second line.

A very good way to save memory for a heading or a Graphics design or a fancy Screen format is to make a String with as many Bytes as you may need. You can construct a String with any values you like from 0 to 255 from a Monitor or else from BASIC using :-

```
A = PEEK (VARPTR(STRING)+1) + PEEK (VARPTR(STRING)+2)*256
then  POKE A + BYTE NO.-1, X
```

An example of this is when you want to have a Graphics symbol in a Screen Diagram. The BASIC line is entered with a "*" in the place where a "CHR$(170)" was wanted. Then from DEBUG in the DOS or from BASIC by PEEKing and POKEing, the Value can be changed (2AH(42) to AAH = CHR$(170)) to give the result you want. When listed the Line shows "KILL" in that spot. See ASCII codes for all reserved word values.

The Space Compression Codes can be very usefully employed in this format to save code when X spaces are needed.

Note :- The Space Compression Codes do not mean PRINT TAB(X) but PRINT X Spaces where X is the number (0-63) of the value above C0H (192), (C0H(192)-FFH(255)).

CHAPTER 2

THE MEMORY REFERENCE

RAM

THE COMMUNICATIONS REGION

In this Chapter we look at the first part of the Communications Region and examine the following areas :-

RESTARTS

In the TRS-80 there are eight of these RESTARTS. When a Restart instruction is given, the Program Counter is set to the location in ROM formed by prefixing "00" to the RST value. For example RST 08H jumps to 0008H or RST 30H to 0030H. At each of these addresses except 0000H, there is a Jump to the Communications Area. The contents at the Target addresses in the Communications Area differ according to whether the Computer was initialised by Level II or Disk.

RST	Code	Addr.	Restart Address Contains	Jumps at Target Address Mod I Lev2	Mod I DOS	Mod III Lev2	Mod III DOS
RST 00H	C7H	0000H	Start of ROM = IPL				
RST 08H	CFH	0008H	JP 4000H	1C96H	1C96H	1C96H	1C96H
RST 10H	D7H	0010H	JP 4003H	1D78H	1D78H	1D78H	1D78H
RST 18H	DFH	0018H	JP 4006H	1C90H	1C90H	1C90H	1C90H
RST 20H	E7H	0020H	JP 4009H	25D9H	25D9H	25D9H	25D9H
RST 28H	EFH	0028H	JP 400CH	Return	4BC2H	Return	4B67H
RST 30H	F7H	0030H	JP 400FH	Return	4609H	Return	45D4H
RST 38H	FFH	0038H	JP 4012H	EI+Ret	45F2H	3018H	3018H

RESTARTS

RST 00H Code C7H The RESET Restart

The first RST, has the code C7H and sends the Computer to Address 0000H i.e. it resets or reboots.

::::::::::::::::::::

RST 08H Code CFH The Compare Byte Restart

0008H contains Jump to 4000H. - 4000H contains :-
Both Level II & DOS - Mods I & III : JP 1C96H (C3 96 1C)

At 1C96H is the ROM routine that compares the byte pointed to by the HL Register with the Byte in Memory immediately following the RST 08H call.

This Restart should only be used when Basic has been initialised since if the bytes do not match the Syntax error code will be accessed and, with Basic not initialised, the necessary communications region pointers will not have been initialised to correct values.

The return address on the stack is incremented past the compare byte and if the bytes match, control passes to the RST 10H code to access the next valid character before returning to the caller.

A typical use of this Restart is to test for the correct syntax within a Basic program statement. As an example the POKE statement requires a comma between the address and the value and the RST 10H is used to ascertain if this separator is present.

The code at 1C96H is as follows :-

```
1C96H  7E        LD    A,(HL)      ; Get value to be compared
1C97H  E3        EX    (SP),HL     ; Save return Address
1C98H  BE        CP    (HL)        ; Compare with Byte following
1C99H  23        INC   HL          ; Bump return address
1C9AH  E3        EX    (SP),HL     ; Restore return Address
1C9BH  CA 78 10  JP    Z,1078H     ; Go to RST 10 if correct
1C9EH  C3 97 19  JP    1997H       ; Signal Error if incorrect
```

RESTARTS

RST 10H Code D7H The numeric test Restart

0010H contains Jump to 4003H - 4003H contains
Both Level II & DOS - Mods I & III : JP 1D78H (C3 78 1D)

The purpose of the RST 10H code is to extract the next character from the code pointed to by the HL Register pair. The HL Register pair is used by the interpreter to point to the current code being executed under Basic. The RST 10H code is used extensively by Basic to test for end of statement and whether or not the next valid character is numeric.

This Address is changed by ESF initialisation to 3082H (C3 82 30). This is the way the ESF intercepts Basic Commands to execute the "@" Functions. The routine at 3082H interprets and passes control to the code involved.

The valid character extraction is accomplished by INCrementing HL, skipping irrevelant characters (Spaces, 09H or 0AH) and returning with the next character in the 'A' Register. The Zero flag is set if the character is a 00H or 3AH (:) both characters signifying end of statement. If the character is numeric the Carry flag is set.

A RST 10H is performed prior to Basic interpreting any Basic state- ment whether from a Basic program or the Command mode. This is the reason that, for the most part, the interpreter will ignore any spaces or CHR$ (10) (line feeds) it encounters in a statement.

The code at 1D78H is as follows :-

```
1D78H  23        INC   HL          ; Bump to next character
1D79H  7E        LD    A,(HL)      ; Get next character
1D7AH  FE 3A     CP    3AH         ; Compare it with colon
1D7CH  D0        RET   NC          ; Return if >=3A (alphabetic)
1D7DH  FE 20     CP    20H         ; Else test for a space
1D7FH  CA 78 1D  JP    Z,1D78H     ; If so get next character
1D82H  FE 08     CP    08H         ; Compare with vertical Tab
1D84H  30 05     JR    NC,1D8BH    ; Go if 0B-39 control, numeric
1D86H  FE 09     CP    09H         ; Test for horizontal Tab or LF
1D88H  D2 78 1D  JP    NC,1D78H    ; If not get next character
1D8BH  FE 30     CP    30H         ; Compare with ASCII "0"
1D8DH  3F        CCF               ; Set Carry Flag if numeric
1D8EH  3C        INC   A           ; Check for zero byte so as
1D8FH  3D        DEC   A           ;  not to affect Carry flag
1D90H  C9        RET               ; Return to caller
```

RESTARTS

RST 18H Code DFH The Compare HL/DE Restart

0018H contains Jump to 4006H. - 4006H contains :-
Both Level II & DOS - Mods I & III : JP 1C90H (C3 90 1C)

This again is a compare, but in this case it is a comparison of the contents of the HL and DE Registers and the result is Signed by both the CARRY and ZERO Flags as follows :-

HL < DE	: C	:	CARRY FLAG SET IF LESS
HL > DE	: NC	:	NO CARRY IF MORE
HL <> DE	: NZ	:	ZERO FLAG RESET IF UNEQUAL
HL = DE	: Z	:	ZERO FLAG SET IF EQUAL

The ROM code at 1C90H is as follows :-

```
1C90H  7C      LD      A,H      ; Get Value of H in A
1C91H  92      SUB     D        ; Subtract D
1C92H  C0      RET     NZ       ; Go back if unequal
1C93H  7D      LD      A,L      ; Get Value of L in A
1C94H  93      SUB     E        ; Subtract E
1C95H  C9      RET              ; Return
```

This Code will not work for signed Integers (unless positive).
It also destroys the previous content of "A".

An example of the use of RST 18H, it is assumed that the numeric input must be within pre-determined maximum and minimum values. In this example, we use the ROM call to 1E5AH to convert the input from ASCII to Binary - the result being returned in the DE Register pair.

```
8000H  21 xx XX  LD      HL,Input  ; After Input
8003H  2B        DEC     HL        ; Decrement for RST 10H
8004H  D7        RST     10H       ;  to ensure it is numeric
8005H  D2 yy YY  JP      NC,Error  ; If not then error message
8008H  CD 5A 1E  CALL    1E5AH     ; Get binary of Input
800BH  21 mm MM  LD      HL,Max    ; Upper limit value
800EH  DF        RST     18H       ; Compare with Input value
800FH  DA yy YY  JP      C,Error   ; Carry if > Maximum
8013H  21 nn NN  LD      HL,Min    ; Lower limit value
8016H  DF        RST     18H       ; Compare with Input value
8017H  D2 yy YY  JP      NC,Error  ; No Carry if < Minimum
801AH  ....... Process value in DE Register from 1E5AH Call
```

RESTARTS

RST 20H Code E7H The Data Type Test Restart

0020H contains Jump to 4009H - 4009H contains
Both Level II & DOS - Mods I & III : JP 25D9H (C3 D9 25)

This tests the Data Type Flag (40AFH) and returns a value in the "A" Register as well as the Flags as follows :-

40AFH Value	Flags Set	Number Returned in "A"
02 = INTEGER	NZ . C . M . E	-1
03 = STRING	Z . C . P . E	0
04 = SINGLE	NZ . C . P . O	1
08 = DOUBLE	NZ . NC . P . E	5

N.B.- The Programmer changing the Number-Type Flag or the Accumulator creates the possibility of them being out of phase.
The code at 25D9H is as follows :-

```
25D9H  3A AF 40   LD      A,(40AFH)   ; Load data type flag
25DCH  FE 08      CP      08H         ; Is it double precision
25DEH  30 05      JP      NC,25E5H    ; If so then jump
25E0H  D6 03      SUB     03H         ; Not double precision
25E2H  B7         OR      A           ;  then reset status flags
25E3H  37         SCF                 ;   and exit with
25E4H  C9         RET                 ;    Carry flag set.
25E5H  D6 03      SUB     03H         ; For double precision reset
25E7H  B7         OR      A           ;   status flags & exit with
25E8H  C9         RET                 ;    Carry flag reset.
```

As an example of the use of RST 20H, assume we test the result of the addition of two Integer values to see if the result is still an Integer (If overflow occurs it would be single precision.) The example uses the ROM call 0B2DH with the result in both the Accumulator and HL.

```
8000H  3E 02          LD      A,2             ; Integer Code
8002H  32 AF 40       LD      (40AFH),A       ; Set Type Flag
8005H  ED 4B 00 90    LD      BC,(VALUE1)     ; First Value
8009H  2A 02 90       LD      HL,(VALUE2)     ; Second Value
800CH  CD 20 0B       CALL    0B20H           ; Integer Add
800FH  E7             RST     20H             ; Overflow ?
8010H  F2 yy YY       JP      P,TOO LARGE     ; Not an Integer
8013H  22 04 90       LD      (RESULT),HL     ; Save Result
8016H  ..............  Process the stored result
```

RESTARTS

RST 28H Code EFH The DOS Function Call Restart

0028H contains Jump to 400CH.
400CH contains Level II : Return (C9)
DOS Model I : JP 4BC2H (C3 C2 4B)
DOS Mod III : JP 4367H (C3 67 43)

In Level II there is no action here. But in the DOS, this Address is used by more than 30 Sequences. Thus it is called the DOS Function Call Restart. Action taken depends on the value in Register "A".

The major use by the DOS of the RST 28H routine is in loading System modules to perform DOS or Basic /SYS functions. The /SYS file Directory entries all have fixed positions relative to the Directory. The /SYS module is not accessed through the HIT (hashed name storage) sector but is found using the contents of the 'A' register, the lower 5 bits of which contains the relative sector (-2) in the Directory and the position in that sector. Thus the name of the /SYS file is not hashed for a match in the HIT sector as user files are, but the Directory entry is accessed directly. The top 3 bits of 'A' are used to differentiate between sub-functions. Sometimes other registers contain parameters for use by the module.

The RST28H code is one of the most practical and thus the most often used function of NEWDOS/80. Some of the uses of an RST 28H are loading a /SYS module or a program file, executing a DOS command, open a file, close a file, kill a file, create a Directory entry, and many others, even insert a user interrupt routine into the Interrupt code. All these functions require parameters to be set for use by the routine. eg: HL pointing to ASCII storage of file name, DE pointing to the FCB, etc.

The first two Bytes at 4BC2H (or 4367H in Mod III) are [illegible] return remove the previous Stack contents (the only RST 28H at all). After processing, control is returned to the previous Address on the Stack. This means that pushing the required return address onto the Stack before execution of the RST 28H will return control to the operating program.

In the process of execution, some modules, to perform part of the function, load other /SYS modules. The possibilities are almost endless.

Examples include :-
Loading SYS1/SYS which is the DOS command interpreter and is accessed to return to the DOS Ready prompt. This module naturally has the Library of commands resident in it. A DOS command does not load any other module.
Debug (SYS5/SYS) is loaded with a value of 27H in 'A' and an RST 28H.
Error conditions are handled by SYS4/SYS for DOS and SYS13/SYS for Basic both loaded by the RST 28H function call.
Accessing the routine in the DOS which executes a USR routine is also handled by this function as well as many other routines.

RESTARTS

RST 30H Code F7H The DEBUG Restart

0030H contains Jump to 400FH - 400FH contains :-

Level II	:	Return (C9)
Model I DOS	:	JP 4609H (C3 09 06)
Model III DOS	:	JP 45D4H (C3 D4 45)

As you see there is again no action in Level II.

But in DOS this is the DEBUG load Restart, the Code being :

```
4609H  F5       RST30H  PUSH   AF       ; Dummy to retain the
                                         ; proper Return Address
460AH  3E 27            LD     A,27H    ; Key for DEBUG Code
460CH  EF               RST    28H      ; Go to DEBUG.
460DH  ....... ....... ....... .......  ; Program continues
```

::::::::::::::::::::

RST 38H Code FFH The Interrupt Restart

0038H contains Jump to 4012H - 4012H contains :-

Model I Lev II	:	Enable Interrupts & Return (FB C9)
Model I DOS	:	JP 45F2H = Interrupt Service Routine
Mod III Lev2 + DOS	:	JP 3018H = Interrupt Service Routine

This is the System Entry point for all interrupts. The TRS80 uses only IM 1 (Interrupt Mode 1). In Level II it executes a "EI" (FBH = ENABLE INTERRUPTS) and a (C9H = RETURN) and immediately returns to the caller.

The byte at 4012H in the Model I has been documented as being an F3H which is Disable Interrupts. However, in reality, it is an FBH, the opposite, Enable Interrupts. This code has the unfortunate effect of causing the CPU to hang if the interrupts are already enabled when an Expansion interface connected.

In the DOS it is a JUMP to the INTERRUPT PROCESSOR. The reason for the Interrupt is determined and the next item from the Interrupt Service Routine is executed (mostly a Clock Interrupt). After this is completed control returns to the point of Interrupt and the Program continues.

DEVICE CONTROL BLOCKS

KEYBOARD CONTROL BLOCK

4015H - 401CH :- This area has the Keyboard Information

Model I -

4015H (16405) - The DCB Type Value (01)
4016H (16406) - Driver Address - least significant byte
4017H (16407) - Driver Address - most significant byte
In Level II = 03E3H : In DOS = 4516H : With ESF : 375CH
4018H (16408) to 401AH (16410) - Not used
401BH (16411) - Holds first of the Keyboard 2 Letter codes "K"
401CH (16412) - Contains second letter "I".

Model III -

4015H (16405) - The DCB Type Value (01)
4016H (16406) - Driver Address - least significant byte
4017H (16407) - Driver Address - most significant byte
In Level II = 3024H : In DOS = 44EBH
4018H (16408) - Right Shift Key status byte
4019H (16409) - Upper/Lower case toggle (0=U/C: Non zero=L/C)
401AH (16410) - Count interrupts per cursor char change (7)
401BH (16411) - Flag byte for cursor change 0=space 1=curs.char
401CH (16412) - Cursor blink toggle (0=Blink:Non zero=no blink)

VIDEO DISPLAY CONTROL BLOCK

401DH - 4024H :- This area has the Video Information

Model I -

401DH (16413) - DCB Type Value (07)
401EH (16414) - Driver Address - least significant byte
401FH (16415) - Driver Address - most significant byte
In Level II = 0458H : In DOS = 4505H : With ESF = 0458H
4020H (16416) - Cursor Position - least significant byte
4021H (16417) - Cursor Position - most significant byte
4022H (16418) - Character under Cursor storage
4023H (16419) - Key Letter - first letter (D)
4024H (16420) - Key Letter - second letter (O)

Model III -

401DH (16413) - DCB Type Value (07)
401EH (16414) - Driver Address - least significant byte
401FH (16415) - Driver Address - most significant byte
In Level II = 0473H : In DOS = 0473H
4020H (16416) - Cursor Position - least significant byte
4021H (16417) - Cursor Position - most significant byte
4022H (16418) - Character under Cursor (0 = Blink Inactive)
4023H (16419) - Cursor Character (0 if default B0H(176))
4024H (16420) - Control Byte - Character sets exchange

DEVICE CONTROL BLOCKS

LINE PRINTER CONTROL BLOCK

4025H TO 402CH :- This area has the Printer DCB Information

Model I -

4025H (16421)	-	THE DCB Type Value (06)
4026H (16422)	-	Driver Address - least significant byte
4027H (16423)	-	Driver Address - most significant byte In Level II & DOS & ESF = 058DH
4028H (16424)	-	Max Lines per Page + 1 : in IPL 43H(67)
4029H (16425)	-	No. of Lines Printed + 1 on the Page so far
402AH (16426)	-	Contains 0
402BH (16427)	-	First Letter of Key Letters (P)
402CH (16428)	-	Second Letter (R)

Model III -

4025H (16421)	-	The DCB Type Value (06)
4026H (16422)	-	Driver Address - least significant byte
4027H (16423)	-	Driver Address - most significant byte In Level II & DOS = 058DH
4028H (16424)	-	No. Lines per Page + 1 - in IPL 43H(67)
4029H (16425)	-	No. of Lines Printed on the Page so far
402AH (16426)	-	Current Printer Tab position
402BH (16427)	-	WIDTH parameter - Set by FORMS (SYS15/SYS) = Maximum Line length - 2
402CH (16428)	-	Default Maximum Characters per Line - 2

. . . . . .

Device Control Blocks - Type Values

Bit 1 : If set Device is capable of being used for Output
Bit 0 : If set Device is capable of being used for Input

The ROM I/O routine handling DCBs checks each Device to see if it has the required capability. The ROUTE command causes the DCB type value to be set to C0H if routed elsewhere. Address 4033H is used by the DOS to vector to the routine handling the ROUTE logic. If a ROUTE Command is not current the contents of 4033H can be used as a Jump to a user routine by loading the Keyboard DCB type value with 0.

That is all the Control Blocks in the Model I. The additional Control Blocks for the Model III are explained on Pages 21 and 22. These involve RS-232C and ROUTE Control Blocks.

THE DATA STORAGE AREA

402DH - 407FH :- 16429 - 16511 : 83 Bytes

This area is used mainly by DOS. There is a Keyboard Work area which is common to Model I & model III both Level II and DOS. Other than this the area is used for DOS storage and the Interrupts.

MODEL I

IN LEVEL II :-

402DH (16429) contains Jump to 5000H (could contain whatever)
4030H (16432) contains RESTART 00H or RESET
4031H - 4032H contain 00 00
4033H (16435) contains 3E 00 C9 = Load A with 0 and RETurn
4036H - 403CH is a Work area for the Keyboard Driver
403DH - 405CH is initialised by the IPL to zeros.
405DH - 407DH (16477-16507) Used by Stack during IPL
could be used for User's USR routines.
407EH - 407FH Storage or not used

UNDER NEWDOS/80 :-

402DH (16429) DOS entry point - Jump to 4400H
4030H (16432) Error already displayed exit to DOS
This is the address DOS jumps to at DOS READY to load the command interpreter (SYS1/SYS).
4033H (16435) Call Device driver in the DOS (Jump 4ADBH)
4036H (16438) Work area for SYS0/SYS and Keyboard Driver
403DH (16445) Control Byte for 64/32 char. screen (see Port FFH)
4040H (16448) Used by Interrupts - Cyclic Counter
4041H - 4046H (16449) Storage - Secs Mins Hrs Year Day Mth order
4047H (16455) Storage - Load address for DOS System Modules (5200H)
4049H (16457) Contains Memory Size as shown in the HIMEM value
404BH (16459) Current Interrupt Status Bytes Level 2 (DOS = 5D0H)
404DH - 405CH (16461-16476) Interrupt Sub-routine Vectors
405DH - 4062H Storage or not used
4063H - 407BH (16483) Temporary DOS routine (Converts DE to ASCII)
407CH - 407EH (16508) Storage Parameters for Time Checking

OTHER DOS STORAGE :-
4312H is the control byte for the Break key
4427H contains the 82H referring to NEWDOS/80 Version 2
442BH contains 01 for Model I
4410H - 4411H is the vector to insert an interrupt routine

DOS DATA STORAGE AREA

MODEL III

402DH - 407FH :- 16429 - 16511 - 83 Bytes

IN LEVEL II :-

402DH (16429) Contains Jump to 5000H
4030H (16432) Contains C7H = RESTART 00H = Reset
4131H - 4132H contain 00 00
4033H - 4034H (16435-16436) contains AF C9 = XOR A (A=0) and Return
4036H - 403CH (16438-16444) Work area for Keyboard Driver
403DH (16445) Interrupt Jump for I/O (35FBH:35FAH if inactive)
4040H (16448) Interrupt Jump for RS-232C Error (365AH)
4043H (16451) Interrupt Jump for Undefined Interrupt (3680H)
4046H (16454) Interrupt Jump for Time Clock (3529H)
4049H (16457) contains C7H = RESTART 00H = Reset
404AH - 404BH contain 00 00
404CH - 407CH (16460-16508) Used by Stack during IPL
407DH - 407FH (16509-16511) Storage or not used

UNDER NEWDOS/80 :-

402DH (16429) DOS entry point - Jump to 4400H
4030H (16432) Error already displayed exit to DOS
4033H (16435) Call Device driver in the DOS (Jump 4A80H)
4036H - 403CH (16438-16444) Work area for Keybrd Driver & SYS0/SYS
403DH (16445) Interrupt Jump for I/O (35FBH:35FAH if inactive)
4040H (16448) Interrupt Jump for RS-232C Error (365AH)
4043H (16451) Interrupt Jump for Undefined Interrupt (3680H)
4046H (16454) Interrupt Jump for Time Clock (4551H)
4049H (16457) Interrupt Jump for I/O during IPL (43A2H)
Return from Non-maskable Interrupt after IPL.
404BH - 4057H (16459) 13 byte work area for Utilities & DOS.
4058H - 407FH (16466-16511) Storage or not used

OTHER DOS STORAGE :-

441FH is the cyclic counter for Model III
4411H - 4412H contain the HIMEM value in Model III
4478H is the control byte for the Break key in Model III
4427H contains the 82H referring to NEWDOS/80 Version 2
442BH contains 03 signifying Model III
447BH - 447CH is the vector to insert an interrupt routine in Model III

THE BASIC STORAGE AREA

Common to Models I & III

4080H - 411CH :- 16512 - 16668 : 157 Bytes

All of this area is used by Basic in both the Models I and III and in both Level II and Disk Basic. It is used entirely for Basic program parameters, routines and addresses. The Basic interpreter relies on this area to find the parameters for any action it takes and uses bytes here for condition storage, addresses and values.

For example the 2 bytes at 40A2-3H tells the interpreter the current line number being executed and if the byte at 40DCH contains a 1 the interpreter knows that a FOR/NEXT loop is in progress.

4080H - 408DH (16512-16525) Division Support Routine
This whole area from 1080H to 10A5H is moved
from 18F7H to 1904H by the IPL for use by Basic.
408EH (16526) Address of USR Routine - Level II
- contains 1E4AH (Illegal function call) if inactive.
4090H - 4092H (16528-16530) Random number seed (3 bytes)
4093H - 4098H (16531-16536) Port handling
4099H (16537) Contains last character typed before Break
409AH (16538) Current ERR error code stored here
409BH (16539) Number of characters in current printer line
409CH (16540) Output Device code: -1=Cassette,0=Video,1=Printer
409DH (16541) Maximum width of video display line (works ONLY if
expression processed by ROM numeric evaluator)
409EH (16542) Size of video line after which " PRINT , " causes
a Line Feed to occur.
409FH (16543) Flag byte for LIST print: 1=inside quotes: 0=outside
(If inside quotes Mod III will print Symbols- not SCC's)
40A0H (16544) Address of String area lower boundary
40A2H (16546) Basic line number being executed (FFFFH if inactive)
40A4H (16548) Address of start of Basic Program
40A6H (16560) Cursor TAB position (POS value)
40A7H (16551) Points to Address of Keyboard Buffer + 2
40A9H (16553) Input flag - 0 = Cassette , else non-zero
40AAH (16554) Random number seed) 4 Byte
40ABH (16555) Value from Refresh Register) Random Number
40ACH (16556) Last 2 byte Random number) storage
40AEH (16558) Flag byte - 0 = locate named variable , 1 = Create
40AFH (16559) Type byte - Accumulator variable type -
2=% Integer, 3=$ String, 4=! Single Prec.& 8=# Double Prec.
40B1H (16561) Top of Memory Pointer

BASIC DATA STORAGE AREA

Basic storage common to Models I & III

40B3H - 40D7H - String work area including :-
  40B3H (16563) - Address of next available location in String area
  40B5H (16565) - 40D2H Contains the String Work Area
  40D3H (16595) - Contains length of current String
  40D4H (16596) - Points to Address of current String
  40D6H (16598) - Address of next available unused String space
40D8H (16600) 2 Functions - last byte executed in current Statement
                or - edit flag during Print Using
40DAH (16602) Line Number of last Data statement read
40DCH (16604) FOR/NEXT flag - 0 = not in progress, 1 = in progress
40DDH (16605) Flag byte - 1= Input of BASIC Statements, 0= inactive
40DEH (16606) Read flag - 0= READ active, 1= Input statement active
40DFH (16607) Execution Address for System Tape
              or VARPTR of current variable
40E1H (16609) Flag - 0= Inactive, 1 = Auto mode active
40E2H (16610) Current line number (used by INPUT or AUTO)
40E4H (16612) Auto line increment
40E6H (16614) Address of Statement being currently executed
40E8H (16616) Contains current Stack Address
40EAH (16618) Line number in which Error occured
40ECH (16620) Current Line number - used by "," edit or "." list.
40EEH (16622) Address of byte after Error (used by RESUME)
40F0H (16624) Addr of target Line # in ON ERROR GOTO Statement.
40F2H (16626) Flag byte - FFH if Error (cleared by Resume)
40F3H (16627) Addr of last byte successfully executed
      also the Address of the Decimal Point in the Print Buffer
40F5H (16629) Last Line number executed. Used by STOP,END,BREAK
40F7H (16631) Addr of next byte to be executed. Used by CONT.
40F9H (16633) Address of simple variables & End Basic Program
40FBH (16635) Address of dimensioned variables
40FDH (16637) Address of start of free Memory
40FFH (16639) Points to byte after last char read (READ Statement)
4101H - 411AH Variable type declaration list. The 26 Bytes here are
  (16641) -   loaded with 04H by a routine at 1B64H. This routine
   (16666)   is called during IPL initialisation, on NEW or the
              DOS command CLEAR, or after completion of EDITing a
              Basic Program line.
411BH (16667) Flag byte - 1 = Trace active, 0 = not active
411CH (16668) Temporary storage for floating point number

THE ACCUMULATORS

411DH - 4124H - 16669 - 16676 : The Main Accumulator (ACC or WRA1)

Address	Integer	Single Precision	Double Precision
411DH (16669)			Least sign. byte
411EH (16670)			Next byte
411FH (16671)			Next byte
4120H (16672)			Next byte
4121H (16673)	LSB	Least sign. byte	Next byte
4122H (16674)	MSB	Next byte	Next byte
4123H (16675)		Most sign. byte	Most sign. byte
4124H (16676)		Exponent	Exponent

4125H (16677) : Holds sign of result of operations
4126H (16678) : Bit bucket used during Double precision addition

4127H - 412EH : 16679 - 16686 : Auxiliary Accumulator (AACC or WRA2)

Address	Integer	Single Precision	Double Precision
4127H (16679)	LSB	Least sign. byte	Least sign. byte
4128H (16680)	MSB	Next byte	Next byte
4129H (16681)		Next byte	Next byte
412AH (16682)		Most sign. byte	Next byte
412BH (16683)		Exponent	Next byte
412CH (16684)			Next byte
412DH (16685)			Most sign. byte
412EH (16686)			Exponent

4130H (16688) - 4149H (16713) : Print buffer
414AH (16714) - 4151H (16721) : Double Precision division storage
4152H (16722) - 41A5H (16808) : Disk Basic Reserved Word jumps
Level II : loaded with jump to L3 Error by IPL
41A6H (16809) - 41E4H (16868) : ROM Call DOS Exits
Level II : loaded with RETURNs by IPL

Model I Level II :-
41E5H (16869) - 42E7H (17127) : Basic Input Buffer
42E9H (17129) : Start of Basic Program.

Model III :-
41E5H (16869) : RS-232C Device Control Blocks (next Page).

Mod III RS-232C CONTROL BLOCKS

41E5H - 41ECH : 16869 - 16876 : The RS-232C Input DCB
- 41E5H (16869) : Recognition Code (01)
- 41E6H (16870) : Driver Address (301EH)
- 41E8H (16872) : Input Byte Buffer
- 41E9H (16873) - 41EAH (16874) : Control Bytes
- 41EBH (16875) - 41ECH (16876) : "RI" - Recognition letters

41EDH - 41F4H : 16877 - 16884 : The RS-232C Output DCB
- 41EDH (16877) : Recognition Code Byte (02)
- 41EEH (16878) : Driver Address (3021H)
- 41F0H (16880) : Output 1 Byte Buffer
- 41F1H (16881) - 41F2H (16882) : Control Bytes
- 41F1H (16881) - 41F4H (16884) : "RO" - Recognition letters

41F5H - 41FCH : 16885 - 16892 : The RS-232C Initialisation DCB
- 41F5H (16885) : Recognition Code Byte (02)
- 41F6H (16886) : Initialisation Driver Address (301BH)
- 41F8H (16888) : Baud Rate : Send = Bits 7-4 : Receive = Bits 3-0

Rate	Error %	Code	Rate	Error %	Code
50	0	0	1800	0	8
75	0	1	2000	0.253	9
110	0	2	2400	0	10
134.5	0.016	3	3600	0	11
150	0	4	4800	0	12
300	0	5	7200	0	13
600	0	6	9600	0	14
1200	0	7	19200	3.125	15

41F9H (16889) : Parity/Word Length/Stop Bit Code Byte as follows :-

Bit 7 -	0 = Odd Parity	1 = Even Parity
Bit 6&5 -	00 = 5 Bit Words	01 = 6 Bit Words
-	10 = 7 Bit Words	11 = 8 Bit Words
Bit 4 -	0 = 1 Stop Bit	1 = 2 Stop Bits
Bit 3 -	0 = Enable Parity	1 = Disable Parity
Bit 2 -	0 = Disable Transmitter	1 = Enable Transmitter
Bit 1 -	0 = Set Data Terminal Ready (DTR) Signal low (Ready)	
	1 = Set Data Terminal Ready Signal high	
Bit 0 -	0 = Set Request to Send (RTS) Signal low (Data ready)	
	1 = Set Request To Send Signal high (No request active)	

41FAH (16890) - Control Byte for Wait for Serial I/O
0 = Do not wait : Non zero = Wait

41FBH (16891) - 41FCH (16892) : "RN" = Recognition letters

N.B. :- After changing RAM Control Bytes, the RS-232C must be reinitiali to have the new parameters for execution.

MODEL III EXTRA RAM ADDRESSES

41FDH - 4202H (16893) : ROM 6 byte Keyboard Work Area
4203H (16899) : DOS - SYS1/SYS load Jump (JP 4030H)
: Level II - JP 202EH = Disable Interrupts & Go to Basic
4206H (16902) : Interrupt Jump for RS-232C Output Register empty
4209H (16905) : Interrupt Jump for RS-232C Input EOT signal
4210H (16912) : Control Byte for Port EC (see Page 32).
4211H (16913) : Cassette Baud Rate Toggle (0=500, NonZero=1500)
4212H (16914) : Cassette Counter Byte (Tape load - flashes "*")
4213H (16915) : Control Byte for Port E0H (Interrupt Byte)
4214H (16916) : Video scroll protect count of lines (0-7)
4216H (16918) : Count interrupts per Time/Date update (25/30)
4217H (16919) to 421CH (16924) - Time and Date storage

421DH - 4224H : 16925 - 16932 = 8 Bytes : ROUTE Control Block
421DH (16925) : Recognition Code (02)
421EH (16926) : Driver Routine Address (3739H)
4220H (16928) : "ROUTE" Command Destination Letters (See below)
4222H (16930) : "ROUTE" Command Source Letters (as below)
KI = Keyboard: DO = Video: PR = Printer: MM = Main Memory: NL = Null
RI = RS-232C Input : RO = RS-232C Output : RN = RS-232C Initialise
4224H (16932) : Status byte for CONTROL (ShftDwnArr) FFH=inactive

Model III Level II :-
4225H - 42E4H : Storage or not used by Level II.
42E5H - 43E7H : Level II Basic Input Buffer
43E9H : Basic programs etc.

Model III NEWDOS/80 :-
4225H - 4274H : DOS Command Buffer
4275H - 427FH : SYSTEM Parameter Storage
4280H - 42B8H : DRIVE Storage including :-
4291H - 429AH = Drive 0 : 429BH - 42A4H = Drive 1
42A5H - 42AEH = Drive 2 : 42AFH - 42B8H = Drive 3
4300H - 43FFH : DOS Sector Buffer as well as 5100H - 51FFH
4400H - on : DOS main SYS0/SYS Module

::::::::::::::::::::

Note :- Control is implemented on the Model III by simultaneously pressing the **LEFT SHIFT key** + the DOWN ARROW key + the key involved. The left SHIFT key must be used. This function is used in some Tandy Software e.g. Press the SHIFT key + DOWN ARROW key + the Asterisk (the multiply key). This is the standard Model III way to get the screen print "JKL" function of NEWDOS/80. The routine is in ROM at 01D9H.

Note also that the Model I Basic Entry point at 06CCH is not available in the Model III. Use 1A19H.

THE MEMORY REFERENCE

CHAPTER 3

DOS

NEWDOS/80 RAM USAGE

MODEL I

```
4152H - 41A8H : Disk Basic Reserved Word Jumps
41A9H - 41E4H : ROM Call DOS Exits
41E5H - 41FFH : DOS Storage
4200H - 42FFH : DOS Sector buffer (also 5100H - 51FFH)
4300H - 4317H : Storage including PDRIVE & SYSTEM specs
4318H - 4367H : DOS Command buffer
4368H - 4370H : SYSTEM Storage
4371H - 439AH : PDRIVE Storage
439BH - 43FFH : DOS Storage
4400H - 4CFFH : DOS Main resident module (SYS0/SYS)
4D00H - 51FFH : DOS Overlay area
5200H - 56FFH : BASIC Overlay area
5700H - 64C6H : Disk BASIC Resident module
64C7H - 65C9H : Disk BASIC Input Buffer
65CAH - 66BCH : Disk BASIC Storage
66BEH - 6A44H : Disk I/O File Buffers (3)
6A46H -       : Start of BASIC Programs
```

MODEL III

```
4152H - 41A8H : Disk Basic Reserved Word Jumps
41A9H - 41E4H : ROM Call DOS Exits
41E5H - 41FCH : RS-232C Device Control Blocks
41FDH - 4224H : Storage
4225H - 4274H : DOS Command Buffer
4275H - 427FH : SYSTEM Storage
4280H - 42B8H : PDRIVE Storage
42B9H - 42FFH : DOS Storage
4300H - 43FFH : DOS Sector Buffer
4400H - 4CFFH : DOS Main resident module (SYS0/SYS)
4D00H - 51FFH : DOS Overlay area
5200H - 56FFH : BASIC Overlay area
5700H - 64C6H : Disk BASIC Resident module
64A0H - 65A2H : Disk BASIC Input Buffer
65CAH - 66BCH : Disk BASIC Storage
66BEH - 6A44H : Disk I/O File Buffers (Default 3)
6A46H -       : Start of BASIC Programs
```

DISK BASIC RESERVED WORD JUMPS

4152H - 41A5H : Contain Disk Basic reserved word vectors. 28 Vectors.

In Level II the IPL loads a Jump to the L3 Error routine in each. The Level II Basic Reserved Word Jump Addrs are on Pages 58 & 59.

In Disk Basic they are initialised to Jumps to Addresses where the particular Disk Basic system has the routines to handle them. These addresses change between DOSes as the routines are placed in areas suitable to the different DOSes. The Jumps given here are those for NEWDOS/80 Version 2.

Address Hex	Address Dec	Reserved Word	Function	Routine Mod I	Routine Mod III
4152H	16722	CVI	Convert to Integer	58F5H	58CEH
4155H	16725	FN	Function	577FH	577EH
4158H	16728	CVS	Convert to Single Precision	58F2H	58CBH
415BH	16731	DEF	Define Variable or Function	5852H	583FH
415EH	16734	CVD	Convert to Double Precision	58EFH	58C8H
4161H	16737	EOF	End of File	5EC5H	5E9EH
4164H	16740	LOC	Location in File	5ED1H	5EAAH
4167H	16743	LOF	Length of File	5EC9H	5EA2H
416AH	16746	MKI$	Make an Integer a String	58DEH	5837H
416DH	16749	MKS$	Make Single Precision String	58DBH	5824H
4170H	16752	MKD$	Make Double Precision String	58D8H	5881H
4173H	16755	CMD	DOS Command	57FFH	57ECH
4176H	16758	TIME$	Retrieving Time	58C4H	3030H
4179H	16761	OPEN	Open FCB on file	5795H	5795H
417CH	16764	FIELD	Define size of var.	5E63H	5E3CH
417FH	16767	GET	Input file data	6126H	60FFH
4182H	16770	PUT	Output file data	6125H	60F2H
4185H	16773	CLOSE	Close a File	5FA1H	5F7AH
4188H	16776	LOAD	Load File into Memory	574AH	574AH
418BH	16779	MERGE	Merge File into Memory	572EH	572EH
418EH	16782	NAME	FC error	1E4AH	1EA4H
4191H	16785	KILL	Remove File from DIR	643CH	6415H
4194H	16788	&	Convert hex to dec.	5790H	5790H
4197H	16791	LSET	Var to buffer pad right	5908H	58E1H
419AH	16794	RSET	Var to buffer pad left	5909H	58E2H
419DH	16797	INSTR	String search	5786H	5786H
41A0H	16800	SAVE	Output to Disk	573BH	573BH
41A3H	16803	LINE	Input function	579FH	579FH

ROM CALL DISK BASIC EXITS

41A6H - 41E4H : 16806 - 16868 : 63 Bytes 21 Vectors

In Level II the IPL loads the first Byte of these with a Return. The DOS will load the Addrs with Jumps appropriate to it's own System.

These Addresses are called by ROM & are Disk Basic exits. The calls from these Routines are for Disk Basic to intercept the call from the function involved and give control to the Disk Basic routine.

Address		Called by		Routine	Routine Model I	Routine Model III
41A6H	16806	19ECH	=	DOS Basic Errors	LD A,2FH + RST28H	
41A9H	16809	27FEH	=	USR	5892H	587FH
41ACH	16812	1A1CH	=	Basic Re-entry	2169H	2169H
41AFH	16815	0368H	=	Keyboard Input	575DH	575DH
41B2H	16818	1AA1H	=	Basic Input	5756H	5756H
41B5H	16821	1AECH	=	(Basic Program line	63B3H	638CH
41B8H	16824	1AF2H	=	(processing	63BCH	6395H
41BBH	16827	1B8CH	=	RUN, CLEAR & NEW	57B6H	5F8FH
41BEH	16830	2174H	=	PRINT	6438H	6411H
41C1H	16833	032CH	=	Output byte	6076H	604FH
41C4H	16836	0358H	=	Scan keyboard	63C0H	6399H
41C7H	16839	1EA6H	=	RUN	5746H	5746H
41CAH	16842	206FH	=	PRINT	5F52H	5F28H
41CDH	16845	20C6H	=	PRINT#	574BH	5F24H
41D0H	16848	2103H	=	End PRINT processing	5FA0H	5F79H
41D3H	16851	2108H	=	PRINT# Cassette write	5F3BH	5F14H
41D6H	16854	219EH	=	INPUT	5EA8H	5E81H
41D9H	16857	2AECH	=	MID$ =	578BH	578BH
41DCH	16860	222DH	=	Middle INPUT/READ	579AH	579AH
41DFH	16863	2278H	=	End INPUT/READ	5EB9H	5E92H
41E2H	16866	02B2H	=	System	5FA0H	5F79H

--- NEWDOS/80 USR ROUTINE POINTER ADDRESSES ---

USR No	Model I		Model III		USR No	Model I		Model III	
0	573AH	22330	589DH	22685	5	5744H	22340	58A7H	22695
1	573CH	22332	589FH	22687	6	5746H	22342	58A9H	22697
2	573EH	22334	58A1H	22689	7	5748H	22344	58ABH	22699
3	5740H	22336	58A3H	22691	8	574AH	22346	58ADH	22701
4	5742H	22338	58A5H	22693	9	574CH	22348	58AFH	22703

NEWDOS/80 DOS MODULES

In Disk IPL, SYS0/SYS loads some 160 Bytes into the Communications Area to set up pointers and parameters for the DOS. This covers 3 main areas. First the restarts are revectored to suit the DOS. Then 2 storage areas for DATE, TIME, interrupts, the PDRIVE and relevant SYSTEM specifications and Device Control Blocks are initialised.

The Main Module of DOS (SYS0/SYS) is loaded to 4400H - 50E7H, but the area 4D00H to the end is only used at BOOT and is overlaid immediately by SYS1/SYS which interprets the Input of DOS Commands.

The only SYS module, either DOS or Basic, that is over 1 gran long is the DOS module SYS6/SYS which handles the FORMAT, COPY and APPEND commands. This module uses 6 grans in an area starting at the DOS overlay area at 4D00H and ending at 6FFFH. This area is used for the Basic resident module as well as the Basic overlay area so it is not surprising that use cannot be made of this module from Basic.

Module	Grans	Resides	Under	Function
SYS0/SYS	3	4400H-4CFFH	DOS-	Resident- Disk I/O, Interrupts etc
SYS1/SYS	1	4D00H-51FFH	DOS-	Interprets DOS commands
SYS2/SYS	1	4D00H-51FFH	DOS-	DCB handling & RENAME
SYS3/SYS	1	4D00H-51FFH	DOS-	DCB handling,BLINK,BREAK, CLOCK,LC,VERIFY & PURGE
SYS4/SYS	1	4D00H-51FFH	DOS-	DOS Error messages
SYS5/SYS	1	4D00H-51FFH	DOS-	DEBUG
SYS6/SYS	7	4D00H-6FFFH	DOS-	FORMAT, COPY & APPEND
SYS7/SYS	1	4D00H-51FFH	DOS-	TIME,DATE,AUTO,ATTRIB, PROT,DUMP,HIMEM
SYS8/SYS	1	4D00H-51FFH	DOS-	DIR & FREE
SYS9/SYS	1	4D00H-51FFH	DOS-	BOOT,CHAIN,CHNON,MDCOPY,PAUSE,STMT
BASIC/CMD	4	5700H-64BFH	BASIC-	Resident Module
SYS10/SYS	1	4D00H-51FFH	BASIC-	GET & PUT
SYS11/SYS	1	4D00H-51FFH	BASIC-	RENUM
SYS12/SYS	1	4D00H-51FFH	BASIC-	REF
SYS13/SYS	1	4D00H-51FFH	BASIC-	ERROR Messages & part RENUM
SYS14/SYS	1	4D00H-51FFH	DOS-	CLEAR,CREATE,ERROR,LIST,PRINT,ROUTE
SYS15/SYS	1	4D00H-51FFH	DOS-	FORMS & SETCOM
SYS16/SYS	1	4D00H-51FFH	DOS-	PDRIVE
SYS17/SYS	1	4D00H-51FFH	DOS-	WRDIRP & SYSTEM
SYS18/SYS	1	5200H-56FFH	BASIC-	Interprets BASIC Commands
SYS19/SYS	1	5200H-56FFH	BASIC-	LOAD,SAVE,RUN,MERGE,CMD"F"
SYS20/SYS	1	5200H-56FFH	BASIC-	BASIC Program executor
SYS21/SYS	1	4D00H-51FFH	BASIC-	CMD"O"

DISK BASIC I/O FILE CONTROL BLOCKS

On entering Basic you are able to allocate space for Disk Input/Output by the number of File areas that you request. This number can be from 0 to 15 (Default = 3). The syntax is as follows :-

From DOS enter the Command - BASIC,n
Also you may enter the Command - BASIC,nV

This action saves "n" file areas for use in the Basic program. The "V" appended to the number of files areas to be set aside doubles the 256 bytes of space allocated to each Buffer. At the beginning of each FCB record in memory there is a block of 13 bytes which contains control information, followed by the 32 bytes of information which in turn is followed by the Buffer allocation. The 32 bytes contain :-

Byte 0 - bit 7 = Active flag
1 - access flags
2 - ASC, ASE & Update flags
3 & 4 - Buffer Address
5 - next record offset
6 - Drive Number
7 - Directory Entry code (DEC)
8 - EOF final sector offset
9 - Record length
5,10,11 - 3 byte RBA of next record
8,12,13 - 3 byte RBA of EOF
14-22 - File Extents
22-23 - FXDE pointer
24-31 - FXDE Extents

The number of file areas is stored in the Model I in 5F88H and in the Model III in 5F62H. Following is a list of Disk I/O FCB & Buffer memory useage for both Model I & Model III.

File Area	Start Record	Start DCB	Start Buffer	Start Basic Program	Start Basic with "V"
0	-	-	-	66BFH = 26303	66BFH = 26303
1	66BEH	66CBH	66EBH	67ECH = 26604	68ECH = 26860
2	67EBH	67F8H	6818H	6919H = 26905	6B19H = 27417
3	6918H	6925H	6945H	6A46H = 27206	6D46H = 27974
4	6A45H	6A52H	6A72H	6B73H = 27507	6F73H = 28531
5	6B72H	6B7FH	6B9FH	6CA0H = 27808	71A0H = 29088
6	6C9FH	6CACH	6CCCH	6DCDH = 28109	73CDH = 29645
7	6DCCH	6DD9H	6DF9H	6EFAH = 28410	75FAH = 30202
8	6EF9H	6F06H	6F26H	7027H = 28711	7827H = 30759
9	7026H	7033H	7053H	7154H = 29012	7A54H = 31316
10	7153H	7160H	7180H	7281H = 29313	7C81H = 31873
11	7280H	728DH	72ADH	73AEH = 29614	7EAEH = 32430
12	73ADH	73BAH	73DAH	74DBH = 29915	80DBH = 32987
13	74DAH	74E7H	7507H	7608H = 30216	8308H = 33544
14	7607H	7614H	7634H	7735H = 30517	8535H = 34101
15	7734H	7741H	7761H	7862H = 30818	8762H = 34658

DOS ROUTINES

The following is a list of NEWDOS/80 DOS Routines. The star conveys no TRSDOS equivalent. The first column is the Entry Address. The 2nd Column is the Jump at that Address OR a Restart 28H value. The Restart value is loaded into "A" to be interpreted by the routine at 4BC2H (Mod III = 4B67H) which is the RST 28H Code.

	Entry	JP/(A)	Mod3	Function
	402DH	4400H	4400H	No error exit to DOS
	4030H	43H	43H	Error already displayed exit to DOS
	4033H	4ADBH	4A80H	Jump to device driver in DOS
	4400H	23H	23H	This is where 402DH jumps to
	4405H	63H	63H	Enter DOS and Execute a Command
	4409H	26H	26H	DOS Error exit
	440DH	4609H	45D4H	DEBUG Entry point A=27H - RST28H)
	4410H	65H	447BH	Enqueue user timer interrupt routine
	4413H	85H	85H	Dequeue a user timer interrupt routine
	4416H	4762H	4711H	Keep the Drives rotating
	4419H	C3H	C3H	Execute a DOS Command and Return
	441CH	83H	83H	Extract a Filespec write Filespec in FCB)
	4420H	44H	44H	Open an FCB to a new or existing File
	4424H	24H	24H	Open an FCB to an existing File
	4428H	25H	25H	Close an FCB
	442CH	45H	45H	Kill a File associated with the FCB
	4430H	A4H	A4H	Load a Program File
	4433H	C4H	C4H	Load and execute a Program File
	4436H	49FCH	49A1H	Read Disk Sector/Move Rec-FCB to user Buff
	4439H	4A36H	49DBH	Write Sector/Move Record-user Buff to FCB
	443CH	4A32H	49D7H	Same as 4439H and a Verify is done.
	443FH	4B4CH	4AF1H	Position FCB to start of File
	4442H	4B73H	4B18H	Position FCB to specified Record in File
	4445H	4B62H	4B07H	Position FCB back 1 Record
	4448H	4854H	4AF9H	Position FCB to EOF
*	444BH	4809H	47AEH	Allocate File space
*	444EH	4B47H	4AECH	Position FCB to specified RBA
*	4451H	C5H	C5H	Write the EOF value from FCB to Directory
*	445BH	4776H	4723H	Select and power-up a specified Drive
*	445EH	47ECH	4791H	Test for a mounted Disk
*	4461H	2BH	2BH	Name routine Enqueue
*	4464H	4BH	4BH	Name routine Dequeue
	4467H	4BA6H	4B4BH	Send a Message to the Video display
	446AH	4BBCH	4B61H	Send a Message to the Printer
	446DH	44A7H	3036H	Convert the Time to HH:MM:SS format
	4470H	44C2H	3033H	Convert the Date to MM/DD/YY format
	4473H	A3H	A3H	Insert default Extension into Filespec

MODEL III TRSDOS CALLS & ADDRESSES

		NAME	Address Hex	Address Dec	Function	Model I Address
`		$DATE	3033H	12339	Returns the Date	
`		$TIME	3036H	12342	Returns the Time	
		$JP2DOS	402DH	16429	Transfer control to DOS	
X		$DATLOC	421AH	16922	Date storage location	(4044H)
X		$TIMLOC	4217H	16919	Time storage location	(4041H)
X		$CMDTXT	4225H	16933	Location of DOS command Buffer	(4318H)
	*	$FILPTR	428DH	17037	Gets Info on any File open	
	*	$RAMDIR	4290H	17040	Loads Directory Info to RAM	
	*	$COMDOS	4299H	17049	Executes a DOS command & goes to DOS	
	*	$CMDDOS	429CH	17052	Executes a DOS command & returns	
		$ERRDSP	4409H	17417	Displays DOS Error messages	
	*	$MEMEND	4411H	17425	Top of Memory storage	
	*	$DSPDIR	4419H	17433	Displays Directory of user Files	
		$SYNTAX	441CH	17436	Move File data to FCB	
		$INIT	4420H	17440	Opens File FCB or creates it	
		$OPEN	4424H	17444	Open existing File FCB	
		$CLOSE	4428H	17448	Closes file last used	
		$KILL	442CH	17452	Deletes Directory entry	
		$READ	4436H	17462	Loads next Record	
		$WRITE	4439H	17465	Writes Buffer to next Record No.	
		$VERF	443CH	17468	Same and then verifies	
		$REWIND	443FH	17471	Positions to first Record	
		$POSN	4442H	17474	Positions File to specified Record	
		$BKSPC	4445H	17477	Positions to previous Record	
		$POSEOF	4448H	17480	Positions to last Record in File	
X	*	$PUTEXT	444BH	17483	Adds Extension to File name	(4473H)
`	*	$DMULT	444EH	17486	Multiplication Routine (16Bit)	
`	*	$DIVIDE	4451H	17489	Division Routine (16Bit)	

* means these addresses are different in NEWDOS/80 (Page 28).
` means these addresses do not exist in the Model I.
X means that the addresses change for the Model I.
(The Model I address follows in brackets)

MOD I PORT FDC & PRINTER INFORMATION

The only Port the Model I uses is Port FF. The Port has a control byte in memory at 403DH. For Write it has the following functions:-

Bit 0 :	0 erases voltage on tape	: 1 (1) writes Positive voltage
Bit 1 :	0 depends on Bit 0	: 1 (2) writes negative voltage
Bit 2 :	0 turns Cassette off	: 1 (4) turns Cassette Motor on
Bit 3 :	0 enables 64 Char Mode	: 1 (8) enables 32 Char Mode

Bits 4 to 7 have no function.

The FDC uses addresses between 37DCH and 37EFH as communication bytes. All the addresses are write only excepting 37ECH which uses 1 byte for both Reading Status and Writing Command . The following is a list of the addresses and functions :-

37E1H	= Drive 0 Select	37ECH	= Command Status Register
37E3H	= Drive 1 Select	37EDH	= Track Register
37E5H	= Drive 2 Select	37EEH	= Sector Register
37E7H	= Drive 3 Select	37EFH	= Data Register

The WRITE Commands for Command Register 37ECH are listed on Page 32 being the same as Port F0H in the Model III.

The READ Command for Status Register 37ECH are dependent on the command type but in general are :-

Bit 0 - If set Disk Controller BUSY	: Bit 4 - If set Missing Record
Bit 1 - If set Data Request active	: Bit 5 - No function
Bit 2 - If set Lost Data	: Bit 6 - No function
Bit 3 - If set CRC Error	: Bit 7 - If set FDC Not Ready

The Model I uses memory location 37E8H as Printer control byte. It has the same Read & Write functions as Port F8H in the Model III (Page 33).

.....

PMC80 & VIDEO GENIE & SYSTEM 80 PORTS

The System 80 has some differences in hardware to the TRS-80 including Port usage. Ports FF and FE are used for Cassette operations while Port FD is used for Printer operations.

Port FF has exactly the same functions as in Model I. With Port FE, the Cassette Number Select Bit is Bit 4. If set the Computer uses Cassette No. 2, If not set No. 1.

In Port FD, Bit 7 returns the Status of the Printer when READ, and to WRITE a Byte to the Printer uses all 8 bits) send the ASCII to Port FD.

MOD III PORT FDC & PRINTER INFO.

The Model III depends on Ports for all Floppy Disk Controller FDC Printer, RS-232C, and Interrupts functions as well as Cassette switching, 32 Char Mode, Video waits and Char set control. Thus bytes in the 37DCH to 37EFH area set aside for I/O in the Model I are not used in this way in the Model III excepting 37E8H, the Printer status byte (same as Model I.)

Some Ports have specialised functions. For example, E0H is the Interrupt control Port. It is read by the Interrupt Service Routine to find which device is interrupting. E4H is the Interrupt request Port. E8H, E9H, EAH and EBH are device control.

Note:- INTRQ = Interrupt request: DRQ = Data request.

PORT	Bit	READ	Bit	WRITE
E0 224 Control Byte At 4213H	0	Cass 1500 BAUD Rising Edge Int	0	0 = Disable: 1 = Enable
	1	Cass 1500 BAUD Falling Edge Int	1	0 = Disable: 1 = Enable
	2	Real Time Clock Interrupt	2	0 = Disable: 1 = Enable
	3	External I/O Interrupt	3	0 = Disable: 1 = Enable
	4	RS232 Transmit Interrupt	4	No WRITE function
	5	RS232 Receive Interrupt	5	No WRITE function
	6	RS232 Error Interrupt	6	0=Disable: 1=Enable Disk INTRQ
	7	No function	7	0=Disable: 1=Enable Disk DRQ
PORT	Bit	READ	Bit	WRITE
E4 228 Inter'pt Request Port	0	No READ function	0	Writing 00
	1	No READ function	1	to Port E4
	2	No READ function	2	disables
	3	No READ function	3	Disk I/O
	4	No READ function	4	non-
	5	RESET is pressed	5	maskable
	6	Disk DATA Request active	6	interrupts
	7	Disk INTERRUPT Request active	7	
PORT		READ		WRITE
E8 232		MODEM Status		UART Reset
PORT		READ		WRITE
E9 233		BAUD Rate Status		BAUD Rate Register
PORT		READ		WRITE
EA 234		UART Status		UART Control Register
PORT		READ		WRITE
EB 235		UART Receive		UART Transmit

MOD III PORT FDC & PRINTER INFORMATION

Port ECH (like Port FFH) is a multi function Port.
ECH has no function for Read other than an interrupt clear.
FFH has no function for Write other than Cassette Bit saving.

Port F0H is the FDC Status/Command Register as the byte at 37ECH is in the Model I.

PORT	Bit	READ	Bit	WRITE
	0	No Read function	0	Clock Print on
EC	1	except	1	Cassette Motor on
236	2	That a Read	2	32 Character Mode
	3	Of this Port	3	Altnerate Char. Set toggle
Control	4	clears	4	External I/O Bus enabled
Byte	5	the RTClock	5	Video Waits enabled
At	6	Interrupts	6	Speed-up : Set = *2
4210H	7		7	No function
PORT	Bit	READ		WRITE
	0	If set Disk Controller BUSY		See the Chart following for details
F0	1	If set Data Request active		
240	2	" " Lost Data		
	3	" " CRC Error		
FDC	4	" " Missing Record		
Command	5	No function		
/Status	6	" "		
Port	7	If set FDController Not Ready		

Write values to Port F0H (240) in Model III or 37ECH in Model I :-

FUNCTION	7,6,5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Restore	000	= 0			Track stepping Rate	
Seek	000	= 1	0= No	Not used	0 = 3ms	0 = 10ms
Step Head	001	0= Track	Verify		1=10or20	1 = 20ms
Step Head In	010	Update	1 =			
Step Head Out	011	1= No Update	Verify			
Read Data	100	0= 1 Sector	0 =NonIBM	0=NoHdSettle	= 0	= 0
Write Data	101	1 = MultSect	1=IBM Fmt	1=10msHdSet1	= Address Mark	
ReadAddrField	110	= 0	= 0	= 1	= 0	= 0
ForceInterrupt	110	= 1	NoCondits	onIndexpulse	Not Ready	on Ready
Read Track	111	= 0	= 0	= 1	= 0	= 0
Write Track	111	= 1	= 0	= 1	= 0	= 0

Note:-) denotes multiple application.

MOD III PORT FDC & PRINTER INFO.

The Ports F1H, F2H, F3H and F4H are Floppy Disk Controller Registers. F1H has the same function as 37EDH in the Model I, F2H the same as 37EEH, and F3H the same as 37EFH. The first 4 bits of Port F4H select the Drive to be used. The top 4 Bits tell the FDC what Density, Side, Write Compensation or Wait state to use.

PORT		READ		WRITE
F1 241				Track Register
PORT		READ		WRITE
F2 242				Sector Register
PORT		READ		WRITE
F3 243		Data Register		
PORT	Bit	READ	Bit	WRITE
	0	No Read function	0	Select Drive 0
F4	1	No Read function	1	Select Drive 1
244	2	No Read function	2	Select Drive 2
	3	No Read function	3	Select Drive 3
Select	4	No Read function	4	0=Select Front Side,1=Back
Port	5	No Read function	5	Write Compensation Enabled
	6	No Read function	6	Wait States Enabled
	7	No Read function	7	0= SingleDensity.1= Double
PORT	Bit	READ	Bit	WRITE
	0	No Read function	0	
F8	1	No Read function	1	ASCII
248	2	No Read function	2	(7Bits)
	3	No Read function	3	to
Printer	4	If set Printer has Fault	4	Printer
Port	5	If set Printer is Selected	5	
	6	If set Printer has no Paper	6	
	7	If set Printer is Busy	7	Not used
PORT	Bit	READ	Bit	WRITE
	0	If set 1500 Baud Rate enabled	0	If set Positive Voltage toTape
FF	1	If set Cassette Motor On	1	If set Negative Voltage toTape
255	2	If set 32 Char Mode enabled	2	No Write function
	3	If set Symbol Chars enabled	3	No Write function
	4	If set Ext I/O Bus enabled	4	No Write function
	5	If set Video Waits enabled	5	No Write function
	6	No Read function	6	No Write function
	7	If set 500 Baud Rate enabled	7	No Write function

PART 2

TRS – 80 INFORMATION

TRS-80 INFORMATION

CHAPTER 4

INITIALISATION

INITIAL PROGRAM LOADER

In this Chapter we examine what happens when the reset button is pressed. The Code here is called the Initialisation process and the code concerned is called the Initial Program Loader (IPL). For Level II initialisation the code is from the TRS80 Model I (the old ROM).

From cold start the computer starts reading the ROM at 0000H. The code in the following pages explains what then happens.

Firstly the Level II Initial Program Loader is explained and the code which is involved is disassembled with remarks.

Then the operations involved with the IPL under Model I DOS are similarly examined. Then the Model III Boot Sector is disassembled. The code involved in the Model III from 4300H to 4348H is nearly an exact copy of the Model I code from 4200H to 4255H but from this point on the codes differ. Disk access in the Model III uses Ports whereas in the Model I the controller is memory mapped. So it might help to read both explanations up to this point as they complement each other.

The TRS-80 uses only IBM format for its disk storage. This entails the existence of a particular pattern of control information between the sectors on the diskette. This information is invisible to the user except with very sophisticated utility programs.

The numbers to the left of the address in the disassembly denote how many times that address is referenced. The extra number immediately following it (e.g. 43A5H on Page 52) indicates that 43A5H is not referenced but that 43A6H is.

IPL — LEVEL II

```
   Addr.    Code          Instruction              Function

   0000H  F3             DI                  ; Turn off Interrupts
   0001H  AF             XOR A               ; Clear A Reg and Flags
   0002H  C3 74 06       JP  0674H           ; Go to start of IPL processing

   ;  ..........  to 0674H
1  0674H  D3 FF          OUT A,0FFH          ; Load Cassette Port with 0
1  0676H  21 D2 06       LD  HL,06D2H        ; Addr. of sequence to be shifted
   0679H  11 00 40       LD  DE,4000H        ; Start of Communications Region
   067CH  01 36 00       LD  BC,0036H        ; No. Bytes = 54
   067FH  ED B0          LDIR                ; Move 06D2-0707 to 4000-4035
   0681H  3D             DEC A               ; 0 = 256  reduced by 2
   0682H  3D             DEC A               ;  till it is 0
   0683H  20 F1          JR  NZ,0676H        ; Do it 128 times
   0685H  06 27          LD  B,27H           ; Load 39 addresses
1  0687H  12             LD  (DE),A          ; (4036-4062) with "A" (=0)
   0688H  13             INC DE              ; Incr. destination Pointer
   0689H  10 FC          DJNZ 0687H          ; Go back if B <> 0
   068BH  3A 40 38       LD  A,(3840H)       ; Test for Break key pressed
   068EH  E6 04          AND 04H             ; Eliminates all except Bit 2
   0690H  C2 75 00       JP  NZ,0075H        ; Go to Level II IPL
   0693H  31 7D 40       LD  SP,407DH        ; New Stack area
   0696H  3A EC 37       LD  A,(37ECH)       ; Load Disk Status
   0699H  3C             INC A               ; Test if Expansion Interface
   069AH  FE 02          CP  02              ;   & Disks present
   069CH  DA 75 00       JP  C,0075H         ; Go if not Disk to Level II
   ;  ..................  And so on to 0075H.
                          ..........................
```

The first of the Communications Area is set up with the proper addresses. It is remarkable that the ROM does this 128 times. The reason for this is to make absolutely sure that Restarts and Device Control Blocks are properly initialised (Memory power-up delay).

The area 4036H to 4062H is zeroed, and a test is performed to see if the Break key is pressed. If so, there is a jump to Level II IPL.

The stack is then set to 407DH. The stack overwrites most of the area between 405DH and 407CH in the IPL in Level II. This is the storage area in the DOS for a subroutine and the parameters for time checking and printing and is free for use in Level II by USR routines from Basic.

The test for the presence of an Interface is then carried out. This is done by reading the value in 37ECH. If it is not present, the value returned will be FFH. The value is incremented to give a Carry when compared with 02H so that control passes to 0075H.

IPL — LEVEL II

```
   Addr.  Code          Instruction              Function

2 0075H  11 80 40      LD  DE,4080     ; Load into 4080H - 40A6H
  0078H  21 F7 18      LD  HL,18F7     ;   from 18F7H - 191DH
  007BH  01 27 00      LD  BC,0027     ; 39 Bytes of Division
  007EH  ED B0         LDIR            ;    support routine, data  etc.
  0080H  21 E5 41      LD  HL,41E5     ; More initialisation
  0083H  36 3A         LD  (HL),3AH    ; 41E5H = 3A Start of Buffer
  0085H  23            INC HL          ; Move on to 41E6H
  0086H  70            LD  (HL),B      ; 41E6H = 0  Buffer code
  0087H  23            INC HL          ; Move on to 41E7H
  0088H  36 2C         LD  (HL),2C     ; 41E7H = 2CHir code
  008AH  23            INC HL          ; Move on to 41E8H
  008BH  22 A7 40      LD  (40A7H),HL  ; Basic Input Buffer Pointer
  008EH  11 2D 01      LD  DE,012D     ; Address of L3 Error
  0091H  06 1C         LD  B,1CH       ; 4152H - 41A5H = 28i
  0093H  21 52 41      LD  HL,4152     ; Move on to 4152H & load
1 0096H  36 C3         LD  (HL),C3     ;   Jump instruction
  0098H  23            INC HL          ; Move on one Byte
  0099H  73            LD  (HL),E      ;   load 2DH
  009AH  23            INC HL          ; Move to MSB
  009BH  72            LD  (HL),D      ;   load 01H
  009CH  23            INC HL          ; Move to next C3H Address
  009DH  10 F7         DJNZ 0096H      ; 28 Times ( 84 locations )
  009FH  06 15         LD  B,15H       ; Loop count of DOS exits
1 00A1H  36 C9         LD  (HL),C9     ; Load a Return
  00A3H  23            INC HL          ;   every 3 Bytes
  00A4H  23            INC HL          ;    changing DOS Exits
  00A5H  23            INC HL          ;     from Jump to Return
  00A6H  10 F9         DJNZ 00A1H      ; Repeat 21 times
```

. . . . . .

Division support routine is loaded from ROM to its address 4080H.

Address 41E5H is loaded with code that loads the register "A" with the contents of 2C00H which in the Model I contains D3H (211).

The Basic Input Buffer pointer (40A7H) is set to 41E8H.

A jump to the L3 Error (Disk Basic reserved word jumps) routine is put at the addresses of the DOS Reserved Word Jumps.

Then the ROM Call DOS Exits are plugged with a C9H (Return).

IPL — LEVEL II

```
   Addr.    Code          Instruction          Function

   00A8H   21 E8 42     LD   HL,42E8H    ; Store 0 in 42E8H, the byte
   00ABH   70           LD   (HL),B      ;   before start of Basic
   00ACH   31 F8 41     LD   SP,41F8H    ; Stack Address during IPL
   00AFH   CD 8F 1B     CALL 1B8FH       ; Initialise Basic pointers
   00B2H   CD C9 01     CALL 01C9H       ; Clear Screen
2  00B5H   21 05 01     LD   HL,0105H    ; ASCII pointer "Memory Size"
   00B8H   CD A7 28     CALL 28A7H       ; Print Message
   00BBH   CD B3 1B     CALL 1BB3H       ; User Input of Memory Size
   00BEH   38 F5        JR   C,00B5H     ; If Break go back
   00C0H   D7           RST  10H         ; Examine response
   00C1H   B7           OR   A           ; Clear Flags
   00C2H   20 12        JR   NZ,00D6H    ; Go past if Constant (val>0)
   00C4H   21 4C 43     LD   HL,434CH    ; If val = 0 get TOPMEM
1  00C7H   23           INC  HL          ;   by testing
   00C8H   7C           LD   A,H         ;     until
   00C9H   B5           OR   L           ;       HL = 0
   00CAH   28 1B        JR   Z,00E7H     ; If 0 then go & store
   00CCH   7E           LD   A,(HL)      ; Store original contents
   00CDH   47           LD   B,A         ;   in B for replacement
   00CEH   2F           CPL              ; Complement (gives test byte)
   00CFH   77           LD   (HL),A      ; Place test byte in Address
   00D0H   BE           CP   (HL)        ; Compare test byte
   00D1H   70           LD   (HL),B      ; Restore original
   00D2H   28 FE        JR   Z,00C7H     ; If OK keep going
   00D4H   18 11        JR   00E7H       ; Else Address is non-existent
```

. . . . . .

The byte at 42E8H is zeroed for start of Basic Programs. This situation is checked by the ROM when the start of Basic is reset or reinitialised.

The Stack is set temporarily at 41F8H.

Basic Pointers are initialised in 1B8FH.

The Screen is cleared by a ROM call to 01C9H.

And the message "Memory Size" is printed at the top of the Screen by another ROM call at 28A7H.

The computer then waits for an Input (terminated by <ENTER>) of the Memory Size value. If a value is entered, the computer jumps to 00D6H (the next page). If no value, the Memory is tested as to how much is present. In a 48K Computer, this accounts for the delay if no Memory Size is entered.

IPL — LEVEL II

```
  Addr.    Code          Instruction          Function

1 00D6H  CD 5A 1E     CALL 1E5AH        ; Get Binary equivalent
  00D9H  B7           OR   A            ; Clear flags
  00DAH  C2 97 19     JP   NZ,1997H     ; Syntax error if no value
  00DDH  EB           EX   DE,HL        ; TOPMEM plus 1 into HL
  00DEH  2B           DEC  HL           ;   less 1
  00DFH  3E 8F        LD   A,8FH        ; Comparison value
  00E1H  46           LD   B,(HL)       ; Store contents in B
  00E2H  77           LD   (HL),A       ; Poke test pattern
  00E3H  BE           CP   (HL)         ; Compare test byte
  00E4H  70           LD   (HL),B       ; Restore original
  00E5H  20 CE        JR   NZ,00B5H     ; Memory not present go back
2 00E7H  2B           DEC  HL           ; Amount Memory less 2
  00E8H  11 14 44     LD   DE,4414H     ; DE = 17428
  00EBH  DF           RST  18H          ; Test for min amount of memory
  00ECH  DA 7A 19     JP   C,197AH      ; Out of memory if carry
  00EFH  11 CE FF     LD   DE,FFCEH     ; = default String space
  00F2H  22 B1 40     LD   (40B1),HL    ; Store TOPMEM
  00F5H  19           ADD  HL,DE        ; Subtract String space
  00F6H  22 A0 40     LD   (40A0),HL    ; Store start of String space
  00F9H  CD 4D 1B     CALL 1B4DH        ; Initialise Basic Var. Pointers
  00FCH  21 11 01     LD   HL,0111H     ; ASCII "Radio Shack"
  00FFH  CD A7 28     CALL 28A7H        ; Print message
  0102H  C3 19 1A     JP   1A19H        ; Go to Ready & Command mode
1 0105H  4D 45 4D 4F 52 59   etc        ; "Memory Size" + terminator
1 0111H  52 41 44 49 4F 20   etc        ; "Radio Shack etc." +terminator
```

. . . . . .

The code at 00D6H calls a routine at 1E5AH that converts the HL value (HIMEM) to an Integer and returns the result in DE with the A register holding the Error condition. If A<>0 an error occurs.

If O.K. the value is transferred back to HL, decremented, tested, decremented again, tested for the minimum amount of Memory, and the value (HIMEM) stored at 40B1H.

Default String space is set, the String boundary is set, and the other Basic pointers initialised.

Radio Shack Level II (or in the later Models R/S etc) is printed and control passes to the Command mode.

MODEL I – DOS INITIALISATION

In this part we examine what happens when the reset button is pressed in DOS. On Page 36 the ROM was disassembled until it branched to the Level II IPL. The code starts from there.

The Disk access by the IPL is handled in a different way to normal DOS I/O. The Boot sector is loaded directly into its execution address because of the absence of control information which is always present in DOS I/O.

Later we will disassemble the Boot sector and see how it loads (SYS0/SYS) to start the DOS initialisation process. But first to carry on from where Level II branched off :-

```
  Addr.    Code        Instruction          Function

  069CH  DA 75 00    JP  C,0075H    ; Go if not Disk to Level II
  ;  ..............  and so to the DOS IPL
  069FH  3E 01       LD  A,1        ; Unit Select Mask for Drive 0
  06A1H  32 E1 37    LD  (37E1H),A  ; Select Drive 0
  06A4H  21 EC 37    LD  HL,37ECH   ; Addr of Command/Status Register
  06A7H  11 EF 37    LD  DE,37EFH   ; Addr of Data Register
  06AAH  36 03       LD  (HL),3     ; 3 = Restore to Command Reg
  06ACH  01 00 00    LD  BC,0       ; Delay Count = 65536
  06AFH  CD 60 00    CALL 0060H     ; Delay for 1 Second
1 06B2H  CB 46       BIT 0,(HL)     ; Test if Controller busy
  06B4H  20 FC       JR  NZ,06B2H   ; Loop till not busy
  06B6H  AF          XOR A          ; Zero A and clear flags
  06B7H  32 EE 37    LD  (37EEH),A  ; Make Sector Reg = 0
  06BAH  01 00 42    LD  BC,4200H   ; Addr to read Data to
  06BDH  3E 8C       LD  A,8CH      ; 8CH = Read Command
  06BFH  77          LD  (HL),A     ; Read Track 0, Sector 0
2 06C0H  CB 4E       BIT 1,(HL)     ; Test if Data ready
  06C2H  28 FC       JR  Z,06C0H    ; Wait until Data there
  06C4H  1A          LD  A,(DE)     ; Get next Byte from Data Reg
  06C5H  02          LD  (BC),A     ; Transfer to 4200H plus in RAM
  06C6H  0C          INC C          ; Increment Buffer Pointer
  06C7H  20 F7       JR  NZ,06C0H   ; Do till C = 0  ( 256 Bytes )
  06C9H  C3 00 42    JP  4200H      ; Transfer control to BOOT Sec
                     ..........................
```

Drive 0 is selected and after a delay, the Disk controller is tested to see if busy. When free, Track 0 Sector 0 (the Boot Sector) is read and written into memory at 4200H. When 256 bytes have been read, control is passed to the Boot sector in Memory.

MODEL I BOOT SECTOR

The Boot program is now in memory at 4200H. As the last line of the code on the previous page shows, control is transferred to the Boot sector immediately it is loaded.

In the 6th Sector of the Disk is the SYS0/SYS program. This is the code that prints the logo for NEWDOS/80 on the screen, and initialises all vectors and addresses to the right values for DOS to execute correctly.

SYS0/SYS is made up in the first part of all these addresses and values so that until it starts at 4400H there is no continuous code, only odd addresses and parameters. (The only exception is the code at 4063H (Mod3-44ADH) which converts a Hex. value in DE to ASCII and stores it in (HL).

The Boot sector loads SYS0/SYS starting at 4000H. The following is a diagram of what the start of Drive 0 Drive Relative Sector 5 looks like using Superzap :-

```
                                                    .
        Tells the computer to load
            This is how many Bytes incl 2 Address Bytes (11 bytes)
                  Address to load DATA to
                        DATA (9 bytes)
                                                                  and again
00 - 01 0B  0C 40  33 C2  4B C3  09 46  C3 F2  45 01  0B 2D

10 - 40 C3  00 44  2E 43  EF C3  DB 4A  01 21  3E 40  00 00
                                                  ---
                                                  and again
```

Note :- Because the 2nd byte is 11 then 9 bytes of data are loaded
If the 2nd byte is 3 then 1 byte is loaded
If the 2nd byte is 0 then 254 bytes are loaded
If the 2nd byte is 2 then 256 bytes are loaded
This is because address bytes and the 1st data byte are loaded before the counter, the Register C, is tested to see if it is zero.

We now consider the BOOT sector. It is disassembled and explained on the next page. Unlike the previous load, the Boot sector loads SYS0/SYS into the Sector Buffer. The code is then transferred to its proper address in memory. The BOOT Sector changes depending on the characteristics of the diskette. However, there is no real change in function, only in parameters. e.g. single/double sided etc.

MODEL I – BOOT SECTOR

```
  Addr.   Code          Instruction          Function

  4200H   00            NOP                  ; No operation
  4201H   FE 11         CP   11H             ; Recognition Code + 17 = Dir Sector
  4203H   F3            DI                   ; Disable Interrupts
  4204H   21 EC 37      LD   HL,37ECH        ; Command/Status Register
  4207H   36 FE         LD   (HL),FEH        ; Code for Request  ) Activate
  4209H   36 D0         LD   (HL),D0H        ; Terminate Request )   Drive
  420BH   23            INC  HL              ; Move to Track Register
  420CH   36 00         LD   (HL),00H        ; Track 0
  420EH   23            INC  HL              ; Sector Register
  420FH   36 00         LD   (HL),00H        ; Sector 0
  4211H   11 05 00      LD   DE,0005H        ; D=Track=0,E=Sector=5
  4214H   D9            EXX                  ; Alternate Registers
  4215H   31 E0 41      LD   SP,41E0H        ; Set Stack to Comm. area
  4218H   21 FF 51      LD   HL,51FFH        ; Set HL to Sector Buffer
1 4218H   CD 52 42      CALL 4252H           ; Get 1st Sector
  421EH   FE 20         CP   20H             ; Set or not Carry Flag
  4220H   47            LD   B,A             ; Store for 422CH
  4221H   30 29         JP   NC,4249H        ; Go to Error handling if > 1FH
  4223H   57            LD   D,A             ; If byte >2 & <32 load to ROM
  4224H   CD 52 42      CALL 4252H           ; 2nd Byte (1st area = 0BH)
  4227H   4F            LD   C,A             ; Store No. of Bytes
  4228H   CD 52 42      CALL 4252H           ; Get 3rd Byte (1st area = 00H
  422BH   5F            LD   E,A             ; LSB of Address to load to
```

After the 1st byte "NOP" is a recognition code "FE" and the Directory starting Track. This value is used by NEWDOS/80 as well as its own record in the 3rd Sector - the PDRIVE specifications.

HL is pointed to the memory address which has the function of READing the status of and WRITing commands to the Floppy Disk Controller. Firstly the FDC is activated by a request. This starts the drive motor rotating. Then a request for termination is sent. HL is moved on to the other memory locations for FDC information. The Track Register and the Sector Register are set to track 0 and sector 0. DE is loaded with the track and sector number that will be required for loading the SYS0/SYS program. The HL, DE and BC register pairs are stored by the EXX for use in the disk-to-sector buffer load and the primed ones are exchanged for sector buffer-to-memory load.

The stack is relocated to the communications area. HL is pointed to the sector buffer address. The call to 4252H increments the value in L and since, in the first instance, L has been loaded with FFH, the RET NZ allows the disk-to-sector buffer load. Before return from the load the the value read from the first byte in the sector buffer is checked and if greater than 31 (1FH) an error routine clears the screen, prints 'NO SYS' and locks up. If <31 the value is loaded into B for DJNZ instructions to decrement and determine the flow.

MODEL I BOOT SECTOR

```
   Addr.    Code      Instruction          Function

   422CH  10 12       DJNZ 4240H     ; If 1st byte <> 1 go to 4240H
   ;     Sector buffer-to-memory transfer
   422EH  CD 52 42    CALL 4252H     ; Go & read MSB of
   4231H  57          LD   D,A       ;    Address to load to
   4232H  0D          DEC  C         ; allow for 2 Byte Addr in
   4233H  0D          DEC  C         ;   count of Bytes to be loaded
   4234H  2C          INC  L         ; Move along in Buffer
   4235H  CC 55 42    CALL Z,4255H   ; if end Buffer load another sector
   4238H  7E          LD   A,(HL)    ; then load DATA Reg
   4239H  12          LD   (DE),A    ;   into Address in DE.
   423AH  13          INC  DE        ; move Address on
   423BH  0D          DEC  C         ; 1 Byte less to load
   423CH  20 F6       JR   NZ,4234H  ; if more go back
   423EH  18 DB       JR   4218H     ; or go load another area
1  4240H  10 F9       DJNZ 423BH     ; If 1st byte <> 2 load to ROM
   ;    Get transfer address and execute or Error
   4242H  CD 52 42    CALL 4252H     ; else read next byte
   4245H  57          LD   D,A       ; MSB of transfer address (4D00H)
   4246H  1A          LD   A,(DE)    ; Check if contents of 4D00H
   4247H  FE A5       CP   A5H       ;    is A5H
   4249H  13          INC  DE        ; Move on to 4D01H
   424AH  D5          PUSH DE        ; To stack : if OK return to
   424BH  C8          RET  Z         ;    transfer addr. or error
1  424CH  21 E5 42    LD   HL,42E5H  ; 'NO SYS' ASCII
   424FH  C3 C3 42    JP   42C3H     ; Go to Error routine.
```

. . . . . .

If the first byte is not 1 or 2 and is less than 32 then the value is given to 'D'. The DJNZ at 422CH branches the program to the DJNZ at 4240H and this passes to 423BH. The code loads the data in the buffer into an address in DE which is in the 300H to 1FFFH range. This allows an ASCII heading at the start of a file.

After storing a machine language program on Disk the DOS adds 02 02 xx xx where the first 02 is the end of program code, as explained above, the second 02 is a byte load count and xx xx is the transfer address. If the call at 421BH reads an 02H then the DJNZ at 422CH plus the one at 4240H cause a fall through to 4242H. The MSB of the transfer address is read and the contents of this address is checked to see if it is A5H. If so control passes to the transfer address + 1. If not the 'NO SYS' error handling routine executes.

The code 4234H to 423CH handles the load from the buffer-to-memory. If the load continues across sectors the call at 4235H tests for the end of the buffer and loads another sector.

MODEL I — BOOT SECTOR

```
   Addr.   Code          Instruction          Function

5 4252H   2C            INC   L              ; Move on 1 Byte in buffer
  4253H   7E            LD    A,(HL)         ; Store in A
  4254H   C0            RET   NZ             ; Go back if not end of buffer

  ;        load a new sector into the buffer
1 4255H   D9            EXX                  ; Alternate Registers
  4256H   06 0A         LD    B,0AH          ; 10 error tries
1 4258H   21 E1 37      LD    HL,37E1H       ; Drive control Reg
  425BH   36 01         LD    (HL),01H       ; select Drive 0
  425DH   D5            PUSH  DE             ; Store both Track# & Sector#
  425EH   C5            PUSH  BC             ; and error count
  425FH   7B            LD    A,E            ; check sector count
  4260H   D6 0A         SUB   0AH            ; is it more than 10 ?
  4262H   38 03         JR    C,4267H        ; if not jump
  4264H   5F            LD    E,A            ; if so subtract 10
  4265H   36 09         LD    (HL),09H       ; select Drive 0
1 4267H   21 EC 37      LD    HL,37ECH       ; Command Register
  426AH   CD CE 42      CALL  42CEH          ; Delay - get ready
  426DH   ED 53 EE 37   LD    (37EEH),DE     ; E into Sect Reg : D=Data Reg
  4271H   36 1B         LD    (HL),1BH       ; 27 = SEEK Track 0, 20ms step
  4273H   CD CE 42      CALL  42CEH          ; Delay
  4276H   36 88         LD    (HL),88H       ; 136 = load 1 sector IBM fmt
  4278H   11 EF 37      LD    DE,37EFH       ; DATA Register
  427BH   01 00 51      LD    BC,5100H       ; Buffer
  427EH   CD D7 42      CALL  42D7H          ; Delay
```

. . . . . .

This subroutine (4252H-42B8H) reads a sector from Disk into the sector buffer. In the 1st read HL is loaded with 51FFH at 4218H. When L is incremented its value is zero and the program falls through the RET NZ (4254H) to the sector buffer load.

The registers are alternated, the error count is set to 10, Drive 0 is selected, the Track number, Sector number and Error count are stored, and if the Sector count >= 10 then 10 is subtracted and Drive 0 again selected. The 9 here instead of 1 probably has a connotation of side.

A delay ensues to give the drive time to be ready, the sector number E is placed in the sector register and the the data register is zeroed. The command register receives a SEEK instruction and after a delay is asked to load a sector. DE is pointed to the data register, BC to the start of the sector buffer and there is another delay for the FDC to become ready.

MODEL I BOOT SECTOR

```
   Addr.    Code          Instruction          Function

1 4281H   7E            LD    A,(HL)     ; Status of Drive
  4282H   E6 83         AND   83H        ; leave Bits 0, 1, and 7
  4284H   E2 81 42      JP    PO,4281H   ; if Bit 7, Drive not ready
4 4287H   1A            LD    A,(DE)     ; Transfer Data to
  4288H   02            LD    (BC),A     ;   Sector Buffer
  4289H   03            INC   BC         ; move on in the Buffer
1 428AH   CB 4E         BIT   1,(HL)     ; If not busy
  428CH   C2 87 42      JP    NZ,4287H   ;  move more data
  428FH   CB 4E         BIT   1,(HL)     ; Wait till the
  4291H   C2 87 42      JP    NZ,4287H   ;   Status Reg
  4294H   CB 4E         BIT   1,(HL)     ;    has Data read
  4296H   20 EF         JR    NZ,4287H   ;     request
  4298H   CB 46         BIT   0,(HL)     ; if Drive is not busy
  429AH   28 08         JR    Z,42A4H    ;       go....
  429CH   CB 4E         BIT   1,(HL)     ; Test again for
  429EH   20 E7         JR    NZ,4287H   ;  data request
  42A0H   CB 7E         BIT   7,(HL)     ; Test if Drive is ready
  42A2H   28 E6         JR    Z,428AH    ; if so go try again
```

. . . . . .

This code reads data from the disk into the buffer. The Data Register (DE) contains the byte placed there by the FDC after the required instruction is sent to the Command Register (HL).

The loop 4281H to 4284H waits for the FDC to be ready. Bit 7 = not busy reading data from the disk. Bit 0 - ready and Bit 1 - not requesting data to be read.

When activated the controller installs a data byte in address 37EFH (DE) and sets Bit 1 of the status register (37ECH). It resets bit 1 when 37EFH is read. The data byte is transferred to the buffer address in Memory (BC).

The status register is read to see if another byte of data is ready (Bit 1). The command register is given a read command and if all bits except 0 and 1 are zero then no error exists.

The status checks from 428AH to 42A2H give the FDC time to fetch some more data and signal by the bits in the status register that it has done so and is ready for the data register to be read again.

If it fails all tests then there is either an error or the buffer has been fully loaded and control must pass to either the data transfer to memory routine or an error routine.

MODEL I – BOOT SECTOR

```
  Addr.   Code        Instruction          Function

1 42A4H   7E          LD    A,(HL)       ; Store status in A
  42A5H   36 00       LD    (HL),00H     ; Interrupt Command
  42A7H   C1          POP   BC           ; Restore original
  42A8H   D1          POP   DE           ;   values for next time
  42A9H   E6 FC       AND   0FCH         ; all but Bits 0 & 1
  42ABH   20 0C       JR    NZ,42B9H     ; if error/not ready Go
  42ADH   1C          INC   E            ; move sector # on
  42AEH   7B          LD    A,E          ;   if < 10 then
  42AFH   D6 0A       SUB   0AH          ;   go through
  42B1H   20 03       JR    NZ,42B6H     ;   else incr Track
  42B3H   14          INC   D            ;   & set Sector
  42B4H   1E 00       LD    E,00H        ;  to 0.
1 42B6H   D9          EXX                ; Back to other Registers
  42B7H   7E          LD    A,(HL)       ; contents of buffer into A
  42B8H   C9          RET                ;  & go back.

  ;  ..............   Test if all Errors used
  42B9H   CD D7 42    CALL  42D7H        ; Delay
  42BCH   36 0B       LD    (HL),0BH     ; Restore to Track 0
  42BEH   10 98       DJNZ  4258H        ; if B <> 1 start again
```

. . . . . .

When the sector is fully read or some error has disrupted the Input the FDC is given a stop (force interrupt) command. This allows the FDC to go into an inactive state and turns the drive off after a wind down delay.

The status is read and by eliminating bits 0 and 1 (42A9H) the error status is found. If an error exists the code after a delay tests to see if all 10 of the value of errors in B are used and if they are the program falls through to the 'ERROR' message.

If there is no error the sector number is incremented and tested as to whether it is 10. If it has then the sector number is zeroed and the track number is incremented. If not there is a relative jump across this code.

The registers used for the buffer-to-memory transfer are restored and the disk-to-buffer registers are stored as the primed ones. The first position in the sector buffer is loaded into the A register for analysis on return and the program returns to the code which made the sector buffer load call.

If there is an error control goes to the delay. If all errors are not used we go back for another try. If the error count B=0 the program falls through to error handling.

MODEL I BOOT SECTOR

```
  Addr.    Code        Instruction          Function

  ;          ERROR  handling
  42C0H   21 0D 42    LD    HL,42DDH    ; 'ERROR' ASCII

  ;        Error print and lock up routine
3 42C3H   7E          LD    A,(HL)      ; Read the byte
  42C4H   FE 03       CP    03H         ; if it is 3 go into
  42C6H   28 FB       JR    Z,42C3H     ;   an endless loop.
  42C8H   23          INC   HL          ; else read
  42C9H   CD 33 00    CALL  0033H       ;  & Print
  42CCH   18 F5       JR    42C3H       ;   more.

  ;          After a delay, test for busy
1 42CEH   CD D7 42    CALL  42D7H       ; Delay subroutine
1 42D1H   CB 46       BIT   0,(HL)      ; Test for busy
  42D3H   20 FC       JR    NZ,42D1H    ; if so wait
  42D5H   7E          LD    A,(HL)      ; Get Status value
  42D6H   C9          RET               ;   & Return

  ;       Delay subroutine
3 42D7H   3E 06       LD    A,06H       ; Delay
1 42D9H   3D          DEC   A           ;   just
  42DAH   20 FD       JR    NZ,42D9H    ;      50 Microseconds
  42DCH   C9          RET               ; and go back

  ;       ASCII  storage
1 42DDH - 42DEH   1C 1F                 ; ASCII - Home & Clear screen
  42DFH - 42E3H   45 52 52 4F 52        ; ASCII - ERROR
  42E4H   03                            ; EOR - terminator
1 42E5H - 42E6H   1C 1F                 ; ASCII - Home Cursor & Clear
  42E7H - 42ECH   4E 4F 20 53 59 53     ; ASCII -  NO SYS
  42EDH   03                            ; End of message.
  42EEH - 42FFH                         ; Garbage
```

. . . . . .

The delay loop loads A with 6 and decrements it to 0 with a delay of about 50 microseconds.

The routines calling 42CEH first go to the delay subroutine, then there is a FDC test for busy which holds up execution until the bit which holds the busy signal is not set.

The error handling routine prints the message (either NO SYS or ERROR), then when finished goes into an endless loop to lock up the computer.

MODEL III BOOT SECTOR

The Boot sector is loaded by the ROM by much the same method as the Model I excepting of course the use of Ports for FDC control instead of memory addresses.

The Boot Sector is loaded into a different address (4300H instead of 4200H) but the sector buffer remains at the same address.

The Boot sector disassembled :-

```
  Addr.   Code        Instruction          Function

  4300H   00          NOP                  ; No action
  4301H   FE 11       CP    11H            ; Recogn Code + DIR Sector
  4303H   F3          DI                   ; Disable Interrupts
  4304H   11 05 00    LD    DE,0005H       ; D = Track E = Sector
  4307H   D9          EXX                  ; Alternate

  ;         Buffer vector and stack setting
  4308H   31 E0 41    LD    SP,41E0H       ; Stack to Comm. Area
  430BH   21 FF 51    LD    HL,51FFH       ; Sector Buffer (5100-51FF)

  ;         Sector buffer load call
1 430EH   CD 45 43    CALL  4345H          ; If L+1 = 0 load Sector
  4311H   FE 20       CP    20H            ; If first byte >1FH- Error
  4313H   47          LD    B,A            ; B for DJNZ at 431FH & 4333H
  4314H   30 29       JR    NC,433FH       ; Error if > 1FH
  4316H   57          LD    D,A            ; If A > 2 & < 20H load to ROM

  ;         C = the number of bytes to load into the buffer
  4317H   CD 45 43    CALL  4345H          ; Get No. bytes to load
  431AH   4F          LD    C,A            ;        into C

  ;         E = the low byte of the address to load to in memory
  431BH   CD 45 43    CALL  4345H          ; Get LSB of Transfer Addr
  431EH   5F          LD    E,A            ;        into E
  431FH   10 12       DJNZ  4333H          ; If lst Byte =1 transfer
```

. . . . . .

This code performs the same function as the Model I code in that it sets up the control code reading and the Sector Buffer Address. This program uses the "INI" mnemonic to load (439CH) and otherwise is very similar to the Model I. Please refer to the Model I explanation as the flow of the program is the same.

MODEL III BOOT SECTOR

```
   Addr.    Code        Instruction          Function

   4321H  CD 45 43     CALL  4345H      ; Get MSB of Transfer Addr.

   ;            Sector buffer to memory transfer routine
   4324H  57           LD    D,A        ;   into D
   4325H  0D           DEC   C          ; Allow for 2 Bytes
   4326H  0D           DEC   C          ; of Transfer Address.
1  4327H  2C           INC   L          ; If end of Buffer
   4328H  CC 48 43     CALL  Z,4348H    ;  load another Sector
   432BH  7E           LD    A,(HL)     ; Load from Buffer
   432CH  12           LD    (DE),A     ;  to Transfer Address
   432DH  13           INC   DE         ; Transfer Addr moved on
1  432EH  0D           DEC   C          ; Count 1 less
   432FH  20 F6        JR    NZ,4327H   ; If more go back

   ;           When the byte count has been transferred ( C = 0 )
   4331H  18 DB        JR    430EH      ; Go load another area

   ;        Pass Control to Transfer Addr. or Error
1  4333H  10 F9        DJNZ  432EH      ; If 1st Byte > 2 load to ROM
   4335H  CD 45 43     CALL  4345H      ; MSB of Transfer Addr (4D00H)
   4338H  57           LD    D,A        ;  into D
   4339H  1A           LD    A,(DE)     ; Check if (4D00H) = A5H
   433AH  FE A5        CP    0A5H       ; If so set Z Flag
   433CH  13           INC   DE         ; Move to 4D01H
   433DH  D5           PUSH  DE         ; to Stack for RET
   433EH  C8           RET   Z          ; If = A5H Transfer to SYS0/SYS.

   ;      Error - Print " NO SYS " and lock up.
1  433FH  21 F0 43     LD    HL,43F0H   ; ASCII "NO SYS"
   4342H  C3 CC 43     JP    43CCH      ; To Error handling
```

. . . . . .

This code checks the 1st byte of the load, if > 2 and < 32 then "D" register contains that value. The second DJNZ at 4033H sends the program to the memory load routine and the code is written to ROM. This is used (e.g. 5) to store ASCII text identifying the Program on Disk.

If the 1st byte was 2, this is the code indicating the end of the program, it falls through to 4335H and gets the transfer MSB from the buffer and checks to see if the contents of this address is A5H. If so PUSH DE and RET passes control to the program just loaded (SYS0/SYS).

If the transfer address does not hold the required test byte A5H then the RET Z allows the error routine to execute.

MODEL III BOOT SECTOR

```
  Addr.   Code        Instruction          Function

  ;     Test for end of Sector Buffer
  ; Note:- H not incremented so Buffer stays in 5100H - 51FFH range
5 4345H  2C           INC    L            ; Move on in Buffer
  4346H  7E           LD     A,(HL)       ; Get Character
  4347H  C0           RET    NZ           ; If not end go back for more
1 4348H  D9           EXX                 ; else load another Sector
  4349H  06 0A        LD     B,0AH        ; Error count
1 434BH  2E 81        LD     L,81H        ; = Dbl Dens: Front Side: Trk 0
                                          ;   for Port F4 later
  434DH  D5           PUSH   DE           ; Store Track & Sector
  434EH  C5           PUSH   BC           ; Error Count & C = 0
  434FH  7B           LD     A,E          ; Sector No.
  4350H  D6 12        SUB    12H          ; If < 18 (Sectors per Track)
  4352H  38 03        JR     C,4357H      ;  jump over side swap
  4354H  5F           LD     E,A          ; Else Sector No to 0
  4355H  2E 91        LD     L,91H        ; Read Back side of disk
1 4357H  CD D7 43     CALL   4307H        ; Delay
  435AH  7D           LD     A,L          ; Command to
  435BH  D3 F4        OUT    (0F4H),A     ;         Select Port
  435DH  7A           LD     A,D          ; Track to
  435EH  D3 F3        OUT    (0F3H),A     ;         Track Port
  4360H  7B           LD     A,E          ; Sector to
  4361H  D3 F2        OUT    (0F2H),A     ;         Sector Port
  4363H  3E 1B        LD     A,1BH        ; Seek - Verify +20ms Step Rate
  4365H  D3 F0        OUT    (0F0H),A     ;     to Command Port
  4367H  CD D7 43     CALL   4307H        ; Delay
  436AH  7D           LD     A,L          ; Change Command to
  436BH  F6 40        OR     40H          ;     Wait States Enabled
  436DH  5F           LD     E,A          ; Store for 439AH
  436EH  16 02        LD     D,02H        ; Data Request for Port F0
```

. . . . . .

The code here sets up all the vectors and values for the disk-to-buffer load. First the disk to sector registers are made active (EXX), the error count is initialised and the current track and sector numbers are stored on the stack.

The sector number is tested for a value over the number of sectors per track. If this is so, the sector number is set to 0 and the back side select bit is set for the command to Port F4 at 435BH. This is what enables NEWDOS/80 to Boot a double sided Disk.

After a delay the FDC select, track and sector Ports are sent the required values. Then the value for the select Port is changed to allow the Wait States. The data request value for the Command register is stored in D.

MODEL III BOOT SECTOR

```
  Addr.    Code          Instruction          Function

  4370H  DB F0          IN    A,(0F0H)    ; Get Status
  4372H  ED 73 A6 43    LD    (43A6H),SP  ; Store Stack Pointer
  4376H  21 A2 43       LD    HL,43A2H    ; Set 4049H to contain
  4379H  22 4A 40       LD    (404AH),HL  ;   Jump to 43A2H (C3 A2 43)
  437CH  3E C3          LD    A,0C3H      ;   for Return to Program
  437EH  32 49 40       LD    (4049H),A   ;   from I/O Interrupt
  4381H  3E C0          LD    A,0C0H      ; Activate
  4383H  D3 E4          OUT   (0E4H),A    ;  Data Request
  4385H  3E 88          LD    A,88H       ; READ - 1 Sector(Bit7)
  4387H  D3 F0          OUT   (0F0H),A    ;        - IBM Format(Bit3)
  4389H  21 00 51       LD    HL,5100H    ; Sector Buffer
  438CH  01 F3 00       LD    BC,00F3H    ; B = 0 = 256 = Count:C = F3 =
DataReg.
  438FH  CD E2 43       CALL  43E2H       ;  Delay
1 4392H  DB F0          IN    A,(0F0H)    ; Status
  4394H  A2             AND   D           ; Mask all but Data Request
  4395H  28 F8          JR    Z,4392H     ; Wait for DRQ
  4397H  ED A2          INI               ; Get a byte

  ;      The INI instruction reads 1 byte from disk via Port (C),
  ;      Decrements Count (B), Transfers Byte to (HL) + Incr HL

  4399H  7B             LD    A,E         ;  Current State to
1 439AH  D3 F4          OUT   (0F4H),A    ;   to SELECT Register
  439CH  ED A2          INI               ; Load 1 more Byte
  439EH  20 FA          JR    NZ,439AH    ; If B <> 0 load more
1 43A0H  18 FE          JR    43A0H       ; Wait for Interrupt
```

. . . . . .

The FDC is activated by a request. The stack pointer is loaded into the address to restore its current level after disk I/O and the I/O interrupt vector is initialised with the address to restore the flow of the program after I/O has been completed.

The Floppy Disk Controller is sent a data request and is notified what Track, Sector and other conditions are required. The sector buffer address is held in HL, the count of the number of bytes to load is placed in B and the C register (used by Port instructions) is set to Port F3 the data register.

After a delay the code "INI" reads 1 byte at a time for as many bytes as are in "B" (256). When the Sector is fully read there is an I/O Interrupt and the program jumps to 43A2H from the interrupt processor vector at 4049H (see the code at 4376H). When the full sector of 256 bytes has been loaded, the CPU locks up till an interrupt.

MODEL III BOOT SECTOR

```
  Addr.    Code        Instruction          Function

1 43A2H   AF          XOR   A            ; Clear A
  43A3H   D3 E4       OUT   (0E4H),A     ; Cancel Data Request

  ;          This location was loaded with the stack address by 4372H
1443A5H   31 00 00    LD    SP,0000H     ; Restore stack pointer
  43A8H   DB F0       IN    A,(0F0H)     ; Get Status

  ;          Test for errors - Test for Jump in 43B2H
  43AAH   E6 FC       AND   0FCH         ; Test(Ready+Error) Set Z Flag
  43ACH   3E D0       LD    A,0D0H       ; Force
  43AEH   D3 F0       OUT   (0F0H),A     ;    Interrupt
  43B0H   C1          POP   BC           ; Restore Error Count (B),(C),
  43B1H   D1          POP   DE           ;  Track (D) & Sector (E)
  43B2H   20 0C       JR    NZ,43C0H     ; If test fails try again
  43B4H   1C          INC   E            ; Next Sector

  ;     On a double sided Disk 43B7H is 24H or 36 Sectors per track
  43B5H   7B          LD    A,E          ; Test if
  43B6H   D6 12       SUB   12H          ;  = 18
  43B8H   20 03       JR    NZ,43BDH     ; Jump over if not else
  43BAH   14          INC   D            ;    Track on 1
  43BBH   1E 00       LD    E,00H        ;    Sector to 0
1 43BDH   D9          EXX                ; Exchange
  43BEH   7E          LD    A,(HL)       ; 1st byte in buffer into A
  43BFH   C9          RET                ; Go back to caller
```

After a sector load, the data request is cancelled and the stack pointer is restored to the address it had before I/O and the FDC is sent an interrupt to reset it to an inactive state.

The original error count (B), the data register Port (C), the track (D) and sector (E) numbers are retrieved from the stack.

Then the FDC is tested for error, and if an error exists control is passed by the test at 43B2H to the routine which decrements the error count and tests if all errors used. If all errors are used the program goes to the error handling routine.

The sector number is incremented and tested to see if it exceeds the sector per track value. If so the sector number is zeroed and the track number incremented. Then the registers are alternated , the start of the sector buffer is read and the program returns to the calling code.

MODEL III BOOT SECTOR

```
  Addr.   Code        Instruction          Function

  ;   If not Ready or Error, Delay, Seek, Decrement Error count & RET
1 43C0H   CD E2 43    CALL   43E2H      ; Delay
  43C3H   3E 08       LD     A,08H      ; SEEK to
  43C5H   D3 F0       OUT    (0F0H),A   ;  Command Register
  43C7H   10 82       DJNZ   434BH      ; If all Errors used fall thru
  43C9H   21 E8 43    LD     HL,43E8H   ; ASCII "ERROR"

  ;   Error handling
3 43CCH   7E          LD     A,(HL)     ; Load it
  43CDH   FE 03       CP     03H        ; If terminator
  43CFH   28 FB       JR     Z,43CCH    ;  go into endless loop
  43D1H   23          INC    HL         ;  else move to next letter
  43D2H   CD 33 00    CALL   0033H      ; Print it
  43D5H   18 F5       JR     43CCH      ;  & go back for more

  ;   Delay and check Status
2 43D7H   C0 E2 43    CALL   43E2H      ; Delay
1 43DAH   DB F0       IN     A,(0F0H)   ; Status
  44DCH   0F          RRCA              ; Test for Bit 0
  43DDH   38 FB       JR     C,43DAH    ;  if Busy wait
  43DFH   DB F0       IN     A,(0F0H)   ;  else get Status
  43E1H   C9          RET               ;  and return

  ;   Delay for 41 Microseconds
3 43E2H   3E 08       LD     A,08H      ; Delay count
1 43E4H   3D          DEC    A          ;  1 less
  43E5H   20 FD       JR     NZ,43E4H   ; Go if Zero
  43E7H   C9          RET               ;  back to caller

  ;   ASCII error messages
1 43E8H - 43EFH   ; Clear Screen + ERROR  + Terminator
1 43F0H - 43F9H   ; Clear Screen + NO SYS + Terminator
  43FAH - 43FFH   ; Irrelevant - not used.
```

. . . . . .

A delay, a seek command is sent to the FDC and if all errors are not used the program returns to try again for a successful read of the sector. If all errors are used the error handling routine takes over and locks up the computer.

: :

TRS-80 INFORMATION

CHAPTER 5

VARIOUS LISTINGS

CHARACTER SETS IN THE MODEL III

The characters stored in the Model I computer for values of 192 to 255 are the Space Compression Codes (SCC). This is also the case for the Model III. However, as well there are two more sets available in the Model III, 64 symbols and 64 Katakana characters. Also to POKE values 0 to 31 onto the screen gives another set of symbols.

The control Byte for the change from SCC's to Symbols is 4024H (16420). If this byte contains 0, the SCC's are enabled. If any other value then the Symbols are enabled. Software control is toggled by PRINT CHR$(21).

The software action PRINT CHR$(22) toggles the Symbols and the Katakana characters. This is controlled by Bit 3 of the Byte at 4210H (16912), which is the Flag Byte for Port ECH (236), as shown below.

```
Bit 0 - ON - Clock enable        - OFF - Clock disable
Bit 1 - ON - Cassette Motor ON - OFF - Motor OFF
Bit 2 - ON - 32 Char Mode        - OFF - 64 Char Mode
Bit 3 - ON - Symbols enabled     - OFF - Katakana enabled
```

As with the 32 character mode, PRINT CHR$(22) changes the format of the screen immediately from Symbols to Katakana or vice versa.

From Basic OUT 236,8 is not permanent as the computer continually updates the Port from 4210H. To obtain the Symbols from Basic command mode, use POKE 16420,255 and POKE 16912, PEEK(16912) OR 8.

Interesting:- Among the Japanese Katakana characters is a dot (CHR$(197)) which is on the centre of the line (not at the bottom) which is the right place for a decimal point.

Also to finish off the hand in CHR$(244,245 & 246) use CHR$(143 or 191+159 or 191 or 252) before them and it will look like you also have a sleeve which is the idea in the classic sign.

The word tilde (opposite) is pronounced 'teelda'.

No	Symbol	No	Symbol
0	Space	16	N tilde - Spanish
1	English Pound	17	O - Nordic
2	Vertical bar - Divisor	18	Phi Cap - Greek
3	e acute - French	19	O tilde - Spanish
4	u umlaut - German	20	B extended low bar
5	A - Nordic	21	U umlaut - German
6	angle - Negate	22	o tilde - Spanish
7	O umlaut - German	23	dollar sign reversed
8	crossed O - Nordic	24	a umlaut - German
9	u grave - French	25	a grave - French
10	n tilde - Spanish	26	a under dot - Nordic
11	grave	27	Section
12	a under bar - Spanish	28	E acute - French
13	R under bar - Spanish	29	AE diphthong
14	A umlaut - German	30	C cedilla - French
15	A tilde - Spanish	31	Alternation (tilde)
192	Spades	224	Omega Cap
193	Hearts	225	Square Root
194	Diamonds	226	Divide
195	Clubs	227	Sigma Cap
196	Smiling Face	228	Approximates
197	Frowning Face	229	Delta Cap
198	Right Leg	230	Integral
199	Left Leg	231	Not Equal
200	Alpha	232	Lightning
201	Beta	233	Per cent (bar)
202	Gamma	234	Alpha Cap
203	Delta	235	Infinity
204	Epsilon	236	Tick (O.K.)
205	Zeta	237	Summation
206	Ita	238	a nice border
207	Theta	239	Copyright
208	Iota	240	Satellite - Asterisk
209	Kappa	241	Paragraph
210	Lambda	242	Cent Sign
211	Mu	243	Registered
212	Nu	244	: Three Parts
213	Xi	245	: of Pointing
214	Omicron	246	: Hand
215	Pi	247	Chemist's Sign
216	Rho	248	Care of
217	Tsi	249	Female
218	Tau	250	Male
219	Upsilon	251	a cent (high)
220	Phi	252	Inverted question mark
221	Chi	253	Man
222	Psi	254	Woman
223	Omega	255	Bishop's Mitre

ASCII CONTROL CODES

This part gives a ready reference for the values of ASCII Control Codes, all the ASCII Symbols and Letters, and the Reserved Word Codes and their Routine addresses. On the first page, the effect of the control codes when PRINTed as a CHR$ is shown as well as the standard set of Printer Control Codes. N.B. 7FH (127) also has a Printer Code.
The 3 before the CHR$ means the function is peculiar to the Model III.

Binary	HEX	DEC	Print CHR$ Function	Std Printer Codes
00000000	00	0	Null	NUL: Null
00000001	00	1	Break key	SOH: Start of Heading
00000010	02	2	Not used	STX: Start of Text
00000011	03	3	Not used	ETX: End of Text
00000100	04	4	Not used	EOT: End Transmission
00000101	05	5	Not used	ENQ: Enquiry
00000110	06	6	Not used	ACK: Acknowledge
00000111	07	7	Not used	BEL: Bell
00001000	08	8	Backspace & erase	BS : Backspace
00001001	09	9	No effect	HT : Horizontal Tab
00001010	0A	10	LF + CR & erase line	LF : Line Feed
00001011	0B	11	No effect	VT : Vertical Tab
00001100	0C	12	No effect	FF : Form Feed
00001101	0D	13	LF + CR & erase line	CR : Carriage Return
00001110	0E	14	Turn on Cursor	SI : Shift In
00001111	0F	15	Turn off Cursor	SO : Shift Out
00010000	10	16	No effect	DLE: Data Link Escape
00010001	11	17	No effect	DC1: Direct Control 1
00010010	12	18	No effect	DC2: Direct Control 2
00010011	13	19	No effect	DC3: Direct Control 3
00010100	14	20	No effect	DC4: Direct Control 4
00010101	15	21	3-Toggles SCC & Symbols	NAK: Neg acknowledge
00010110	16	22	3-Toggles Symb & Kata.	SYN: Synchronous Idle
00010111	17	23	Double size Chars	ETB: End Transm block
00011000	18	24	Backspace no erase	CAN: Cancel
00011001	19	25	Advance Cur no erase	EM : End Medium
00011010	1A	26	Move DOWN 1 line	SUB: Substitute
00011011	1B	27	Move UP 1 line	ESC: Escape
00011100	1C	28	Cursor to 0/0	FS : Form Separator
00011101	1D	29	Curs start line no erase	GS : Group Separator
00011110	1E	30	Erase to end line	RS : Record Separator
00011111	1F	31	Erase to end frame	US : Unit Separator

KEYABLE CHARACTERS

The range of ASCII values on this page cover all the characters that can be accessed from the keyboard. Looking at the columns, the only difference between upper and lower case is Bit 5 (01? . . . . .) so changing case only involves Bit 5 (to reset AND with 5FH, to set OR with 20H).

Symbols & Numbers				Upper Case Letters				Lower Case Letters			
Binary	Hex	Dec	CHR$	Binary	Hex	Dec	CHR$	Binary	Hex	Dec	CHR$
00100000	20	32	Space	01000000	40	64	@	01100000	60	96	`
00100001	21	33	!	01000001	41	65	A	01100001	61	97	a
00100010	22	34	"	01000010	42	66	B	01100010	62	98	b
00100011	23	35	#	01000011	43	67	C	01100011	63	99	c
00100100	24	36	$	01000100	44	68	D	01100100	64	100	d
00100101	25	37	%	01000101	45	69	E	01100101	65	101	e
00100110	26	38	&	01000110	46	70	F	01100110	66	102	f
00100111	27	39	'	01000111	47	71	G	01100111	67	103	g
00101000	28	40	(	01001000	48	72	H	01101000	68	104	h
00101001	29	41	)	01001001	49	73	I	01101001	69	105	i
00101010	2A	42	*	01001010	4A	74	J	01101010	6A	106	j
00101011	2B	43	+	01001011	4B	75	K	01101011	6B	107	k
00101100	2C	44	,	01001100	4C	76	L	01101100	6C	108	l
00101101	2D	45	-	01001101	4D	77	M	01101101	6D	109	m
00101110	2E	46	.	01001110	4E	78	N	01101110	6E	110	n
00101111	2F	47	/	01001111	4F	79	O	01101111	6F	111	o
00110000	30	48	0	01010000	50	80	P	01110000	70	112	p
00110001	31	49	1	01010001	51	81	Q	01110001	71	113	q
00110010	32	50	2	01010010	52	82	R	01110010	72	114	r
00110011	33	51	3	01010011	53	83	S	01110011	73	115	s
00110100	34	52	4	01010100	54	84	T	01110100	74	116	t
00110101	35	53	5	01010101	55	85	U	01110101	75	117	u
00110110	36	54	6	01010110	56	86	V	01110110	76	118	v
00110111	37	55	7	01010111	57	87	W	01110111	77	119	w
00111000	38	56	8	01011000	58	88	X	01111000	78	120	x
00111001	39	57	9	01011001	59	89	Y	01111001	79	121	y
00111010	3A	58	:	01011010	5A	90	Z	01111010	7A	122	z
00111011	3B	59	;	01011011	5B	91	UpA	01111011	7B	123	{
00111100	3C	60	<	01011100	5C	92	DnA	01111100	7C	124	|
00111101	3D	61	=	01011101	5D	93	LfA	01111101	7D	125	}
00111110	3E	62	>	01011110	5E	94	RtA	01111110	7E	126	~
00111111	3F	63	?	01011111	5F	95	Cur	01111111	7F	127	DEL

RESERVED WORDS & GRAPHICS CHARACTERS

Bit 0	Bit 1
Bit 2	Bit 3
Bit 4	Bit 5

This Page has the values for the Graphics Codes for PRINT CHR$(128-191). If the Pixels to be set are known then the Value to use is very easily figured. Look at the chart on the left. Work out which Bits are to be set. Then see the value from the Binary when these Bits are set and you have it. Conversely Graphics can be worked from the bits.

Binary	Hex	Dec	Res.Word	Vector	Binary	Hex	Dec	Res.Word	Vector
10000000	80	128	END	1DAEH	10100000	A0	160	OUT	2AFBH
10000001	81	129	FOR	1CA1H	10100001	A1	161	ON	1F6CH
10000010	82	130	RESET	0138H	10100010	A2	162	OPEN	4179H
10000011	83	131	SET	0135H	10100011	A3	163	FIELD	417CH
10000100	84	132	CLS	01C9H	10100100	A4	164	GET	417FH
10000101	85	133	CMD	4173H	10100101	A5	165	PUT	4182H
10000110	86	134	RANDOM	01D3H	10100110	A6	166	CLOSE	4185H
10000111	87	135	NEXT	22B6H	10100111	A7	167	LOAD	4188H
10001000	88	136	DATA	1F05H	10101000	A8	168	MERGE	418BH
10001001	89	137	INPUT	219AH	10101001	A9	169	NAME	418EH
10001010	8A	138	DIM	2608H	10101010	AA	170	KILL	4191H
10001011	8B	139	READ	21EFH	10101011	AB	171	LSET	4197H
10001100	8C	140	LET	1F21H	10101100	AC	172	RSET	419AH
10001101	8D	141	GOTO	1EC2H	10101101	AD	173	SAVE	41A0H
10001110	8E	142	RUN	1EA3H	10101110	AE	174	SYSTEM	02B2H
10001111	8F	143	IF	2039H	10101111	AF	175	LPRINT	2067H
10010000	90	144	RESTORE	1D91H	10110000	B0	176	DEF	415BH
10010001	91	145	GOSUB	1EB1H	10110001	B1	177	POKE	2CB1H
10010010	92	146	RETURN	1EDEH	10110010	B2	178	PRINT	206FH
10010011	93	147	REM	1F07H	10110011	B3	179	CONT	1DE4H
10010100	94	148	STOP	1DA9H	10110100	B4	180	LIST	2B2EH
10010101	95	149	ELSE	1F07H	10110101	B5	181	LLIST	2B29H
10010110	96	150	TRON	1DF7H	10110110	B6	182	DELETE	2BC6H
10010111	97	151	TROFF	1DF8H	10110111	B7	183	AUTO	2008H
10011000	98	152	DEFSTR	1E00H	10111000	B8	184	CLEAR	1E7AH
10011001	99	153	DEFINT	1E03H	10111001	B9	185	CLOAD	2C1FH
10011010	9A	154	DEFSNG	1E06H	10111010	BA	186	CSAVE	2BF5H
10011011	9B	155	DEFDBL	1E09H	10111011	BB	187	NEW	1B49H
10011100	9C	156	LINE	41A3H	10111100	BC	188	TAB(	2137H
10011101	9D	157	EDIT	2E60H	10111101	BD	189	TO	= FOR
10011110	9E	158	ERROR	1FF4H	10111110	BE	190	FN	4155H
10011111	9F	159	RESUME	1FAFH	10111111	BF	191	USING	2CBDH

RESERVED WORDS & SPACE COMP. CODES

See Page 24 for actual DOS Addresses for the Reserved Word Routines.
The values on this Page have another Function when PRINTed.
The Space Compression Codes have the value of from 192 to 255.
Thus PRINT CHR$(193) will print 1 space and CHR$(255) will print 63 spaces.
Pages 54 & 55 explain and display the Model III character set symbols.

Binary	Hex	Dec	ResWord	Vector	Binary	Hex	Dec	ResWord	Vector
11000000	C0	192	VARPTR	24EBH	11100000	E0	224	EXP	1439H
11000001	C1	193	USR	27FEH	11100001	E1	225	COS	1541H
11000010	C2	194	ERL	24CFH	11100010	E2	226	SIN	1457H
11000011	C3	195	ERR	24DDH	11100011	E3	227	TAN	15A8H
11000100	C4	196	STRING$	2A2FH	11100100	E4	228	ATN	15BDH
11000101	C5	197	INSTR	419DH	11100101	E5	229	PEEK	2CAAH
11000110	C6	198	POINT	0132H	11100110	E6	230	CVI	4152H
11000111	C7	199	TIME$	4176H	11100111	E7	231	CVS	4158H
11001000	C8	200	MEM	27C9H	11101000	E8	232	CVD	415EH
11001001	C9	201	INKEY$	4099H	11101001	E9	233	EOF	4161H
11001010	CA	202	THEN =	IF	11101010	EA	234	LOC	4164H
11001011	CB	203	NOT =	Arithm	11101011	EB	235	LOF	4167H
11001100	CC	204	STEP	2B01H	11101100	EC	236	MKI$	416AH
11001101	CD	205	+ =	Arithm	11101101	ED	237	MKS$	416DH
11001110	CE	206	- =	Arithm	11101110	EE	238	MKD$	4170H
11001111	CF	207	* =	Arithm	11101111	EF	239	CINT	0A7FH
11010000	D0	208	/ =	Arithm	11110000	F0	240	CSNG	0AB1H
11010001	D1	209	UpArw =	Arithm	11110001	F1	241	CDBL	0ADBH
11010010	D2	210	AND	25FDH	11110010	F2	242	FIX	0B26H
11010011	D3	211	OR	25F7H	11110011	F3	243	LEN	2A03H
11010100	D4	212	> =	Arithm	11110100	F4	244	STR$	2836H
11010101	D5	213	= =	Arithm	11110101	F5	245	VAL	2AC5H
11010110	D6	214	< =	Arithm	11110110	F6	246	ASC	2A0FH
11010111	D7	215	SGN	098AH	11110111	F7	247	CHR$	2A1FH
11011000	D8	216	INT	0B37H	11111000	F8	248	LEFT$	2A61H
11011001	D9	217	ABS	0977H	11111001	F9	249	RIGHT$	2A91H
11011010	DA	218	FRE	27D4H	11111010	FA	250	MID$	2A9AH
11011011	DB	219	INP	2AEFH	11111011	FB	251	'	(REM)
11011100	DC	220	POS	27F5H	11111100	FC	252	No Res.	Word
11011101	DD	221	SQR	13E7H	11111101	FD	253	No Res.	Word
11011110	DE	222	RND	14C9H	11111110	FE	254	No Res.	Word
11011111	DF	223	LOG	0809H	11111111	FF	255	No Res.	Word

VARIABLE STORAGE

There are 4 types of Variables to be stored :-

1) A simple numeric Variable
2) An Array numeric Variable
3) A simple String Variable
4) An Array String Variable

As well there are 3 types of each numeric variable :-

%- An Integer value which takes up 2 Bytes (No.Type Flag = 2)
!- A Single Precision value takes up 4 Bytes (No.Type Flag = 4)
#- A Double Precision value takes up 8 Bytes (No.Type Flag = 8)

The charts below set out the system used by Basic for this storage and the meaning of each byte of storage. The bytes marked with 3 asterisks are the address Basic returns in the VARPTR function. It should be noted that the first element of an array always has element number 0. Thus in an array DIM VA(4) there are 5 elements and in an array not DIMmed there is a 10 default which means 11 elements.

The memory taken up by simple Numerics is fixed at 3 plus the NTF value i.e. Int = 5, SP = 7, DP = 11 bytes.

Array usage is the No. of Dimensions * 2 + NTF * No. of elements + 6 bytes overhead. e.g. VA!(5,2,3) uses 3*2 + 4*(6*3*4) = 294 + 6. Bytes 4 & 5 hold the offset + 1 from Byte 6 to the NTF of the next variable, e.g. 295 in the case of VA!(5,2,3).

The example below uses VA for simple and Array variables. All variables have a value of 256, the Array is DIMmed to VA(0,0).

SIMPLE NUMERIC

Byte	Type Char		Int	SP	DP
1	No.Type Flag		2	4	8
2	Name 2nd Char		A	A	A
3	" 1st "		V	V	V
4	Data	***	0	0	0
5	Data		1	0	0
6	Data			0	0
7	Data			137	0
8	Data				0
9	Data				0
10	Data				0
11	Data				137

ARRAY NUMERIC

Byte	Type Char		Int	SP	DP
1	No.Type Flag		2	4	8
2	Name 2nd Char		A	A	A
3	" 1st "		V	V	V
4	Offset to	LSB	12	14	18
5	next var.	MSB	0	0	0
6	Dimensions		2	2	2
7	No. Elements in		1	1	1
8	Dimension 1.		0	0	0
9	No.Elements in		1	1	1
10	Dimension 2.		0	0	0
11	Data	***	0	0	0
12	Data		1	0	0
13	Data			0	0
14	Data			137	0
15	Data				0
16	Data				0
17	Data				0
18	Data				137

VARIABLE STORAGE

In String storage (Number Type Flag = 3) the VARPTR points to the length of the String. This is followed by the actual address of the String up in String storage or wherever it may be. Though String storage is used from the top down the letters in the simple String VA$ are stored and read from the bottom up.
That is, Bytes 5 & 6 point to "A" of "ABC".

Simple String storage uses 6 plus the number of bytes in the String. i.e. "ABCDEF" takes up 6 + 6 = 12.
In a String Array, the no. of bytes used is 6 plus No. Dimensions * 2 plus the No. Elements * 3 plus total length of all Elements.

The example below uses the variable VA$ for a String Variable and a String Array variable DIMmed VA$(1,1). VA$ and all Elements of VA$ Array are given the characters "ABC".

The Computer stores in bytes 4 & 5 the offset to the next variable (17 bytes). This is actually 5 less than the storage of information about the Array and that does not include any Data storage. To obtain the memory taken up by the use of this array add PEEK(Byte4)+PEEK(Byte5)*256+5+(total length of all elements).

SIMPLE STRING

Byte	Type Char		Char
1	No.Type Flag		3
2	Name 2nd Char		A
3	" 1st "		V
4	Len(VA$	***	3
5	Addr of VA$		LSB
6	Addr of VA$		MSB
	..... .		
	At Addr in bytes 5 & 6		
	from low Addr up		
1	Data		A
2	Data		B
3	Data		C

ARRAY STRING

Byte	Type Char		Char
1	No.Type Flag		3
2	Name 2nd Char		A
3	" 1st "		V
4	Offset to next array	LSB	17
5	Offset to next array	MSB	0
6	No. of Dimensions		2
7	No. Element	LSB	2
8	in Dimension 1	MSB	0
9	No. Element	LSB	2
10	in Dimension 2	MSB	0
11	LEN(VA$(0,0))	***	3
12	Addr of VA$(0,0)		LSB
13	Addr " "		MSB
14	LEN(VA$(1,0))	***	3
15	Addr of VA$(1,0)		LSB
16	Addr " "		MSB
17	LEN(VA$(0,1))	***	3
18	Addr of VA$(0,1)		LSB
19	Addr " "		MSB
20	LEN(VA$(1,1))	***	3
21	Addr of VA$(1,1)		LSB
22	Addr " "		MSB

ERROR MESSAGES

Dec	Hex	NEWDOS/80 Vers.2	TRSDOS
00	00	No error	No error
01	01	Bad File Data	CRC Error in Disk I/O
02	02	Seek error during Read	Drive not in System
03	03	Lost Data during Read	Lost Data in Disk I/O
04	04	Parity error during Read	CRC error in Disk I/O
05	05	Data Record not found	Disk Sector not found
06	06	Tried Read Locked/Del Rec	Drive Hardware fault
07	07	Tried Read System Record	Undefined error code
08	08	Device not available	Drive not ready
09	09	Undefined error code	Illegal I/O attempt
10	0A	Seek error during Write	Parameter not found
11	0B	Lost Data during Write	Illegal Parameter
12	0C	Parity error during Write	Time out on Drive
13	0D	Data not found	I/O to Non-System Disk
14	0E	Write fault on Drive	Write fault on Disk I/O
15	0F	Write protected Disk	Write protected Disk
16	10	Device not available	Illegal File Number
17	11	Directory Read error	Directory Read error
18	12	Directory Write error	Directory Write error
19	13	Illegal File name	Invalid File name
20	14	Track Number too high	GAT Read error
21	15	Illegal function DOS call	GAT Write error
22	16	Undefined error code	HIT Read error
23	17	Undefined error code	HIT Write error
24	18	File not in Directory	File not found
25	19	File access denied	File access denied-Password
26	1A	Directory space full	Directory space full
27	1B	Disk space full	Disk space full
28	1C	End of File encountered	Attempt Read past EOF
29	1D	Past end of File	Attempt Read outside File
30	1E	Directory Full	No more Extents Available
31	1F	Program not found	Program not found
32	20	Illegal or Missing Drive	Invalid Drive number
33	21	No Device space available	Undefined error code
34	22	Load File Format error	Use non-program File
35	23	Memory fault	Memory fault during Load
36	24	Tried to load ROM	Undefined error code
37	25	Illegal access Prot. File	File acc. denied-Password
38	26	File not Open	I/O attempt to Unopen File
39	27	Illegal Init Data on Disk	Invalid Command Parameter
40	28	Illegal Track count	File already in Directory
41	29	Illegal File Number	Open File already Open
42	2A	Illegal DOS function	
43	2B	Illegal Chaining function	
44	2C	Bad Directory data	End TRSDOS Errors

Dec	Hex	NEWDOS/80 Cont
45	2D	Bad FCB data
46	2E	System Program not found
47	2F	Bad Parameter(s)
48	30	Bad Filespec
49	31	Wrong Disk Record type
50	32	Boot Read error
51	33	DOS Fatal error
52	34	Illegal Keyword/Sep./Term.
53	35	File already exists
54	36	Command too long
55	37	Disk access denied
56	38	Illegal Mini-DOS Function
57	39	Op/Prog/Parameter terminated
58	3A	Data compare mismatch
59	3B	Insufficient Memory
60	3C	Incompatible Drive/Disk
61	3D	ASE=N Can't extend File
62	3E	Can't extend File via Read

Level 2	ROM Errors		Routine
1	NF	Next without For	00-199DH
2	SN	Syntax Error	02-1997H
3	RG	Return w'out Gosub	04-1EECH
4	OD	Out of Data	06-2212H
5	FC	Illeg. Funct. Call	08-1E4AH
6	OV	Overflow	0A-07B2H
7	OM	Out of Memory	0C-197AH
8	UL	Undefined Line	0E-1ED9H
9	BS	Subscript out range	10-273DH
10	DD	Redimensioned Array	12-2733H
11	/0	Division by Zero	14-199AH
12	ID	Illegal Direct	16-2831H
13	TM	Type Mismatch	18-0AF6H
14	OS	Out of String Space	1A-28DBH
15	LS	String too long	1C-29A3H
16	ST	$ form. too complex	1E-28A1H
17	CN	Cannot continue	20-1DE9H
18	NR	No Resume	22-198AH
19	RW	Resume w'out Error	24-19A0H
20	UE	Unprintable Error	26-2003H
21	MO	Missing Operand	28-24A0H
22	FD	Bad File Data	2A-218AH
23	L3	Disk Basic only	2C-012DH

No	Disk Basic
* 51	Field overflow
* 52	Internal error
* 53	Bad File Number
* 54	File not found
* 55	Bad File mode
56	File already Open
* 58	DOS error
59	File already exists
* 62	Disk full
* 63	Input past end
* 64	Bad Record number
* 65	Bad File name
66	Mode mismatch
* 67	Direct Stmt in File
* 68	Too many Files
* 69	Disk Write protected
* 70	File access denied
71	SEQ number overflow
72	Record overflow
73	Illegal to extend File
75	Prev. displayed error
76	Can't process Line #
77	Bad File type
78	IGEL Syntax error
79	IGEL item Syntax error
80	Bad/Illeg/Miss IGEL Pref
82	Bad Record length
83	Stmt uses 2 File names
84	Bad File position Param

The Disk Basic errors marked with "*" are identical in TRSDOS and are the only errors given by TRSDOS.

SYS4/SYS is the NEWDOS/80 DOS Error Message Module.

SYS13/SYS is the NEWDOS/80 Disk Basic and ROM Basic Error Module.

NEWDOS/80 SYSTEM COMMAND OPTIONS

PDRIVE and SYSTEM parameters are stored on the disk in Drive Relative Sector 2. On the right of the SYSTEM option and the PDRIVE parameter lists, a column shows which byte in this sector holds the information. Some parameters are stored in bits. These are a simple choice of Y (set) or N (reset). The bit involved is also shown in these cases.

AA	=	Enable Password checking "Y" or not "N"	F0-7
AB	=	Force RUN-only condition "Y" or not "N"	F0-6
AC	=	Enable Debounce "Y" or not "N" (requires AJ=Y) Model I only	F8-7
AD	=	Enable 'JKL' Screen to Printer "Y" or not "N"	F8-6
AE	=	Enable '123' Debug entry "Y" or not "N" (requires AB=N)	F8-5
AF	=	Enable 'DFG' Mini-DOS entry "Y" or not "N"(requires AB=N)	F8-4
AG	=	Enable Break Key "Y" or not "N" (requires AB=N)	F0-5
AI	=	Lower Case Mod installed "Y" or not "N" (Model I only)	F0-4
AJ	=	Enable DOS Keyboard intercept routine "Y" or not "N"	F8-1
AL	=	Number of Drives in the System (1 to 4)	A0
AM	=	Number of Read attempts before Error (normally 10)	A6
AN	=	Default Drive number for DIR command (0-3)	A2
AO	=	Lowest Drive No for new File creation if not specified	A3
AP	=	Himem value installed on Boot	D0,D1
AQ	=	Enable Clear key to function "Y" or not "N"	F8-2
AR	=	Allow Full(5) or CBF(6) Copy without Pswrd "Y" or not"N"	F0-1
AS	=	Force Basic text input to Upper Case "Y" or not "N"	F0-0
AT	=	Allow input requests from Chain File "Y" or not "N"	F1-7
AU	=	Enable Keyboard repeat "Y" or not "N"	F8-0
AV	=	No. 25ms intervals before 1st key repeat (requires AU=Y)	A7
AW	=	Number of Write with Verify attempts before Error	A1
AX	=	Highest ASCII Printable Character for Printer (U/C only = 5AH(90) or L/C = 7AH(122))	A8
AY	=	Require Date/Time entry at cold start "Y" or not "N"	F9-7
AZ	=	Require Date/Time on Reset "Y" or use previous values "N"	F9-6
BA	=	Force ROUTE DO,NL (no Video output) at Reset "Y"; not"N"	F9-5
BB	=	Clock is 50cps "Y" or 60cps "N" (Model III only)	F9-4
BC	=	Permit Chaining pause or cancel "Y" or not "N" (req.AB=N)	F1-6
BD	=	Permit "AUTO" negate at reset with Enter key "Y"; not "N"	F9-3
BE	=	Enable "R" as repeat DOS command "Y" or not "N"	F1-5
BF	=	Force LCDVR,N at reset "Y" or not "N" (Model I only)	F9-2
BG	=	Force LC,Y at reset "Y" or not "N" (req. BF=Y if Mod I)	F9-1
BH	=	Enable Blinking Cursor "Y" or not "N"	F9-0
BI	=	ASCII value of cursor char (Mod I-0 = 95:Mod III-0 = 176)	A5
BJ	=	Timing loop multiplier (for speed-up mods)	A9
BK	=	Enable WRDIRP & Dircheck W & C functions "Y" or not "N"	F1-4
BM	=	Follow Format by Verify "Y"; not "N" (eliminated by Zap)	F1-3
BN	=	Directory readable Mod III NEWDOS "Y" or Mod I TRSDOS "N"	FA-7

AH, AK, BL, BM, BO-ZZ are not used.

PDRIVE PARAMETERS

```
TI ......... Type of Interface                stored in bytes 02,07,0D,0E
       A - Standard Tandy Interface                          02 & 0D both=0
       B - Omikron mapper Interface (Mod I)                  02-3,02-2,0D-0
       C - Percom doubler Interface (Mod I)                  02-4,     ,0D-1
       D - Apparat Disk Controller  (Mod III)                02-3,     ,0D-2
       E - LNW type Interface (Mod I)                        02-4,02-2,0D-3
   ......... Special conditions = Second letter (2or3 may be used)
       H - Head settle delay enabled (8 inch Drives)              02-7,0D-7
       I - Sector 1 = lowest Sector on each Track                 02-3,07-4
       J - Track 1 = lowest Track on the Diskette                 07-1,0E-1
       K - Track 0 formatted in opp dens to the rest 02-2+6,07-1,0E-2
       L - 2 steps between Trks (Read 40Trk on 80TrkDr)           07-2,0E-3
       M - TRSDOS ModI 2.3B or higher or ModIII read.             02-5,0E-4
TD ......... Type of Drive ............... 5 inch ... 8 inch  0F-0to3
       Single density, single sided            A          B
       Single density, Double sided            C          D
       Double density, Single sided            E          F
       Double density, Double sided            G          H
TC ......... Number of Tracks formatted on the Disk                   03
SPT ........ Number of Sectors per Track                              04
TSR ........ Track Step Rate(0=5,1=10,2=20,3=40ms) Bits0,1,2 of 02&0C
GPL ........ Granules per Lump                                        05
DDSL ....... Disk Directory Starting Lump (used only by Format) 00&08
DDGA ....... Number of Granules allocated to Directory                09
,A ......... If no errors, new PDRIVE active (else not till Reboot)
```

Some examples....

Model I	TI	TD	TC	SPT	TSR	GPL	DDSL	DDGA
NEWDOS/80 Std S/Dens S/Sided	A	A	40	10	1	2	17	2
TRSDOS Std (READ)	A	A	35	10	3	2	17	2
Percom Doubler	CK	E	39	18	1	2	17	2
Double Sided S/Density	A	C	40	20	1	4	17	4
D/Sided Double Dens (Percom)	CK	G	39	36	1	4	17	4
NEWDOS/80 Std 80 Track	A	A	80	10	1	8	35	6
Double Sided S/Density	A	C	80	20	1	8	35	6
D/Sided Double Dens (Percom)	CK	G	79	36	1	8	35	6

Model III	TI	TD	TC	SPT	TSR	GPL	DDSL	DDGA
NEWDOS/80 Std	A	E	40	18	1	2	17	2
TRSDOS Std (READ)	AM	E	40	18	3	6	17	2
Double Sided	A	G	40	36	1	4	17	4
NEWDOS/80 Std 80 Track	A	E	80	18	1	4	35	4
80 Track Double Sided	A	G	80	36	1	8	35	6

DIRECTORY ENTRIES

File Primary Directory Entries (FPDE) store information for the DOS to gain the fast and easy access to files stored on disk which so distinguishes NEWDOS/80 Version 2. A typical entry would be :-

```
10 00 00 B7  00 43 4F 4D  50 54 53 45  52 42 41 52  .....COMPUTERBAS
96 42 96 42  03 00 1D 20  FF FF FF FF  FF FF FF FF  .B.B.............
```

```
Byte 0  =  10H  =  File type & Access level.
Byte 1  =  00H  =  Flags - Bit 7 = ASE :Bit 6 = ASC :Bit 5 = update
Byte 2  =  00H  =  Storage - not used in NEWDOS/80
Byte 3  =  B7H  =  EOF offset in the final sector
                =  Last byte used in final sector
Byte 4  =  00H  =  Length of each record (0-255, 0=256)
Bytes 5  to  12 =  ASCII of File name filled with blanks from right
Bytes 13 to  15 =  ASCII of extension to file name filled as above
Bytes 16 and 17 =  Update password hash code - if unused = 9642H
Bytes 18 and 19 =  Access password hash code - if unused = 9642H
Bytes 20 and 21 =  No. of sectors (from 1) occupied by file (LSB,MSB)
    If byte 3 =0, 3 byte value (bytes 21,20,3)= No bytes used by file
    if byte 3 >0 the value is No bytes used +256, = EOF in RBA format
Byte 22 =  1DH  =  Lump number in which the file starts (from 0)
Byte 23 =  20H  =  Bits 0,1,2,3,4 =  No. of grans used in this extent
   Bits 5,6,7   =  1st granule in the lump the file occupies (0 to 7)
```

The next 3 pairs of bytes have the same format & function as 22 & 23 if the extents are used, otherwise all FF's.

```
Bytes 24 and 25 =  2nd section of contiguous sectors occupied by file
Bytes 26 and 27 =  3rd section of contiguous sectors occupied by file
Bytes 28 and 29 =  4th section of contiguous sectors occupied by file
Byte 30 =  FFH  =  Bit 0 is the flag for the existence of an FXDE
                   Thus FFH if no FXDE, FEH if one exists.
Byte 31 =  FFH  =  DEC. Bits 0-4 = Sector number (0-27) of entry in
   the directory sectors (less 2 used by GAT & HIT) of the next FXDE.
   Bits 5-7 = The FXDE entry number in the sector (# * 32 = byte No.)
```

File Extended Directory Entries (FXDE) are used when more than 4 space areas are assigned to one file. The format is :-

```
Byte 0 = FXDE recognition code. Bits 7 & 4 are both 1 (90H).
Byte 1 = Refers back to the DEC of previous FPDE or FXDE of this file
Bytes 2 to 21 are unused in a FXDE.
Bytes 22 to 31 are the same as an FPDE.
```

GRANULE ALLOCATION TABLE

The Granule Allocation Table (GAT) sector has the information which tells the DOS what granules are free to use and which are used. Also it has bytes describing the mode of formatting, a hash code of the Master Password, the Disk name, and space for an AUTO CHAIN file.

N.B.:- a byte is the space allocated to a lump in the GAT table.

Bytes 0 to (the number of lumps -1) are used to describe the used or unused state of each granule. To gain more space capacity for the Granule table and still keep as far as possible the old 5 sectors per gran, NEWDOS/80 has severed the connection 1 track = 1 byte. Thus 1 lump = 1 byte. The number of bits used to convey this is dependent on the number of granules per lump. Thus the smallest amount of allocatable space (a granule) = 1 bit. In the simplest case, 5 sectors per gran and 2 grans per track (Single density, Single sided, 40 track), the bytes up to 27H (the 40th byte) would be used and only bits 0 and 1 would be used so that the values in bytes 0 to 27H would be FFH (all used) or FCH (all unused) or FEH (the first gran is unused) or FDH (the last gran is unused).

These conditions apply even though the bytes used for the table overrun what Tandy uses as a lockout table (bytes 60H to BFH). This is the reason that only the physical maximum of 202 (CAH) cylinders are allowed on hard drives.

Bytes C0H to CAH (the 193rd to the 203rd bytes) unused except possibly for hard drives.

Byte CBH (the 204th byte) is often used to identify the DOS & format of the disk. TRSDOS = 23H : NEWDOS/80 = 82H : LDOS = 51H : DOSPLUS = 34.

Byte CCH (the 205th byte) is used by some DOS's to store the number of tracks above 35 used on the disk. NEWDOS/80 uses Byte 1FH of HIT Sector.

Byte CDH (the 206th byte) is used in some cases to describe the density, no. of sides, and the number of grans per track. NEWDOS = 0.

Bytes CEH to CFH contain the hash of the master password.
With the password inactive the value is 42E0H.

Bytes D0H to D7H = ASCII string of the disk name (displayed in DIR)

Bytes D8H to DFH = ASCII of date used on creation or changed by PROT

Bytes E0H to FFH = AUTO command buffer. Possible use involves only 31 bytes as the buffer must have an End of Record of 0DH.

HASH INDEX TABLE

The Hash Index table (HIT) sector contains information for the DOS to locate a Directory entry of a file to load it from the disk. When DOS has to load a file, the file name and extension are read and hashed (coded) into a number between 1 & 255. The HIT sector is read to find a match for the hash, then the position of this code in the HIT sector, the DEC, is used to find the entry involved.

Note - DEC = Directory entry code = the position of an entry in the hash table. It is this value that enables fast access to the file.
All files' FPDEs and FXDEs each have an entry in the table.

The HIT sector uses every second line in the sector (0,2,4,etc) to store these hashes. The low 5 bits 0,1,2,3,4 value (0-27) of the byte number are the sector number in the Directory sectors on which the file resides (remembering the first 2 sectors are used by GAT & HIT). The high 3 bits 5,6,7 (0-7) value is the number of the 32 byte entry of that FPDE in the sector. If the Hit sector byte is 00H then bit 4 of the first byte of the entry must be 0 too as the directory entry slot is free. Byte 1FH (the 32nd byte) is allocated to store the number of extra (beyond 10) sectors the directory uses.

Note that the first hash found may be a DEC for an FXDE for the same file or a file with an altogether different name and extension which just happens to hash the same code. The entry ASCII is compared with the required file. If not the same the process starts again.

NEWDOS/80 unlike other DOSes does not exclude the use of the fixed slots for SYS files by other files if the SYS files are not on the disk. The usual track and sector number of the SYS files on a disk which has SPT=10 & GPL=2 (i.e.1 lump = 1 track) is :-

File	Trk No.	Gran No.	Secs used	Len Grans	DEC	File	Trk No.	Gran No.	Secs used	Len Grans	DEC
BOOT /SYS	0	0	0- 1	1	00	SYS20/SYS	15	1	155-159	1	62
SYS0 /SYS	0	1	5- 19	3	20	SYS1 /SYS	16	0	160-164	1	30
SYS17/SYS	10	1	105-109	1	32	SYS2 /SYS	16	1	165-169	1	40
SYS16/SYS	11	0	110-114	1	22	DIR /SYS	17	0	170-179	2	10
SYS11/SYS	11	1	115-119	1	51	SYS3 /SYS	18	0	180-184	1	50
SYS15/SYS	12	0	120-124	1	12	SYS4 /SYS	18	1	185-189	1	60
SYS14/SYS	12	1	125-129	1	02	SYS18/SYS	19	0	190-194	1	21
SYS7 /SYS	13	0	130-134	1	11	SYS9 /SYS	19	1	195-199	1	31
SYS19/SYS	13	1	135-139	1	52	SYS5 /SYS	20	0	200-204	1	70
SYS18/SYS	14	0	140-144	1	42	SYS6 /SYS	20	1	205-239	7	01
SYS13/SYS	14	1	145-149	1	71	SYS12/SYS	24	0	240-245	1	61
SYS10/SYS	15	0	150-154	1	41	SYS21/SYS	24	1	245-249	1	72

HARD DRIVES with NEWDOS/80 2.5

A hard Drive may have
a Data File containing up to 16,000,000 bytes
a Data Volume (the old Drive concept) of up to 65535 Sectors
a Directory of 246 user Files
a PDRIVE allowing 1 to 8 slots (volumes) for a Hard Drive
a Data Volume spanning Tracks and Surfaces

The limitations
65535 Sectors per Volume (PDRIVE slot).
256 Sectors per Track
65536 Tracks (or Recording Surfaces or Heads) per Cylinder
65535 Tracks per Surface - all qualified by
Tracks per Surface times Tracks per Cylinder must be < 65536

The upgrade to 2.5 from 2.0 comes with 4 extra programs :-

1. HDBACKUP/CMD - for backing up or restoring to a Hard Drive
2. HDFMTAPP/CMD - for formatting a Hard Drive
3. EXTPDRIV/BAS - for setting up PDRIVE parameter sets
4. EXTPDAPP/DAT - for holding the PDRIVE parameter sets
   as well as modifications to /SYS programs.

The number of active PDRIVE slots (Drives) is limited to 8 of which up to 4 may be used for Floppy Drives. 1 Floppy Drive is required for backups but 2 is more practical.

The method of storing the backed up Hard Disk files on the floppy is changed so that they cannot be read by standard DOS functions. The sectors containing the programs can be read and changed with Superzap when accessed by Sector not File. Track numbers and File Sector numbers of the Hard Drive are not displayed. Also DIR and FREE will not display Track counts for Hard Drive Volumes.

Bytes 20(14H) and 21 of a Directory entry still contain the Sector count so that it is physically limited to 65535. Again for the same reason the NEWDOS/80 application of Sector I/O limits a Volume (old concept of Drive) to 65535 Sectors. But when you consider that this allows for over 16 million bytes to a Volume or File that restiction fades into impracticality.

With the advent of Hard Drives and their complications a bit of a byte in the GAT Sector of the Directory still refers to a Granule or the minimum amount of space allocatable. Well as you guessed with all that capacity, the Sectors per Gran must increase beyond 5 to be able to cope but the Granules per Lump is best left alone at 8.

HARD DRIVE CONFIGURATION

NEWDOS/80 Version 2.5 is a modified 2.0. Compatibility remains with Tandy's own hardware and for this reason it is confined to the Port usage and Addresses used by the Tandy Hard Drive.

Most of the code driving the Hard Disk is contained in an extra 5 sectors of SYS0/SYS, and is loaded and initialised on Boot-up. After the normal SYS0/SYS program the next Org address on the disk is F900H and this code contains the Hard Drive routines. The extra program protects it's lower boundary (F900H) and up to the end of memory is allocated to the extra routines. Thus 700H (1792) bytes may not be used for anything else. This robs the user of very little when the extra capacity and speed of the Hard Drives are taken into account.

Ports used for Hard Disk I/O by the 2.5 version are C0 to CF. Corvus uses DE and DF and is thus incompatible. The Corvus people have made code (called NEWDOS/80 2.C) to adapt NEWDOS/80 to their Drives and a Host Adaptor to interface to the TRS-80. The hard drives use

Tandy - 256 byte Sectors - 32 Sects per Trk - 153 Trks per Surface
Apparat - 256 byte Sectors - 32 Sects per Trk - 306 Trks per Surface
Corvus - 512 byte Sectors - 20 Sects per Trk - 561 Trks per Surface

The most difficult decision in using Hard Drives involves the PDRIVE HDS parameters. There are 12 of these. The next page has a listing, but first an explanation of some of the terms involved.

DRIVE - A physical entity. = 1 Drive i.e. you may have the maximum of 8 Volumes (PDRIVE slots) on 1 Drive.

VOLUME - 1 PDRIVE slot. A section of a Drive, be it a whole Drive or a part, which has it's own Directory.

SURFACE - Recording surface - 1 per Drive Head -
RSC = Recording Surface count = TPC = Tracks per Cylinder

SECTION - Usually means a recording surface entity.

CYLINDER - Vertically all the same numbered tracks in a Drive i.e. All Cylinders have (RSC) or (TPC) tracks.

GRANULE - Minimum recording space that can be allocated. May contain 5-255 sectors in a Hard Drive. (Std 8)

LUMP - as used in Granules per Lump. An area of the Drive used in the DDSL parameter for the Directory. (See DDSL).

SECTOR INTERLEAVE COUNT - Delay between reading sectors. Optimal, almost mandatory, 21 for NEWDOS/80 2.5 (used in Format)

HARD DRIVE PDRIVE PARAMETERS

1. HDDN - Hard Disk Drive Number
   Just means the Drive No.(0 to 3). (Max. 4 Hard Drives.)
   It specifies the Drive the Volume being PDRIVEd is on.

2. TPS - Tracks per Surface
   The number of Tracks per recording surface.
   Apparat Std = 306 : Tandy Std = 153 : Corvus Std = 561

3. SFS - Section's First Surface
   The recording surface the Volume starts on.
   Can be 0 to Number of Recording Surfaces -1.

4. SSC - Section Surface Count
   The number of surfaces the Volume spans.
   Thus SFS + SSC must be <= the total number of surfaces.

5. SPT - Sectors Per Track
   Number of 256 byte sectors on each track of each surface
   32 in Apparat & Tandy, 20 for Corvus (512 bytes/sector).

6. TSR - Track Stepping Rate
   Apparat = 0 : Tandy = 6 : Corvus = 6

7. VFS - Volume First Sector
   This is the Sector number on which the Volume starts
   Can be 1 to 65535.

8. VSC - Volume Sector Count
   Number of sectors in the volume. If VSC = 0 the PDRIVE
   slot is used but computer reports Device not available.

9. SPG - Sectors per Granule
   Floppy standard & default = 5. If a whole Hard Drive is
   used as 1 Volume, the SPG is increased to keep DDSL<191.

10. GPL - (The old) Granules per Lump
    Between 2 & 8. A Lump in the GAT Sector is 1 Byte.
    A Granule is 1 Bit. The Hard Disk std is the max. 8.

11. DDSL - Default Directory Starting Lump
    The Lump in this volume at whose 1st sector is DIR/SYS.
    Note - Calculated into Sectors by DDSL * GPL * SPG.

12. DDSA - Default Directory Sector Allocation
    Different to DDGA in Floppies because the SPG changes.
    Standard on Hard Drives = 33.

TRS-80 INFORMATION

CHAPTER 6

Z-80 INSTRUCTION SET

THE Z-80 INSTRUCTION SET LISTINGS

There are 4 listings in this Chapter.

Numeric Listing. The purpose of this list is to find what mnemonics the different code values have.

Alphabetic Listing. The purpose of this list is to find the code for a particular Instruction.

Bit Mapped Listing. There is really no practical use to this list. However, in the view of people interested in Machine Code it is a most interesting view of the patterns of bits and an excellent overall picture of the order of things.

Code Action Explained. In an effort to keep this section as concise as possible the format of some bits in the code are only seen from tables accompaning the explanation. The DAA instruction (Decimally Adjust the Accumulator) is given a full Page of explanation as it holds great mystery to the novice. The action of the operations is explained to give as clear as possible a picture of the result as well as any side effects (Flags etc).

The Z80 has 4 sets of Register pairs, one set being currently in use, and the primed set holding their values in an inactive state. These are the AF BC DE and HL, and the AF' BC' DE' and HL' register pairs.

The other register pairs are the IX & IY index registers, the Stack pointer SP, and the Program Counter PC.

There are 2 registers (8 bit) which are very seldom accessed by the user, the I and the R registers. The Interrupt register I is not used in the TRS-80, but the Refresh or Random register is used by the Z80 in the memory refresh cycle and changes every clock cycle.

PLEASE NOTE THAT IF A FLAG IS SET (or reset) UNDER A CERTAIN CONDITION, IF THE CONDITION IS NOT MET THEN THE FLAG IS RESET (or set).

MNEMONICS FOR THE Z-80

The format of an assembled machine language program is :-

Addr.	Code	Line#	Label	Opcode	Operands	ASCII	Comments
7FF0H	2A 20 40	00210	CURSOR	LD	HL,(CURPOS)	! @	; Get Cursor Pos
7FF3H	7D	00220		LD	A,L	.	; LSB of Pos
7FF4H	E6 C0	00230		AND	C0H	..	; Remove bits 0-5
7FF6H	6F	00240		LD	L,A	o	; Start of line.

The first 2 columns (Address & Hex) only appear in an assembly and are not entered by the user. They are formed by the assembler to show the code generated and the address at which it resides.

The 3rd column (Line#) is started and incremented by the computer when the user is entering Mnemonics to the users request or to the default (Start at line 100 and increment by 10)

The 4th column (Label) is an optional address label. The name is used to reference the address involved.

The 5th column (Opcode) contains the operation or action code. This may be a single byte code or up to 4 byte code.

The 6th column (Operand(s)) displays the operands involved, if any. The 5th & 6th columns are the user supplied information that the assembler uses (together with an ORG statement to obtain a start address) to generate the Hex values to place in memory and together they are referred to as the Mnemonics.

The 7th column (ASCII) is an optional output of the disassembler (some don't have it) which shows the ASCII equivalent of the Hex values. When the values are not in the printable range of 21H(33) to 7FH(127) they are usually represented by a full stop.

The 8th column (Comment) is an optional comment on or an explanation of what effect the code has.

Z-80 FLAGS

The flag register is the second part of the AF or the alternate AF' register pair. The F register cannot be accessed in the way other registers can. The manipulation of the A register is all that can be done to this pair.

The chart shows the names of the Flags, Bit numbers, their values and the symbols used to convey conditions.

Bit number Flag name	7 Sign	6 Zero	5 X	4 Half-Carry	3 X	2 Parity/Overflow	1 Add-Subtr.	0 Carry
Symbol if SET Symbol if RESET	M P	Z NZ	X X	- -	X X	PO PE	- -	C NC
Value Decimal Value Hex	128 80H	64 40H	32 20H	16 10H	8 08H	4 04H	2 02H	1 01H

X means this flag slot is not used.
- Means this flag slot is used only by the DAA instruction.
See Page 114 for a full explanation of the DAA instruction.

Generally 8 and 16 Bit Loads do not affect the Flags. The exception is the LD A,R & LD A,I instructions. Incrementing or decrementing 2 byte Registers also does not affect the Flags. However the 8 Bit INC's and DEC's do.

The effect of logical operations on the bits of an operand are

XOR results in a 1 only if the corresponding bits are opposing
XORing with all bits 1 (FFH) inverts all bits (1's complement)
XORing with all bits 0 (00H) leaves the value unchanged
XORing A with itself zeroes A & clears all flags except Zero

AND a bit with 1 does not change the bit,
ANDing with 0 resets it

OR a bit with 0 leaves it unchanged,
ORing with 1 sets it.
'OR A' clears the Carry, Half Carry & Add/Subtract flags
and reflects the current condition in the remaining flags.
An AND is like a Subtract. An OR is like an Add.

All logical instructions except CPL clear the CARRY flag.
SCF & CCF do not affect any flags other than the CARRY.
All PUSHes and POPs do not affect flags (excepting POP AF).

Z-80 FLAGS

BIT 0 = The CARRY flag
- This flag is set (made 1) if there is a carry over to or a borrow from the flag from bit 7 of the accumulator as a Result of an arithmetic or bit operation.

BIT 1 = The ADD/SUBTRACT flag
- This flag is affected only by the Decimally Adjust Accumulator (DAA) operation.

BIT 2 = The PARITY/OVERFLOW flag
- This flag is affected by Arithmetic,Bit,Block Transfer & Search operations and for Interrupt & I/O testing.
Arithmetic - Adding operands with the same sign, if the Result has the opposite sign Bit 2 indicates Overflow.
Bit - Set if addition of values of all 8 bits is EVEN.
Block Transfer & Search - Reset until the counter (BC) is zero.
Interrupts - Reading the Interrupt (I) or refresh (R) Registers the flag receives the Contents of the Interrupt enable (IFF2).
I/O - the flag reflects the parity of the data read.

BIT 3 = not used - generally reflects the value in Bit 3 of A.

BIT 4 = The HALF CARRY flag
- This flag is affected according to the Carry/Borrow status between bits 3 and 4 in the A register (accumulator) after an 8 bit Arithmetic operation. This flag is also used only by the DAA operation.

BIT 5 = not used - generally reflects the value in Bit 5 of A.

BIT 6 = The ZERO flag
- This flag is set by 8 bit arithmetic and logical Operations if A=0, by bit testing if the bit is reset, and by block compare instructions, and in Device I/O (INI,IND,OUTI,OUTD) if the counter (B) is zeroed.

BIT 7 = The SIGN flag
- This flag simply stores the most significant bit (Bit 7) of the A register (accumulator).

POINTS OF INTEREST

DISPLACEMENT VALUES.

The value range for j the jump displacement is given as -126 to 129. This is the Mnemonics offset from the address of the JR instruction itself whereas the Computer uses the value of the Program Counter (the address of the next Instruction).

When making code, the Assembler decrements the Mnemonics value by 2 (the No. Bytes used by the JR instruction) so the actual range has the signed 1 byte possible value of -128 to 127.

THE ACCUMULATOR or A REGISTER.

Though sometimes not specified all 8 Bit Arithmetic and Logical operations except INC & DEC involve the A register.

STACK OPERATIONS.

The address pointed to by the SP (Stack Pointer) holds the 16 bit value that is next to come off the stack. The stack is a last in first off storage which holds a 16 bit value stored for later use or an address (e.g. executed by a RETurn instruction).

The stack pointer can be set to an address by a direct statement or by an indirect register load i.e. LD SP,nn where nn is the address required or with the address held by the HL, IX or IY register pairs. However, there is no instruction to load the SP value into an address or register pair (maybe to find out just where the SP points to).

When a register pair is stored onto the stack (e.g. PUSH BC) the current SP value is first decremented. Then the high byte (B of BC or H of HL) is loaded into this address and the SP is decremented again. Lastly the low byte is loaded to the SP address. Thus the SP points to the low byte of a 16 bit value or address stored in memory in LSB MSB format.

THE PROGRAM COUNTER.

The PC holds the address of the next instruction to be executed. The PC is affected by some instructions which disturb the normal sequential progress of the flow of a program.

A RETurn instruction, for example loads the PC with the current SP value so that execution transfers to a non-sequential address.

There are no listed instructions involving the Program Counter but by clever manipulation or with the help of the ROM the current PC value can be ascertained.

There is at a very low ROM address 2 instructions which may be used for this purpose. At 000BH the byte is E1 (POP HL) and at 000CH is E9 (JP (HL)).

A call to 000BH will cause the PC value to be PUSHed onto the stack. Then a POP HL will transfer this value to HL and a JP (HL) will start the program back at the RETurn address without using the stack.

NUMERIC INSTRUCTION SET : 00H-3FH

Code	Mnemonic		Code	Mnemonic	
00	NOP		20 FE	JR	NZ,j
01 EF CD	LD	BC,nn	21 EF CD	LD	HL,nn
02	LD	(BC),A	22 EF CD	LD	(nn),HL
03	INC	BC	23	INC	HL
04	INC	B	24	INC	H
05	DEC	B	25	DEC	H
06 20	LD	B,v	26 20	LD	H,v
07	RLCA		27	DAA	
08	EX	AF,AF'	28 FE	JR	Z,j
09	ADD	HL,BC	29	ADD	HL,HL
0A	LD	A,(BC)	2A EF CD	LD	HL,(nn)
0B	DEC	BC	2B	DEC	HL
0C	INC	C	2C	INC	L
0D	DEC	C	2D	DEC	L
0E 20	LD	C,v	2E 20	LD	L,v
0F	RRCA		2F	CPL	
10 FE	DJNZ	j	30 FE	JR	NC,j
11 EF CD	LD	DE,nn	31 EF CD	LD	SP,nn
12	LD	(DE),A	32 EF CD	LD	(nn),A
13	INC	DE	33	INC	SP
14	INC	D	34	INC	(HL)
15	DEC	D	35	DEC	(HL)
16 20	LD	D,v	36 20	LD	(HL),v
17	RLA		37	SCF	
18 FE	JR	j	38 FE	JR	C,j
19	ADD	HL,DE	39	ADD	HL,SP
1A	LD	A,(DE)	3A EF CD	LD	A,(nn)
1B	DEC	DE	3B	DEC	SP
1C	INC	E	3C	INC	A
1D	DEC	E	3D	DEC	A
1E 20	LD	E,v	3E 20	LD	A,v
1F	RRA		3F	CCF	

nn = 16 bit Address 0 to 65535 (CDEFH is used in each case)
d = 8 bit Displacement for IX & IY -128 to 127 (05H used)
v = 8 bit Value 0 to 255 (20H used)
j = 8 bit Jump displacement -126 to 129 (FEH used)
p = 8 bit Port address 0 to 255 (FFH used)
(HL) = Contents of the address pointed to by the HL register pair.
The register pairs HL, DE, BC, IX, IY, or SP may be involved.

NUMERIC INSTRUCTION SET : 40H-BFH

Code	Mnemonic		Code	Mnemonic		Code	Mnemonic		Code	Mnemonic	
40	LD	B,B	60	LD	H,B	80	ADD	A,B	A0	AND	B
41	LD	B,C	61	LD	H,C	81	ADD	A,C	A1	AND	C
42	LD	B,D	62	LD	H,D	82	ADD	A,D	A2	AND	D
43	LD	B,E	63	LD	H,E	83	ADD	A,E	A3	AND	E
44	LD	B,H	64	LD	H,H	84	ADD	A,H	A4	AND	H
45	LD	B,L	65	LD	H,L	85	ADD	A,L	A5	AND	L
46	LD	B,(HL)	66	LD	H,(HL)	86	ADD	A,(HL)	A6	AND	(HL)
47	LD	B,A	67	LD	H,A	87	ADD	A,A	A7	AND	A
48	LD	C,B	68	LD	L,B	88	ADC	A,B	A8	XOR	B
49	LD	C,C	69	LD	L,C	89	ADC	A,C	A9	XOR	C
4A	LD	C,D	6A	LD	L,D	8A	ADC	A,D	AA	XOR	D
4B	LD	C,E	6B	LD	L,E	8B	ADC	A,E	AB	XOR	E
4C	LD	C,H	6C	LD	L,H	8C	ADC	A,H	AC	XOR	H
4D	LD	C,L	6D	LD	L,L	8D	ADC	A,L	AD	XOR	L
4E	LD	C,(HL)	6E	LD	L,(HL)	8E	ADC	A,(HL)	AE	XOR	(HL)
4F	LD	C,A	6F	LD	L,A	8F	ADC	A,A	AF	XOR	A
50	LD	D,B	70	LD	(HL),B	90	SUB	B	B0	OR	B
51	LD	D,C	71	LD	(HL),C	91	SUB	C	B1	OR	C
52	LD	D,D	72	LD	(HL),D	92	SUB	D	B2	OR	D
53	LD	D,E	73	LD	(HL),E	93	SUB	E	B3	OR	E
54	LD	D,H	74	LD	(HL),H	94	SUB	H	B4	OR	H
55	LD	D,L	75	LD	(HL),L	95	SUB	L	B5	OR	L
56	LD	D,(HL)	76	HALT		96	SUB	(HL)	B6	OR	(HL)
57	LD	D,A	77	LD	(HL),A	97	SUB	A	B7	OR	A
58	LD	E,B	78	LD	A,B	98	SBC	A,B	B8	CP	B
59	LD	E,C	79	LD	A,C	99	SBC	A,C	B9	CP	C
5A	LD	E,D	7A	LD	A,D	9A	SBC	A,D	BA	CP	D
5B	LD	E,E	7B	LD	A,E	9B	SBC	A,E	BB	CP	E
5C	LD	E,H	7C	LD	A,H	9C	SBC	A,H	BC	CP	H
5D	LD	E,L	7D	LD	A,L	9D	SBC	A,L	BD	CP	L
5E	LD	E,(HL)	7E	LD	A,(HL)	9E	SBC	A,(HL)	BE	CP	(HL)
5F	LD	E,A	7F	LD	A,A	9F	SBC	A,A	BF	CP	A

```
 nn = 16 bit Address          0 to 65535   ( CDEFH is used in each case )
  d =  8 bit Displacement for IX & IY        -128 to 127   ( 05H used )
  v =  8 bit Value                              0 to 255   ( 20H used )
  j =  8 bit Jump displacement               -126 to 129   ( FEH used )
  p =  8 bit Port address                       0 to 255   ( FFH used )
(HL) = Contents of the address pointed to by the HL register pair.
       The register pairs HL, DE, BC, IX, IY, or SP may be involved.
```

NUMERIC INSTRUCTION SET : C0H-FFH

Code	Mnemonic	Code	Mnemonic
C0	RET NZ	E0	RET PO
C1	POP BC	E1	POP HL
C2 EF CD	JP NZ,nn	E2 EF CD	JP PO,nn
C3 EF CD	JP nn	E3	EX (SP),HL
C4 EF CD	CALL NZ,nn	E4 EF CD	CALL PO,nn
C5	PUSH BC	E5	PUSH HL
C6 20	ADD A,v	E6 20	AND v
C7	RST 00H	E7	RST 20H
C8	RET Z	E8	RET PE
C9	RET	E9	JP (HL)
CA EF CD	JP Z,nn	EA EF CD	JP PE,nn
CB Codes	Pages 78,79	EB	EX DE,HL
CC EF CD	CALL Z,nn	EC EF CD	CALL PE,nn
CD EF CD	CALL nn	ED Codes	Page 83
CE 20	ADC A,v	EE 20	XOR v
CF	RST 08H	EF	RST 28H
D0	RET NC	F0	RET P
D1	POP DE	F1	POP AF
D2 EF CD	JP NC,nn	F2 EF CD	JP P,nn
D3 FF	OUT (p),A	F3	DI
D4 EF CD	CALL NC,nn	F4 EF CD	CALL P,nn
D5	PUSH DE	F5	PUSH AF
D6 20	SUB v	F6 20	OR v
D7	RST 10H	F7	RST 30H
D8	RET C	F8	RET M
D9	EXX	F9	LD SP,HL
DA EF CD	JP C,nn	FA EF CD	JP M,nn
DB FF	IN A,(p)	FB	EI
DC EF CD	CALL C,nn	FC EF CD	CALL M,nn
DD Codes	Page 80	FD Codes	Page 82
DE 20	SBC A,v	FE 20	CP v
DF	RST 18H	FF	RST 38H

nn = 16 bit Address 0 to 65535 (CDEFH is used in each case)
d = 8 bit Displacement for IX & IY -128 to 127 (05H used)
v = 8 bit Value 0 to 255 (20H used)
j = 8 bit Jump displacement -126 to 129 (FEH used)
p = 8 bit Port address 0 to 255 (FFH used)
(HL) = Contents of the address pointed to by the HL register pair. The register pairs HL, DE, BC, IX, IY, or SP may be involved.

NUMERIC INSTRUCTION SET : CB 00H-7FH

Code	Mnemonic	Code	Mnemonic	Code	Mnemonic	Code	Mnemonic
CB 00	RLC B	CB 20	SLA B	CB 40	BIT 0,B	CB 60	BIT 4,B
CB 01	RLC C	CB 21	SLA C	CB 41	BIT 0,C	CB 61	BIT 4,C
CB 02	RLC D	CB 22	SLA D	CB 42	BIT 0,D	CB 62	BIT 4,D
CB 03	RLC E	CB 23	SLA E	CB 43	BIT 0,E	CB 63	BIT 4,E
CB 04	RLC H	CB 24	SLA H	CB 44	BIT 0,H	CB 64	BIT 4,H
CB 05	RLC L	CB 25	SLA L	CB 45	BIT 0,L	CB 65	BIT 4,L
CB 06	RLC (HL)	CB 26	SLA (HL)	CB 46	BIT 0,(HL)	CB 66	BIT 4,(HL)
CB 07	RLC A	CB 27	SLA A	CB 47	BIT 0,A	CB 67	BIT 4,A
CB 08	RRC B	CB 28	SRA B	CB 48	BIT 1,B	CB 68	BIT 5,B
CB 09	RRC C	CB 29	SRA C	CB 49	BIT 1,C	CB 69	BIT 5,C
CB 0A	RRC D	CB 2A	SRA D	CB 4A	BIT 1,D	CB 6A	BIT 5,D
CB 0B	RRC E	CB 2B	SRA E	CB 4B	BIT 1,E	CB 6B	BIT 5,E
CB 0C	RRC H	CB 2C	SRA H	CB 4C	BIT 1,H	CB 6C	BIT 5,H
CB 0D	RRC L	CB 2D	SRA L	CB 4D	BIT 1,L	CB 6D	BIT 5,L
CB 0E	RRC (HL)	CB 2E	SRA (HL)	CB 4E	BIT 1,(HL)	CB 6E	BIT 5,(HL)
CB 0F	RRC A	CB 2F	SRA A	CB 4F	BIT 1,A	CB 6F	BIT 5,A
CB 10	RL B	CB 30	SLI B	CB 50	BIT 2,B	CB 70	BIT 6,B
CB 11	RL C	CB 31	SLI C	CB 51	BIT 2,C	CB 71	BIT 6,C
CB 12	RL D	CB 32	SLI D	CB 52	BIT 2,D	CB 72	BIT 6,D
CB 13	RL E	CB 33	SLI E	CB 53	BIT 2,E	CB 73	BIT 6,E
CB 14	RL H	CB 34	SLI H	CB 54	BIT 2,H	CB 74	BIT 6,H
CB 15	RL L	CB 35	SLI L	CB 55	BIT 2,L	CB 75	BIT 6,L
CB 16	RL (HL)	CB 36	SLI (HL)	CB 56	BIT 2,(HL)	CB 76	BIT 6,(HL)
CB 17	RL A	CB 37	SLI A	CB 57	BIT 2,A	CB 77	BIT 6,A
CB 18	RR B	CB 38	SRL B	CB 58	BIT 3,B	CB 78	BIT 7,B
CB 19	RR C	CB 39	SRL C	CB 59	BIT 3,C	CB 79	BIT 7,C
CB 1A	RR D	CB 3A	SRL D	CB 5A	BIT 3,D	CB 7A	BIT 7,D
CB 1B	RR E	CB 3B	SRL E	CB 5B	BIT 3,E	CB 7B	BIT 7,E
CB 1C	RR H	CB 3C	SRL H	CB 5C	BIT 3,H	CB 7C	BIT 7,H
CB 1D	RR L	CB 3D	SRL L	CB 5D	BIT 3,L	CB 7D	BIT 7,L
CB 1E	RR (HL)	CB 3E	SRL (HL)	CB 5E	BIT 3,(HL)	CB 7E	BIT 7,(HL)
CB 1F	RR A	CB 3F	SRL A	CB 5F	BIT 3,A	CB 7F	BIT 7,A

nn = 16 bit Address 0 to 65535 (CDEFH is used in each case)
d = 8 bit Displacement for IX & IY -128 to 127 (05H used)
v = 8 bit Value 0 to 255 (20H used)
j = 8 bit Jump displacement -126 to 129 (FEH used)
p = 8 bit Port address 0 to 255 (FFH used)
(HL) = Contents of the address pointed to by the HL register pair.
The register pairs HL, DE, BC, IX, IY, or SP may be involved.

NUMERIC INSTRUCTION SET : CB - 80H-FFH

Code	Mnemonic	Code	Mnemonic	Code	Mnemonic	Code	Mnemonic
CB 80	RES 0,B	CB A0	RES 4,B	CB C0	SET 0,B	CB E0	SET 4,B
CB 81	RES 0,C	CB A1	RES 4,C	CB C1	SET 0,C	CB E1	SET 4,C
CB 82	RES 0,D	CB A2	RES 4,D	CB C2	SET 0,D	CB E2	SET 4,D
CB 83	RES 0,E	CB A3	RES 4,E	CB C3	SET 0,E	CB E3	SET 4,E
CB 84	RES 0,H	CB A4	RES 4,H	CB C4	SET 0,H	CB E4	SET 4,H
CB 85	RES 0,L	CB A5	RES 4,L	CB C5	SET 0,L	CB E5	SET 4,L
CB 86	RES 0,(HL)	CB A6	RES 4,(HL)	CB C6	SET 0,(HL)	CB E6	SET 4,(HL)
CB 87	RES 0,A	CB A7	RES 4,A	CB C7	SET 0,A	CB E7	SET 4,A
CB 88	RES 1,B	CB A8	RES 5,B	CB C8	SET 1,B	CB E8	SET 5,B
CB 89	RES 1,C	CB A9	RES 5,C	CB C9	SET 1,C	CB E9	SET 5,C
CB 8A	RES 1,D	CB AA	RES 5,D	CB CA	SET 1,D	CB EA	SET 5,D
CB 8B	RES 1,E	CB AB	RES 5,E	CB CB	SET 1,E	CB EB	SET 5,E
CB 8C	RES 1,H	CB AC	RES 5,H	CB CC	SET 1,H	CB EC	SET 5,H
CB 8D	RES 1,L	CB AD	RES 5,L	CB CD	SET 1,L	CB ED	SET 5,L
CB 8E	RES 1,(HL)	CB AE	RES 5,(HL)	CB CE	SET 1,(HL)	CB EE	SET 5,(HL)
CB 8F	RES 1,A	CB AF	RES 5,A	CB CF	SET 1,A	CB EF	SET 5,A
CB 90	RES 2,B	CB B0	RES 6,B	CB D0	SET 2,B	CB F0	SET 6,B
CB 91	RES 2,C	CB B1	RES 6,C	CB D1	SET 2,C	CB F1	SET 6,C
CB 92	RES 2,D	CB B2	RES 6,D	CB D2	SET 2,D	CB F2	SET 6,D
CB 93	RES 2,E	CB B3	RES 6,E	CB D3	SET 2,E	CB F3	SET 6,E
CB 94	RES 2,H	CB B4	RES 6,H	CB D4	SET 2,H	CB F4	SET 6,H
CB 95	RES 2,L	CB B5	RES 6,L	CB D5	SET 2,L	CB F5	SET 6,L
CB 96	RES 2,(HL)	CB B6	RES 6,(HL)	CB D6	SET 2,(HL)	CB F6	SET 6,(HL)
CB 97	RES 2,A	CB B7	RES 6,A	CB D7	SET 2,A	CB F7	SET 6,A
CB 98	RES 3,B	CB B8	RES 7,B	CB D8	SET 3,B	CB F8	SET 7,B
CB 99	RES 3,C	CB B9	RES 7,C	CB D9	SET 3,C	CB F9	SET 7,C
CB 9A	RES 3,D	CB BA	RES 7,D	CB DA	SET 3,D	CB FA	SET 7,D
CB 9B	RES 3,E	CB BB	RES 7,E	CB DB	SET 3,E	CB FB	SET 7,E
CB 9C	RES 3,H	CB BC	RES 7,H	CB DC	SET 3,H	CB FC	SET 7,H
CB 9D	RES 3,L	CB BD	RES 7,L	CB DD	SET 3,L	CB FD	SET 7,L
CB 9E	RES 3,(HL)	CB BE	RES 7,(HL)	CB DE	SET 3,(HL)	CB FE	SET 7,(HL)
CB 9F	RES 3,A	CB BF	RES 7,A	CB DF	SET 3,A	CB FF	SET 7,A

```
  nn = 16 bit Address          0 to 65535  ( CDEFH is used in each case )
   d =  8 bit Displacement for IX & IY       -128 to 127   ( 05H used )
   v =  8 bit Value                             0 to 255   ( 20H used )
   j =  8 bit Jump displacement              -126 to 129   ( FEH used )
   p =  8 bit Port address                      0 to 255   ( FFH used )
(HL) =  Contents of the address pointed to by the HL register pair.
        The register pairs HL, DE, BC, IX, IY, or SP may be involved.
```

NUMERIC INSTRUCTION SET : DD Codes

Code	Mnemonic	Code	Mnemonic
DD 09	ADD IX,BC	DD 6C	LD LX,HX
DD 19	ADD IX,DE	DD 6D	LD LX,LX
		DD 6E 05	LD L,(IX+d)
DD 21 EF CD	LD IX,nn	DD 6F	LD LX,A
DD 22 EF CD	LD (nn),IX	DD 70 05	LD (IX+d),B
DD 23	INC IX	DD 71 05	LD (IX+d),C
DD 24	INC HX	DD 72 05	LD (IX+d),D
DD 25	DEC HX	DD 73 05	LD (IX+d),E
DD 26 20	LD HX,v	DD 74 05	LD (IX+d),H
DD 29	ADD IX,IX	DD 75 05	LD (IX+d),L
DD 2A EF CD	LD IX,(nn)	DD 77 05	LD (IX+d),A
DD 2B	DEC IX	DD 7C	LD A,HX
DD 2C	INC LX	DD 7D	LD A,LX
DD 2D	DEC LX	DD 7E 05	LD A,(IX+d)
DD 2E 20	LD LX,v		
DD 34 05	INC (IX+d)	DD 84	ADD A,HX
DD 35 05	DEC (IX+d)	DD 85	ADD A,LX
DD 36 05 20	LD (IX+d),v	DD 86 05	ADD A,(IX+d)
DD 39	ADD IX,SP	DD 8C	ADC A,HX
		DD 8D	ADC A,LX
DD 44	LD B,HX	DD 8E 05	ADC A,(IX+d)
DD 45	LD B,LX	DD 94	SUB HX
DD 46 05	LD B,(IX+d)	DD 95	SUB LX
DD 4C	LD C,HX	DD 96 05	SUB (IX+d)
DD 4D	LD C,LX	DD 9C	SBC A,HX
DD 4E 05	LD C,(IX+d)	DD 9D	SBC A,LX
DD 54	LD D,HX	DD 9E 05	SBC A,(IX+d)
DD 55	LD D,LX		
DD 56 05	LD D,(IX+d)	DD A4	AND HX
DD 5C	LD E,HX	DD A5	AND LX
DD 5D	LD E,LX	DD A6 05	AND (IX+d)
DD 5E 05	LD E,(IX+d)	DD AC	XOR HX
		DD AD	XOR LX
DD 60	LD HX,B	DD AE 05	XOR (IX+d)
DD 61	LD HX,C	DD B4	OR HX
DD 62	LD HX,D	DD B5	OR LX
DD 63	LD HX,E	DD B6 05	OR (IX+d)
DD 64	LD HX,HX	DD BC	CP HX
DD 65	LD HX,LX	DD BD	CP LX
DD 66 05	LD H,(IX+d)	DD BE 05	CP (IX+d)
DD 67	LD HX,A		
DD 68	LD LX,B	DD E1	POP IX
DD 69	LD LX,C	DD E3	EX (SP),IX
DD 6A	LD LX,D	DD E5	PUSH IX
DD 6B	LD LX,E	DD E9	JP (IX)
		DD F9	LD SP,IX

NUMERIC INSTRUCTION SET : CB — DD & FD

Code	Mnemonic	Code	Mnemonic
DD CB 05 06	RLC (IX+d)	FD CB 05 06	RLC (IY+d)
DD CB 05 0E	RRC (IX+d)	FD CB 05 0E	RRC (IY+d)
DD CB 05 16	RL (IX+d)	FD CB 05 16	RL (IY+d)
DD CB 05 1E	RR (IX+d)	FD CB 05 1E	RR (IY+d)
DD CB 05 26	SLA (IX+d)	FD CB 05 26	SLA (IY+d)
DD CB 05 2E	SRA (IX+d)	FD CB 05 2E	SRA (IY+d)
DD CB 05 36	SLI (IX+d)	FD CB 05 36	SLI (IY+d)
DD CB 05 3E	SRL (IX+d)	FD CB 05 3E	SRL (IY+d)
DD CB 05 46	BIT 0,(IX+d)	FD CB 05 46	BIT 0,(IY+d)
DD CB 05 4E	BIT 1,(IX+d)	FD CB 05 4E	BIT 1,(IY+d)
DD CB 05 56	BIT 2,(IX+d)	FD CB 05 56	BIT 2,(IY+d)
DD CB 05 5E	BIT 3,(IX+d)	FD CB 05 5E	BIT 3,(IY+d)
DD CB 05 66	BIT 4,(IX+d)	FD CB 05 66	BIT 4,(IY+d)
DD CB 05 6E	BIT 5,(IX+d)	FD CB 05 6E	BIT 5,(IY+d)
DD CB 05 76	BIT 6,(IX+d)	FD CB 05 76	BIT 6,(IY+d)
DD CB 05 7E	BIT 7,(IX+d)	FD CB 05 7E	BIT 7,(IY+d)
DD CB 05 86	RES 0,(IX+d)	FD CB 05 86	RES 0,(IY+d)
DD CB 05 8E	RES 1,(IX+d)	FD CB 05 8E	RES 1,(IY+d)
DD CB 05 96	RES 2,(IX+d)	FD CB 05 96	RES 2,(IY+d)
DD CB 05 9E	RES 3,(IX+d)	FD CB 05 9E	RES 3,(IY+d)
DD CB 05 A6	RES 4,(IX+d)	FD CB 05 A6	RES 4,(IY+d)
DD CB 05 AE	RES 5,(IX+d)	FD CB 05 AE	RES 5,(IY+d)
DD CB 05 B6	RES 6,(IX+d)	FD CB 05 B6	RES 6,(IY+d)
DD CB 05 BE	RES 7,(IX+d)	FD CB 05 BE	RES 7,(IY+d)
DD CB 05 C6	SET 0,(IX+d)	FD CB 05 C6	SET 0,(IY+d)
DD CB 05 CE	SET 1,(IX+d)	FD CB 05 CE	SET 1,(IY+d)
DD CB 05 D6	SET 2,(IX+d)	FD CB 05 D6	SET 2,(IY+d)
DD CB 05 DE	SET 3,(IX+d)	FD CB 05 DE	SET 3,(IY+d)
DD CB 05 E6	SET 4,(IX+d)	FD CB 05 E6	SET 4,(IY+d)
DD CB 05 EE	SET 5,(IX+d)	FD CB 05 EE	SET 5,(IY+d)
DD CB 05 F6	SET 6,(IX+d)	FD CB 05 F6	SET 6,(IY+d)
DD CB 05 FE	SET 7,(IX+d)	FD CB 05 FE	SET 7,(IY+d)

nn = 16 bit Address 0 to 65535 (CDEFH is used in each case)
d = 8 bit Displacement for IX & IY -128 to 127 (05H used)
v = 8 bit Value 0 to 255 (20H used)
j = 8 bit Jump displacement -126 to 129 (FEH used)
p = 8 bit Port address 0 to 255 (FFH used)
(HL) = Contents of the address pointed to by the HL register pair.
The register pairs HL, DE, BC, IX, IY, or SP may be involved.

NUMERIC INSTRUCTION SET : FD Codes

Code	Mnemonic		Code	Mnemonic	
FD 09	ADD	IY,BC	FD 6C	LD	LY,HY
FD 19	ADD	IY,DE	FD 6D	LD	LY,LY
			FD 6E 05	LD	L,(IY+d)
FD 21 EF CD	LD	IY,nn	FD 6F	LD	LY,A
FD 22 EF CD	LD	(nn),IY	FD 70 05	LD	(IY+d),B
FD 23	INC	IY	FD 71 05	LD	(IY+d),C
FD 24	INC	HY	FD 72 05	LD	(IY+d),D
FD 25	DEC	HY	FD 73 05	LD	(IY+d),E
FD 26 20	LD	HY,v	FD 74 05	LD	(IY+d),H
FD 29	ADD	IY,IY	FD 75 05	LD	(IY+d),L
FD 2A EF CD	LD	IY,(nn)	FD 77 05	LD	(IY+d),A
FD 2B	DEC	IY	FD 7C	LD	A,HY
FD 2C	INC	LY	FD 7D	LD	A,LY
FD 2D	DEC	LY	FD 7E 05	LD	A,(IY+d)
FD 2E 20	LD	LY,v			
FD 34 05	INC	(IY+d)	FD 84	ADD	A,HY
FD 35 05	DEC	(IY+d)	FD 85	ADD	A,LY
FD 36 05 20	LD	(IY+d),v	FD 86 05	ADD	A,(IY+d)
FD 39	ADD	IY,SP	FD 8C	ADC	A,HY
			FD 8D	ADC	A,LY
FD 44	LD	B,HY	FD 8E 05	ADC	A,(IY+d)
FD 45	LD	B,LY	FD 94	SUB	HY
FD 46 05	LD	B,(IY+d)	FD 95	SUB	LY
FD 4C	LD	C,HY	FD 96 05	SUB	(IY+d)
FD 4D	LD	C,LY	FD 9C	SBC	A,HY
FD 4E 05	LD	C,(IY+d)	FD 9D	SBC	A,LY
FD 54	LD	D,HY	FD 9E 05	SBC	A,(IY+d)
FD 55	LD	D,LY			
FD 56 05	LD	D,(IY+d)	FD A4	AND	HY
FD 5C	LD	E,HY	FD A5	AND	LY
FD 5D	LD	E,LY	FD A6 05	AND	(IY+d)
FD 5E 05	LD	E,(IY+d)	FD AC	XOR	HY
			FD AD	XOR	LY
FD 60	LD	HY,B	FD AE 05	XOR	(IY+d)
FD 61	LD	HY,C	FD B4	OR	HY
FD 62	LD	HY,D	FD B5	OR	LY
FD 63	LD	HY,E	FD B6 05	OR	(IY+d)
FD 64	LD	HY,HY	FD BC	CP	HY
FD 65	LD	HY,LY	FD BD	CP	LY
FD 66 05	LD	H,(IY+d)	FD BE 05	CP	(IY+d)
FD 67	LD	HY,A			
FD 68	LD	LY,B	FD E1	POP	IY
FD 69	LD	LY,C	FD E3	EX	(SP),IY
FD 6A	LD	LY,D	FD E5	PUSH	IY
FD 6B	LD	LY,E	FD E9	JP	(IY)
			FD F9	LD	SP,IY

NUMERIC INSTRUCTION SET : ED Codes

Code	Mnemonic	Code	Mnemonic
ED 40	IN B,(C)	ED 62	SBC HL,HL
ED 41	OUT (C),B	ED 67	RRD
ED 42	SBC HL,BC		
ED 43 EF CD	LD (nn),B	ED 68	IN L,(C)
ED 44	NEG	ED 69	OUT (C),L
ED 45	RETN	ED 6A	ADC HL,HL
ED 46	IM 0	ED 6F	RLD
ED 47	LD I,A		
		ED 72	SBC HL,SP
ED 48	IN C,(C)	ED 73 EF CD	LD (nn),SP
ED 49	OUT (C),C	ED 78	IN A,(C)
ED 4A	ADC HL,BC	ED 79	OUT (C),A
ED 4B EF CD	LD BC,(nn)	ED 7A	ADC HL,SP
ED 4D	RETI	ED 7B EF CD	LD SP,(nn)
ED 4F	LD R,A		
		ED A0	LDI
ED 50	IN D,(C)	ED A1	CPI
ED 51	OUT (C),D	ED A2	INI
ED 52	SBC HL,DE	ED A3	OUTI
ED 53 EF CD	LD (nn),DE	ED A8	LDD
ED 56	IM 1	ED A9	CPD
ED 57	LD A,I	ED AA	IND
		ED AB	OUTD
ED 58	IN E,(C)		
ED 59	OUT (C),E	ED B0	LDIR
ED 5A	ADC HL,DE	ED B1	CPIR
ED 5B EF CD	LD DE,(nn)	ED B2	INIR
ED 5E	IM 2	ED B3	OTIR
ED 5F	LD A,R	ED B8	LDDR
		ED B9	CPDR
ED 60	IN H,(C)	ED BA	INDR
ED 61	OUT (C),H	ED BB	OTDR

nn = 16 bit Address 0 to 65535 (CDEFH is used in each case)
d = 8 bit Displacement for IX & IY -128 to 127 (05H used)
v = 8 bit Value 0 to 255 (20H used)
j = 8 bit Jump displacement -126 to 129 (FEH used)
p = 8 bit Port address 0 to 255 (FFH used)
(HL) = Contents of the address pointed to by the HL register pair.
The register pairs HL, DE, BC, IX, IY, or SP may be involved.

ALPHABETIC INSTRUCTION SET : ADC - BIT

Code	Mnemonic	Code	Mnemonic	Code	Mnemonic
8E	ADC A,(HL)	FD 39	ADD IY,SP	CB 55	BIT 2,L
DD 8E 05	ADC A,(IX+d)	A6	AND (HL)	CB 5E	BIT 3,(HL)
FD 8E 05	ADC A,(IY+d)	DD A6 05	AND (IX+d)	DD CB 05 5E	BIT 3,(IX+d)
CE 20	ADC A,v	FD A6 05	AND (IY+d)	FD CB 05 5E	BIT 3,(IY+d)
8F	ADC A,A	E6 20	AND v	CB 5F	BIT 3,A
88	ADC A,B	A7	AND A	CB 58	BIT 3,B
89	ADC A,C	A0	AND B	CB 59	BIT 3,C
8A	ADC A,D	A1	AND C	CB 5A	BIT 3,D
8B	ADC A,E	A2	AND D	CB 5B	BIT 3,E
8C	ADC A,H	A3	AND E	CB 5C	BIT 3,H
DD 8C	ADC A,HX	A4	AND H	CB 5D	BIT 3,L
FD 8C	ADC A,HY	DD A4	AND HX	CB 66	BIT 4,(HL)
8D	ADC A,L	FD A4	AND HY	DD CB 05 66	BIT 4,(IX+d)
DD 8D	ADC A,LX	A5	AND L	FD CB 05 66	BIT 4,(IY+d)
FD 8D	ADC A,LY	DD A5	AND LX	CB 67	BIT 4,A
ED 4A	ADC HL,BC	FD A5	AND LY	CB 60	BIT 4,B
ED 5A	ADC HL,DE	CB 46	BIT 0,(HL)	CB 61	BIT 4,C
ED 6A	ADC HL,HL	DD CB 05 46	BIT 1,(IX+d)	CB 62	BIT 4,D
ED 7A	ADC HL,SP	FD CB 05 46	BIT 1,(IY+d)	CB 63	BIT 4,E
86	ADD A,(HL)	CB 47	BIT 0,A	CB 64	BIT 4,H
DD 86 05	ADD A,(IX+d)	CB 40	BIT 0,B	CB 65	BIT 4,L
FD 86 05	ADD A,(IY+d)	CB 41	BIT 0,C	CB 6E	BIT 5,(HL)
C6 20	ADD A,v	CB 42	BIT 0,D	DD CB 05 6E	BIT 5,(IX+d)
87	ADD A,A	CB 43	BIT 0,E	FD CB 05 6E	BIT 5,(IY+d)
80	ADD A,B	CB 44	BIT 0,H	CB 6F	BIT 5,A
81	ADD A,C	CB 45	BIT 0,L	CB 68	BIT 5,B
82	ADD A,D	CB 4E	BIT 1,(HL)	CB 69	BIT 5,C
83	ADD A,E	DD CB 05 4E	BIT 1,(IX+d)	CB 6A	BIT 5,D
84	ADD A,H	FD CB 05 4E	BIT 1,(IY+d)	CB 6B	BIT 5,E
DD 84	ADD A,HX	CB 4F	BIT 1,A	CB 6C	BIT 5,H
FD 84	ADD A,HY	CB 48	BIT 1,B	CB 6D	BIT 5,L
85	ADD A,L	CB 49	BIT 1,C	CB 76	BIT 6,(HL)
DD 85	ADD A,LX	CB 4A	BIT 1,D	DD CB 05 76	BIT 6,(IX+d)
FD 85	ADD A,LY	CB 4B	BIT 1,E	DD CB 05 76	BIT 6,(IY+d)
09	ADD HL,BC	CB 4C	BIT 1,H	CB 77	BIT 6,A
19	ADD HL,DE	CB 4D	BIT 1,L	CB 70	BIT 6,B
29	ADD HL,HL	CB 56	BIT 2,(HL)	CB 71	BIT 6,C
39	ADD HL,SP	DD CB 05 56	BIT 2,(IX+d)	CB 72	BIT 6,D
DD 09	ADD IX,BC	FD CB 05 56	BIT 1,(IY+d)	CB 73	BIT 6,E
DD 19	ADD IX,DE	CB 57	BIT 2,A	CB 74	BIT 6,H
DD 29	ADD IX,IX	CB 50	BIT 2,B	CB 75	BIT 6,L
DD 39	ADD IX,SP	CB 51	BIT 2,C	CB 7E	BIT 7,(HL)
FD 09	ADD IY,BC	CB 52	BIT 2,D	DD CB 05 7E	BIT 7,(IX+d)
FD 19	ADD IY,DE	CB 53	BIT 2,E	FD CB 05 7E	BIT 7,(IY+d)
FD 29	ADD IY,IY	CB 54	BIT 2,H	CB 7F	BIT 7,A

ALPHABETIC INSTRUCTION SET : BIT to LD

Code	Mnemonic
CB 78	BIT 7,B
CB 79	BIT 7,C
CB 7A	BIT 7,D
CB 7B	BIT 7,E
CB 7C	BIT 7,H
CB 7D	BIT 7,L
DC EF CD	CALL C,nn
CD EF CD	CALL nn
FC EF CD	CALL M,nn
D4 EF CD	CALL NC,nn
C4 EF CD	CALL NZ,nn
F4 EF CD	CALL P,nn
EC EF CD	CALL PE,nn
E4 EF CD	CALL PO,nn
CC EF CD	CALL Z,nn
3F	CCF
BE	CP (HL)
DD BE 05	CP (IX+d)
FD BE 05	CP (IY+d)
FE 20	CP v
BF	CP A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
DD BC	CP HX
FD BC	CP HY
BD	CP L
DD BD	CP LX
FD BD	CP LY
ED A9	CPD
ED B9	CPDR
ED A1	CPI
ED B1	CPIR
2F	CPL
27	DAA
35	DEC (HL)
DD 35 05	DEC (IX+d)
FD 35 05	DEC (IY+d)
3D	DEC A
05	DEC B
0B	DEC BC
0D	DEC C
15	DEC D
1B	DEC DE
1D	DEC E
25	DEC H
2B	DEC HL
DD 25	DEC HX
FD 25	DEC HY
DD 2B	DEC IX
FD 2B	DEC IY
2D	DEC L
DD 2D	DEC LX
FD 2D	DEC LY
3B	DEC SP
F3	DI
10 80	DJNZ j
FB	EI
E3	EX (SP),HL
DD E3	EX (SP),IX
FD E3	EX (SP),IY
08	EX AF,AF'
EB	EX DE,HL
D9	EXX
76	HALT
ED 46	IM 0
ED 56	IM 1
ED 5E	IM 2
ED 78	IN A,(C)
DB FF	IN A,p
ED 40	IN B,(C)
ED 48	IN C,(C)
ED 50	IN D,(C)
ED 58	IN E,(C)
ED 60	IN H,(C)
ED 68	IN L,(C)
34	INC (HL)
DD 34 05	INC (IX+d)
FD 34 05	INC (IY+d)
3C	INC A
04	INC B
03	INC BC
0C	INC C
14	INC D
13	INC DE
1C	INC E
24	INC H
23	INC HL
DD 24	INC HX
FD 24	INC HY
DD 23	INC IX
FD 23	INC IY
2C	INC L
DD 2C	INC LX
FD 2C	INC LY
33	INC SP
ED AA	IND
ED BA	INDR
ED A2	INI
ED B2	INIR
E9	JP (HL)
DD E9	JP (IX)
FD E9	JP (IY)
DA EF CD	JP C,nn
C3 EF CD	JP nn
FA EF CD	JP M,nn
D2 EF CD	JP NC,nn
C2 EF CD	JP NZ,nn
F2 EF CD	JP P,nn
EA EF CD	JP PE,nn
E2 EF CD	JP PO,nn
CA EF CD	JP Z,nn
18 80	JR j
38 80	JR C,j
30 80	JR NC,j
20 80	JR NZ,j
28 80	JR Z,j
02	LD (BC),A
32 EF CD	LD (nn),A
ED 43 EF CD	LD (nn),BC
ED 53 EF CD	LD (nn),DE
22 EF CD	LD (nn),HL
DD 22 EF CD	LD (nn),IX
FD 22 EF CD	LD (nn),IY
ED 73 EF CD	LD (nn),SP
12	LD (DE),A
36 20	LD (HL),v
77	LD (HL),A
70	LD (HL),B
71	LD (HL),C
72	LD (HL),D
73	LD (HL),E
74	LD (HL),H

Code	Mnemonic	Code	Mnemonic	Code	Mnemonic
75	LD (HL),L	43	LD B,E	5F	LD E,A
DD 36 05 20	LD (IX+d),v	44	LD B,H	58	LD E,B
DD 77 05	LD (IX+d),A	DD 44	LD B,HX	59	LD E,C
DD 70 05	LD (IX+d),B	FD 44	LD B,HY	5A	LD E,D
DD 71 05	LD (IX+d),C	45	LD B,L	5B	LD E,E
DD 72 05	LD (IX+d),D	DD 45	LD B,LX	5C	LD E,H
DD 73 05	LD (IX+d),E	FD 45	LD B,LY	DD 5C	LD E,HX
DD 74 05	LD (IX+d),H	ED 4B EF CD	LD BC,(nn)	FD 5C	LD E,HY
DD 75 05	LD (IX+d),L	01 EF CD	LD BC,nn	5D	LD E,L
FD 36 05 20	LD (IY+d),v	4E	LD C,(HL)	DD 5D	LD E,LX
FD 77 05	LD (IY+d),A	DD 4E 05	LD C,(IX+d)	FD 5D	LD E,LY
FD 70 05	LD (IY+d),B	FD 4E 05	LD C,(IY+d)	66	LD H,(HL)
FD 71 05	LD (IY+d),C	0E 20	LD C,v	DD 66 05	LD H,(IX+d)
FD 72 05	LD (IY+d),D	4F	LD C,A	FD 66 05	LD H,(IY+d)
FD 73 05	LD (IY+d),E	48	LD C,B	26 20	LD H,v
FD 74 05	LD (IY+d),H	49	LD C,C	67	LD H,A
FD 75 05	LD (IY+d),L	4A	LD C,D	60	LD H,B
0A	LD A,(BC)	4B	LD C,E	61	LD H,C
3A EF CD	LD A,(nn)	4C	LD C,H	62	LD H,D
1A	LD A,(DE)	DD 4C	LD C,HX	63	LD H,E
7E	LD A,(HL)	FD 4C	LD C,HY	64	LD H,H
DD 7E 05	LD A,(IX+d)	4D	LD C,L	65	LD H,L
FD 7E 05	LD A,(IY+d)	DD 4D	LD C,LX	2A EF CD	LD HL,(nn)
3E 20	LD A,v	FD 4D	LD C,LY	21 EF CD	LD HL,nn
7F	LD A,A	56	LD D,(HL)	DD 26 20	LD HX,v
78	LD A,B	DD 56 05	LD D,(IX+d)	DD 67	LD HX,A
79	LD A,C	FD 56 05	LD D,(IY+d)	DD 60	LD HX,B
7A	LD A,D	16 20	LD D,v	DD 61	LD HX,C
7B	LD A,E	57	LD D,A	DD 62	LD HX,D
7C	LD A,H	50	LD D,B	DD 63	LD HX,E
DD 7C	LD A,HX	51	LD D,C	DD 64	LD HX,HX
FD 7C	LD A,HY	52	LD D,D	DD 65	LD HX,LX
ED 57	LD A,I	53	LD D,E	FD 26 20	LD HY,v
7D	LD A,L	54	LD D,H	FD 67	LD HY,A
DD 7D	LD A,LX	DD 54	LD D,HX	FD 60	LD HY,B
FD 7D	LD A,LY	FD 54	LD D,HY	FD 61	LD HY,C
ED 5F	LD A,R	55	LD D,L	FD 62	LD HY,D
46	LD B,(HL)	DD 55	LD D,LX	FD 63	LD HY,E
DD 46 05	LD B,(IX+d)	FD 55	LD D,LY	FD 64	LD HY,HY
FD 46 05	LD B,(IY+d)	ED 5B EF CD	LD DE,(nn)	FD 65	LD HY,LY
06 20	LD B,v	11 EF CD	LD DE,nn	ED 47	LD I,A
47	LD B,A	5E	LD E,(HL)	DD 2A EF CD	LD IX,(nn)
40	LD B,B	DD 5E 05	LD E,(IX+d)	DD 21 EF CD	LD IX,nn
41	LD B,C	FD 5E 05	LD E,(IY+d)	FD 2A EF CD	LD IY,(nn)
42	LD B,D	1E 20	LD E,v	FD 21 EF CD	LD IY,nn

ALPHABETIC INSTRUCTION SET : LD to RES

Code	Mnemonic	Code	Mnemonic	Code	Mnemonic
6E	LD L,(HL)	B1	OR C	FD CB 05 8E	RES 1,(IY+d)
DD 6E 05	LD L,(IX+d)	B2	OR D	CB 8F	RES 1,A
FD 6E 05	LD L,(IY+d)	B3	OR E	CB 88	RES 1,B
2E 20	LD L,v	B4	OR H	CB 89	RES 1,C
6F	LD L,A	DD B4	OR HX	CB 8A	RES 1,D
68	LD L,B	FD B4	OR HY	CB 8B	RES 1,E
69	LD L,C	B5	OR L	CB 8C	RES 1,H
6A	LD L,D	DD B5	OR LX	CB 8D	RES 1,L
6B	LD L,E	FD B5	OR LY	CB 96	RES 2,(HL)
6C	LD L,H	ED BB	OTDR	DD CB 05 96	RES 2,(IX+d)
6D	LD L,L	ED B3	OTIR	FD CB 05 96	RES 2,(IY+d)
DD 2E 20	LD LX,v	ED 79	OUT (C),A	CB 97	RES 2,A
DD 6F	LD LX,A	ED 41	OUT (C),B	CB 90	RES 2,B
DD 68	LD LX,B	ED 49	OUT (C),C	CB 91	RES 2,C
DD 69	LD LX,C	ED 51	OUT (C),D	CB 92	RES 2,D
DD 6A	LD LX,D	ED 59	OUT (C),E	CB 93	RES 2,E
DD 6B	LD LX,E	ED 61	OUT (C),H	CB 94	RES 2,H
DD 6C	LD LX,HX	ED 69	OUT (C),L	CB 95	RES 2,L
DD 6D	LD LX,LX	D3 FF	OUT p,A	CB 9E	RES 3,(HL)
FD 2E 20	LD LY,v	ED AB	OUTD	DD CB 05 9E	RES 3,(IX+d)
FD 6F	LD LY,A	ED A3	OUTI	FD CB 05 9E	RES 3,(IY+d)
FD 68	LD LY,B	F1	POP AF	CB 9F	RES 3,A
FD 69	LD LY,C	C1	POP BC	CB 98	RES 3,B
FD 6A	LD LY,D	D1	POP DE	CB 99	RES 3,C
FD 6B	LD LY,E	E1	POP HL	CB 9A	RES 3,D
FD 6C	LD LY,HY	DD E1	POP IX	CB 9B	RES 3,E
FD 6D	LD LY,LY	FD E1	POP IY	CB 9C	RES 3,H
ED 4F	LD R,A	F5	PUSH AF	CB 9D	RES 3,L
ED 7B EF CD	LD SP,(nn)	C5	PUSH BC	CB A6	RES 4,(HL)
31 EF CD	LD SP,nn	D5	PUSH DE	DD CB 05 A6	RES 4,(IX+d)
DD F9	LD SP,IX	E5	PUSH HL	FD CB 05 A6	RES 4,(IY+d)
FD F9	LD SP,IY	DD E5	PUSH IX	CB A7	RES 4,A
F9	LD SP,HL	FD E5	PUSH IY	CB A0	RES 4,B
ED A8	LDD	CB 86	RES 0,(HL)	CB A1	RES 4,C
ED B8	LDD	DD CB 05 86	RES 0,(IX+d)	CB A2	RES 4,D
ED A0	LDI	FD CB 05 86	RES 0,(IY+d)	CB A3	RES 4,E
ED B0	LDIR	CB 87	RES 0,A	CB A4	RES 4,H
ED 44	NEG	CB 80	RES 0,B	CB A5	RES 4,L
00	NOP	CB 81	RES 0,C	CB AE	RES 5,(HL)
B6	OR (HL)	CB 82	RES 0,D	DD CB 05 AE	RES 5,(IX+d)
DD B6 05	OR (IX+d)	CB 83	RES 0,E	FD CB 05 AE	RES 5,(IY+d)
FD B6 05	OR (IY+d)	CB 84	RES 0,H	CB AF	RES 5,A
F6 20	OR v	CB 85	RES 0,L	CB A8	RES 5,B
B7	OR A	CB 8E	RES 1,(HL)	CB A9	RES 5,C
B0	OR B	DD CB 05 8E	RES 1,(IX+d)	CB AA	RES 5,D

ALPHABETIC INSTRUCTION SET : RES - SET

Code	Mnemonic	Code	Mnemonic	Code	Mnemonic
CB AB	RES 5,E	CB 06	RLC (HL)	FD 9E 05	SBC A,(IY+d)
CB AC	RES 5,H	DD CB 05 06	RLC (IX+d)	DE 20	SBC A,v
CB AD	RES 5,L	FD CB 05 06	RLC (IY+d)	9F	SBC A,A
CB B6	RES 6,(HL)	CB 07	RLC A	98	SBC A,B
DD CB 05 B6	RES 6,(IX+d)	CB 00	RLC B	99	SBC A,C
FD CB 05 B6	RES 6,(IY+d)	CB 01	RLC C	9A	SBC A,D
CB B7	RES 6,A	CB 02	RLC D	9B	SBC A,E
CB B0	RES 6,B	CB 03	RLC E	9C	SBC A,H
CB B1	RES 6,C	CB 04	RLC H	DD 9C	SBC A,HX
CB B2	RES 6,D	CB 05	RLC L	FD 9C	SBC A,HY
CB B3	RES 6,E	07	RLCA	9D	SBC A,L
CB B4	RES 6,H	ED 6F	RLD	DD 9D	SBC A,LX
CB B5	RES 6,L	CB 1E	RR (HL)	FD 9D	SBC A,LY
CB BE	RES 7,(HL)	DD CB 05 1E	RR (IX+d)	ED 42	SBC HL,BC
DD CB 05 BE	RES 7,(IX+d)	FD CB 05 1E	RR (IY+d)	ED 52	SBC HL,DE
FD CB 05 BE	RES 7,(IY+d)	CB 1F	RR A	ED 62	SBC HL,HL
CB BF	RES 7,A	CB 18	RR B	ED 72	SBC HL,SP
CB B8	RES 7,B	CB 19	RR C	37	SCF
CB B9	RES 7,C	CB 1A	RR D	CB C6	SET 0,(HL)
CB BA	RES 7,D	CB 1B	RR E	DD CB 05 C6	SET 0,(IX+d)
CB BB	RES 7,E	CB 1C	RR H	FD CB 05 C6	SET 0,(IY+d)
CB BC	RES 7,H	CB 1D	RR L	CB C7	SET 0,A
CB BD	RES 7,L	1F	RRA	CB C0	SET 0,B
C9	RET	CB 0E	RRC (HL)	CB C1	SET 0,C
D8	RET C	DD CB 05 0E	RRC (IX+d)	CB C2	SET 0,D
F8	RET M	FD CB 05 0E	RRC (IY+d)	CB C3	SET 0,E
D0	RET NC	CB 0F	RRC A	CB C4	SET 0,H
C0	RET NZ	CB 08	RRC B	CB C5	SET 0,L
F0	RET P	CB 09	RRC C	CB CE	SET 1,(HL)
E8	RET PE	CB 0A	RRC D	DD CB 05 CE	SET 1,(IX+d)
E0	RET PO	CB 0B	RRC E	FD CB 05 CE	SET 1,(IY+d)
C8	RET Z	CB 0C	RRC H	CB CF	SET 1,A
ED 4D	RETI	CB 0D	RRC L	CB C8	SET 1,B
ED 45	RETN	0F	RRCA	CB C9	SET 1,C
CB 16	RL (HL)	ED 67	RRD	CB CA	SET 1,D
DD CB 05 16	RL (IX+d)	C7	RST 00H	CB CB	SET 1,E
FD CB 05 16	RL (IY+d)	CF	RST 08H	CB CC	SET 1,H
CB 17	RL A	D7	RST 10H	CB CD	SET 1,L
CB 10	RL B	DF	RST 18H	CB D6	SET 2,(HL)
CB 11	RL C	E7	RST 20H	DD CB 05 D6	SET 2,(IX+d)
CB 12	RL D	EF	RST 28H	FD CB 05 D6	SET 2,(IY+d)
CB 13	RL E	F7	RST 30H	CB D7	SET 2,A
CB 14	RL H	FF	RST 38H	CB D0	SET 2,B
CB 15	RL L	9E	SBC A,(HL)	CB D1	SET 2,C
17	RLA	DD 9E 05	SBC A,(IX+d)	CB D2	SET 2,D

ALPHABETIC INSTRUCTION SET : SET to XOR

Code	Mnemonic
CB D3	SET 2,E
CB D4	SET 2,H
CB D5	SET 2,L
CB DE	SET 3,(HL)
DD CB 05 DE	SET 3,(IX+d)
FD CB 05 DE	SET 3,(IY+d)
CB DF	SET 3,A
CB D8	SET 3,B
CB D9	SET 3,C
CB DA	SET 3,D
CB DB	SET 3,E
CB DC	SET 3,H
CB DD	SET 3,L
CB E6	SET 4,(HL)
DD CB 05 E6	SET 4,(IX+d)
FD CB 05 E6	SET 4,(IY+d)
CB E7	SET 4,A
CB E0	SET 4,B
CB E1	SET 4,C
CB E2	SET 4,D
CB E3	SET 4,E
CB E4	SET 4,H
CB E5	SET 4,L
CB EE	SET 5,(HL)
DD CB 05 EE	SET 5,(IX+d)
FD CB 05 EE	SET 5,(IY+d)
CB EF	SET 5,A
CB E8	SET 5,B
CB E9	SET 5,C
CB EA	SET 5,D
CB EB	SET 5,E
CB EC	SET 5,H
CB ED	SET 5,L
CB F6	SET 6,(HL)
DD CB 05 F6	SET 6,(IX+d)
FD CB 05 F6	SET 6,(IY+d)
CB F7	SET 6,A
CB F0	SET 6,B
CB F1	SET 6,C
CB F2	SET 6,D
CB F3	SET 6,E
CB F4	SET 6,H
CB F5	SET 6,L
CB FE	SET 7,(HL)
DD CB 05 FE	SET 7,(IX+d)
FD CB 05 FE	SET 7,(IY+d)
CB FF	SET 7,A
CB F8	SET 7,B
CB F9	SET 7,C
CB FA	SET 7,D
CB FB	SET 7,E
CB FC	SET 7,H
CB FD	SET 7,L
CB 26	SLA (HL)
DD CB 05 26	SLA (IX+d)
FD CB 05 26	SLA (IY+d)
CB 27	SLA A
CB 20	SLA B
CB 21	SLA C
CB 22	SLA D
CB 23	SLA E
CB 24	SLA H
CB 25	SLA L
CB 36	SLI (HL)
DD CB 05 36	SLI (IX+d)
FD CB 05 36	SLI (IY+d)
CB 37	SLI A
CB 30	SLI B
CB 31	SLI C
CB 32	SLI D
CB 33	SLI E
CB 34	SLI H
CB 35	SLI L
CB 2E	SRA (HL)
DD CB 05 2E	SRA (IX+d)
FD CB 05 2E	SRA (IY+d)
CB 2F	SRA A
CB 28	SRA B
CB 29	SRA C
CB 2A	SRA D
CB 2B	SRA E
CB 2C	SRA H
CB 2D	SRA L
CB 3E	SRL (HL)
DD CB 05 3E	SRL (IX+d)
FD CB 05 3E	SRL (IY+d)
CB 3F	SRL A
CB 38	SRL B
CB 39	SRL C
CB 3A	SRL D
CB 3B	SRL E
CB 3C	SRL H
CB 3D	SRL L
CB 96	SUB (HL)
DD 96 05	SUB (IX+d)
FD 96 05	SUB (IY+d)
D6 20	SUB v
97	SUB A
90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
DD 94	SUB HX
FD 94	SUB HY
95	SUB L
DD 95	SUB LX
FD 95	SUB LY
AE	XOR (HL)
DD AE 05	XOR (IX+d)
FD AE 05	XOR (IY+d)
EE 20	XOR v
AF	XOR A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
DD AC	XOR HX
FD AC	XOR HY
AD	XOR L
DD AD	XOR LX
DD AE	XOR LY

Key to Symbols

nn = 16 bit Address
d = 8 bit Displ'ment
v = 8 bit Value
j = 8 bit Jump displ.
p = Port Address
( ) = The contents of

INSTRUCTION SET — BIT MAPPED

1 Byte Codes

Bits	.... .000	.... .001	.... .010	.... .011
0000 0...	00 NOP	01 LD BC,nn	02 LD (BC),A	03 INC BC
0000 1...	08 EX AF,AF'	09 ADD HL,BC	0A LD A,(BC)	0B DEC BC
0001 0...	10 DJNZ j	11 LD DE,nn	12 LD (DE),A	13 INC DE
0001 1...	18 JR j	19 ADD HL,DE	1A LD A,(DE)	1B DEC DE
0010 0...	20 JR NZ,j	21 LD HL,nn	22 LD (nn),HL	23 INC HL
0010 1...	28 JR Z,j	29 ADD HL,HL	2A LD HL,(nn)	2B DEC HL
0011 0...	30 JR NC,j	31 LD SP,nn	32 LD (nn),A	33 INC SP
0011 1...	38 JR C,j	39 ADD HL,SP	3A LD A,(nn)	3B DEC SP
0100 0...	40 LD B,B	41 LD B,C	42 LD B,D	43 LD B,E
0100 1...	48 LD C,B	49 LD C,C	4A LD C,D	4B LD C,E
0101 0...	50 LD D,B	51 LD D,C	52 LD D,D	53 LD D,E
0101 1...	58 LD E,B	59 LD E,C	5A LD E,D	5B LD E,E
0110 0...	60 LD H,B	61 LD H,C	62 LD H,D	63 LD H,E
0110 1...	68 LD L,B	69 LD L,C	6A LD L,D	6B LD L,E
0111 0...	70 LD (HL),B	71 LD (HL),C	72 LD (HL),D	73 LD (HL),E
0111 1...	78 LD A,B	79 LD A,C	7A LD A,D	7B LD A,E
1000 0...	80 ADD A,B	81 ADD A,C	82 ADD A,D	83 ADD A,E
1000 1...	88 ADC A,B	89 ADC A,C	8A ADC A,D	8B ADC A,E
1001 0...	90 SUB B	91 SUB C	92 SUB D	93 SUB E
1001 1...	98 SBC A,B	99 SBC A,C	9A SBC A,D	9B SBC A,E
1010 0...	A0 AND B	A1 AND C	A2 AND D	A3 AND E
1010 1...	A8 XOR B	A9 XOR C	AA XOR D	AB XOR E
1011 0...	B0 OR B	B1 OR C	B2 OR D	B3 OR E
1011 1...	B8 CP B	B9 CP C	BA CP D	BB CP E
1100 0...	C0 RET NZ	C1 POP BC	C2 JP NZ,nn	C3 JP nn
1100 1...	C8 RET Z	C9 RET	CA JP Z,nn	CB Codes
1101 0...	D0 RET NC	D1 POP DE	D2 JP NC,nn	D3 OUT p,A
1101 1...	D8 RET C	D9 EXX	DA JP C,nn	DB IN A,p
1110 0...	E0 RET PO	E1 POP HL	E2 JP PO,nn	E3 EX (SP),HL
1110 1...	E8 RET PE	E9 JP (HL)	EA JP PE,nn	EB EX DE,HL
1111 0...	F0 RET P	F1 POP AF	F2 JP P,nn	F3 DI
1111 1...	F8 RET M	F9 LD SP,HL	FA JP M,nn	FB EI
Bits	.... .000	.... .001	.... .010	.... .011

INSTRUCTION SET - BIT MAPPED

1 Byte Codes

.... .100	.... .101	.... .110	.... .111	Bits
04 INC B	05 DEC B	06 LD B,v	07 RLCA	0000 0...
0C INC C	0D DEC C	0E LD C,v	0F RRCA	0000 1...
14 INC D	15 DEC D	16 LD D,v	17 RLA	0001 0...
1C INC E	1D DEC E	1E LD E,v	1F RRA	0001 1...
24 INC H	25 DEC H	26 LD H,v	27 DAA	0010 0...
2C INC L	2D DEC L	2E LD L,v	2F CPL	0010 1...
34 INC (HL)	35 DEC (HL)	36 LD (HL),v	37 SCF	0011 0...
3C INC A	3D DEC A	3E LD A,v	3F CCF	0011 1...
44 LD B,H	45 LD B,L	46 LD B,(HL)	47 LD B,A	0100 0...
4C LD C,H	4D LD C,L	4E LD C,(HL)	4F LD C,A	0100 1...
54 LD D,H	55 LD D,L	56 LD D,(HL)	57 LD D,A	0101 0...
5C LD E,H	5D LD E,L	5E LD E,(HL)	5F LD E,A	0101 1...
64 LD H,H	65 LD H,L	66 LD H,(HL)	67 LD H,A	0110 0...
6C LD L,H	6D LD L,L	6E LD L,(HL)	6F LD L,A	0110 1...
74 LD (HL),H	75 LD (HL),L	76 HALT	77 LD (HL), A	0111 0...
7C LD A,H	7D LD A,L	7E LD A,(HL)	7F LD A,A	0111 1...
84 ADD A,H	85 ADD A,L	86 ADD A,(HL)	87 ADD A,A	1000 0...
8C ADC A,H	8D ADC A,L	8E ADC A,(HL)	8F ADC A,A	1000 1...
94 SUB H	95 SUB L	96 SUB (HL)	97 SUB A	1001 0...
9C SBC A,H	9D SBC A,L	9E SBC A,(HL)	9F SBC A,A	1001 1...
A4 AND H	A5 AND L	A6 AND (HL)	A7 AND A	1010 0...
AC XOR H	AD XOR L	AE XOR (HL)	AF XOR A	1010 1...
B4 OR H	B5 OR L	B6 OR (HL)	B7 OR A	1011 0...
BC CP H	BD CP L	BE CP (HL)	BF CP A	1011 1...
C4 CALL NZ,nn	C5 PUSH BC	C6 ADD A,v	C7 RST 00H	1100 0...
CC CALL Z,nn	CD CALL nn	CE ADC A,v	CF RST 08H	1100 1...
D4 CALL NC,nn	D5 PUSH DE	D6 SUB A,v	D7 RST 10H	1101 0...
DC CALL C,nn	DD Codes	DE SBC A,v	DF RST 18H	1101 1...
E4 CALL PO,nn	E5 PUSH HL	E6 AND v	E7 RST 20H	1110 0...
EC CALL PE,nn	ED Codes	EE XOR v	EF RST 28H	1110 1...
F4 CALL P,nn	F5 PUSH AF	F6 OR v	F7 RST 30H	1111 0...
FC CALL M,nn	FD Codes	FE CP v	FF RST 38H	1111 1...
.... .100	.... .101	.... .110	.... .111	Bits

INSTRUCTION SET - BIT MAPPED

2nd Byte of 2 Byte CB .. Codes

Bits	.... .000	.... .001	.... .010	.... .011
0000 0...	00 RLC B	01 RLC C	02 RLC D	03 RLC E
0000 1...	08 RRC B	09 RRC C	0A RRC D	0B RRC E
0001 0...	10 RL B	11 RL C	12 RL D	13 RL E
0001 1...	18 RR B	19 RR C	1A RR D	1B RR E
0010 0...	20 SLA B	21 SLA C	22 SLA D	23 SLA E
0010 1...	28 SRA B	29 SRA C	2A SRA D	2B SRA E
0011 0...	30 SLI B	31 SLI C	32 SLI D	33 SLI E
0011 1...	38 SRL B	39 SRL C	3A SRL D	3B SRL E
0100 0...	40 BIT 0,B	41 BIT 0,C	42 BIT 0,D	43 BIT 0,E
0100 1...	48 BIT 1,B	49 BIT 1,C	4A BIT 1,D	4B BIT 1,E
0101 0...	50 BIT 2,B	51 BIT 2,C	52 BIT 2,D	53 BIT 2,E
0101 1...	58 BIT 3,B	59 BIT 3,C	5A BIT 3,D	5B BIT 3,E
0110 0...	60 BIT 4,B	61 BIT 4,C	62 BIT 4,D	63 BIT 4,E
0110 1...	68 BIT 5,B	69 BIT 5,C	6A BIT 5,D	6B BIT 5,E
0111 0...	70 BIT 6,B	71 BIT 6,C	72 BIT 6,D	73 BIT 6,E
0111 1...	78 BIT 7,B	79 BIT 7,C	7A BIT 7,D	7B BIT 7,E
1000 0...	80 RES 0,B	81 RES 0,C	82 RES 0,D	83 RES 0,E
1000 1...	88 RES 1,B	89 RES 1,C	8A RES 1,D	8B RES 1,E
1001 0...	90 RES 2,B	91 RES 2,C	92 RES 2,D	93 RES 2,E
1001 1...	98 RES 3,B	99 RES 3,C	9A RES 3,D	9B RES 3,E
1010 0...	A0 RES 4,B	A1 RES 4,C	A2 RES 4,D	A3 RES 4,E
1010 1...	A8 RES 5,B	A9 RES 5,C	AA RES 5,D	AB RES 5,E
1011 0...	B0 RES 6,B	B1 RES 6,C	B2 RES 6,D	B3 RES 6,E
1011 1...	B8 RES 7,B	B9 RES 7,C	BA RES 7,D	BB RES 7,E
1100 0...	C0 SET 0,B	C1 SET 0,C	C2 SET 0,D	C3 SET 0,E
1100 1...	C8 SET 1,B	C9 SET 1,C	CA SET 1,D	CB SET 1,E
1101 0...	D0 SET 2,B	D1 SET 2,C	D2 SET 2,D	D3 SET 2,E
1101 1...	D8 SET 3,B	D9 SET 3,C	DA SET 3,D	DB SET 3,E
1110 0...	E0 SET 4,B	E1 SET 4,C	E2 SET 4,D	E3 SET 4,E
1110 1...	E8 SET 5,B	E9 SET 5,C	EA SET 5,D	EB SET 5,E
1111 0...	F0 SET 6,B	F1 SET 6,C	F2 SET 6,D	F3 SET 6,E
1111 1...	F8 SET 7,B	F9 SET 7,C	FA SET 7,D	FB SET 7,E
Bits	.... .000	.... .001	.... .010	.... .011

INSTRUCTION SET — BIT MAPPED

2nd Byte of 2 Byte CB .. Codes

.... .100	.... .101	.... .110	.... .111	Bits
04 RLC H	05 RLC L	06 RLC (HL)	07 RLC A	0000 0...
0C RRC H	0D RRC L	0E RRC (HL)	0F RRC A	0000 1...
14 RL H	15 RL L	16 RL (HL)	17 RL A	0001 0...
1C RR H	1D RR L	1E RR (HL)	1F RR A	0001 1...
24 SLA H	25 SLA L	26 SLA (HL)	27 SLA A	0010 0...
2C SRA H	2D SRA L	2E SRA (HL)	2F SRA A	0010 1...
34 SLI H	35 SLI L	36 SLI (HL)	37 SLI A	0011 0...
3C SRL H	3D SRL L	3E SRL (HL)	3F SRL A	0011 1...
44 BIT 0,H	45 BIT 0,L	46 BIT 0,(HL)	47 BIT 0,A	0100 0...
4C BIT 1,H	4D BIT 1,L	4E BIT 1,(HL)	4F BIT 1,A	0100 1...
54 BIT 2,H	55 BIT 2,L	56 BIT 2,(HL)	57 BIT 2,A	0101 0...
5C BIT 3,H	5D BIT 3,L	5E BIT 3,(HL)	5F BIT 3,A	0101 1...
64 BIT 4,H	65 BIT 4,L	66 BIT 4,(HL)	67 BIT 4,A	0110 0...
6C BIT 5,H	6D BIT 5,L	6E BIT 5,(HL)	6F BIT 5,A	0110 1...
74 BIT 6,H	75 BIT 6,L	76 BIT 6,(HL)	77 BIT 6,A	0111 0...
7C BIT 7,H	7D BIT 7,L	7E BIT 7,(HL)	7F BIT 7,A	0111 1...
84 RES 0,H	85 RES 0,L	86 RES 0,(HL)	87 RES 0,A	1000 0...
8C RES 1,H	8D RES 1,L	8E RES 1,(HL)	8F RES 1,A	1000 1...
94 RES 2,H	95 RES 2,L	96 RES 2,(HL)	97 RES 2,A	1001 0...
9C RES 3,H	9D RES 3,L	9E RES 3,(HL)	9F RES 3,A	1001 1...
A4 RES 4,H	A5 RES 4,L	A6 RES 4,(HL)	A7 RES 4,A	1010 0...
AC RES 5,H	AD RES 5,L	AE RES 5,(HL)	AF RES 5,A	1010 1...
B4 RES 6,H	B5 RES 6,L	B6 RES 6,(HL)	B7 RES 6,A	1011 0...
BC RES 7,H	BD RES 7,L	BE RES 7,(HL)	BF RES 7,A	1011 1...
C4 SET 0,H	C5 SET 0,L	C6 SET 0,(HL)	C7 SET 0,A	1100 0...
CC SET 1,H	CD SET 1,L	CE SET 1,(HL)	CF SET 1,A	1100 1...
D4 SET 2,H	D5 SET 2,L	D6 SET 2,(HL)	D7 SET 2,A	1101 0...
DC SET 3,H	DD SET 3,L	DE SET 3,(HL)	DF SET 3,A	1101 1...
E4 SET 4,H	E5 SET 4,L	E6 SET 4,(HL)	E7 SET 4,A	1110 0...
EC SET 5,H	ED SET 5,L	EE SET 5,(HL)	EF SET 5,A	1110 1...
F4 SET 6,H	F5 SET 6,L	F6 SET 6,(HL)	F7 SET 6,A	1111 0...
FC SET 7,H	FD SET 7,L	FE SET 7,(HL)	FF SET 7,A	1111 1...
.... .100	.... .101	.... .110	.... .111	Bits

INSTRUCTION SET — BIT MAPPED

2nd Byte of 2 Byte DD .. Codes
or 2nd Byte of 3 Byte DD .. displ Codes

Bits	.... .000	.... .001	.... .010	.... .011
0000 0...	00	01	02	03
0000 1...	08	09 ADD IX,BC	0A	0B
0001 0...	10	11	12	13
0001 1...	18	19 ADD IX,DE	1A	1B
0010 0...	20	21 LD IX,nn	22 LD (nn),IX	23 INC IX
0010 1...	28	29 ADD IX,IX	2A LD IX,(nn)	2B DEC IX
0011 0...	30	31	32	33
0011 1...	38	39 ADD IX,SP	3A	3B
0100 0...	40	41	42	43
0100 1...	48	49	4A	4B
0101 0...	50	51	52	53
0101 1...	58	59	5A	5B
0110 0...	60 LD HX,B	61 LD HX,C	62 LD HX,D	63 LD HX,E
0110 1...	68 LD LX,B	69 LD LX,C	6A LD LX,D	6B LD LX,E
0111 0...	70 LD (IX+d),B	71 LD (IX+d),C	72 LD (IX+d),D	73 LD (IX+d),E
0111 1...	78	79	7A	7B
1000 0...	80	81	82	83
1000 1...	88	89	8A	8B
1001 0...	90	91	92	93
1001 1...	98	99	9A	9B
1010 0...	A0	A1	A2	A3
1010 1...	A8	A9	AA	AB
1011 0...	B0	B1	B2	B3
1011 1...	B8	B9	BA	BB
1100 0...	C0	C1	C2	C3
1100 1...	C8	C9	CA	CB
1101 0...	D0	D1	D2	D3
1101 1...	D8	D9	DA	DB
1110 0...	E0	E1 POP IX	E2	E3 EX (SP),IX
1110 1...	E8	E9 JP (IX)	EA	EB
1111 0...	F0	F1	F2	F3
1111 1...	F8	F9 LD SP,IX	FA	FB
Bits	.... .000	.... .001	.... .010	.... .011

INSTRUCTION SET - BIT MAPPED

2nd Byte of 2 Byte DD .. Codes
or 2nd Byte of 3 Byte DD .. displ Codes

.... .100	.... .101	.... .110	.... .111	Bits
04	05	06	07	0000 0...
0C	0D	0E	0F	0000 1...
14	15	16	17	0001 0...
1C	1D	1E	1F	0001 1...
24 INC HX	25 DEC HX	26 LD HX,v	27	0010 0...
2C INC LX	2D DEC LX	2E LD LX,v	2F	0010 1...
34 INC (IX+d)	35 DEC (IX+d)	36 LD (IX+d),v	37	0011 0...
3C	3D	3E	3F	0011 1...
44 LD B,HX	45 LD B,LX	46 LD B,(IX+d)	47	0100 0...
4C LD C,HX	4D LD C,LX	4E LD C,(IX+d)	4F	0100 1...
54 LD D,HX	55 LD D,LX	56 LD D,(IX+d)	57	0101 0...
5C LD E,HX	5D LD E,LX	5E LD E,(IX+d)	5F	0101 1...
64 LD HX,HX	65 LD HX,LX	66 LD H,(IX+d)	67 LD HX,A	0110 0...
6C LD LX,HX	6D LD LX,LX	6E LD L,(IX+d)	6F LD LX,A	0110 1...
74 LD (IX+d),H	75 LD (IX+d),L	76	77 LD (IX+d),A	0111 0...
7C LD A,HX	7D LD A,LX	7E LD A,(IX+d)	7F	0111 1...
84 ADD A,HX	85 ADD A,LX	86 ADD A,(IX+d)	87	1000 0...
8C ADC A,HX	8D ADC A,LX	8E ADC A,(IX+d)	8F	1000 1...
94 SUB HX	95 SUB LX	96 SUB (IX+d)	97	1001 0...
9C SBC A,HX	9D SBC A,LX	9E SBC A,(IX+d)	9F	1001 1...
A4 AND HX	A5 AND LX	A6 AND (IX+d)	A7	1010 0...
AC XOR HX	AD XOR LX	AE XOR (IX+d)	AF	1010 1...
B4 OR HX	B5 OR LX	B6 OR (IX+d)	B7	1011 0...
BC CP HX	BD CP LX	BE CP (IX+d)	BF	1011 1...
C4	C5	C6	C7	1100 0...
CC	CD	CE	CF	1100 1...
D4	D5	D6	D7	1101 0...
DC	DD	DE	DF	1101 1...
E4	E5 PUSH IX	E6	E7	1110 0...
EC	ED	EE	EF	1110 1...
F4	F5	F6	F7	1111 0...
FC	FD	FE	FF	1111 1...
.... .100	.... .101	.... .110	.... .111	Bits

INSTRUCTION SET - BIT MAPPED

4th Byte of 4 Byte DD CB displ .. Codes

Bits	.... .000	.... .001	.... .010	.... .011
0000 0...	00	01	02	03
0000 1...	08	09	0A	0B
0001 0...	10	11	12	13
0001 1...	18	19	1A	1B
0010 0...	20	21	22	23
0010 1...	28	29	2A	2B
0011 0...	30	31	32	33
0011 1...	38	39	3A	3B
0100 0...	40	41	42	43
0100 1...	48	49	4A	4B
0101 0...	50	51	52	53
0101 1...	58	59	5A	5B
0110 0...	60	61	62	63
0110 1...	68	69	6A	6B
0111 0...	70	71	72	73
0111 1...	78	79	7A	7B
1000 0...	80	81	82	83
1000 1...	88	89	8A	8B
1001 0...	90	91	92	93
1001 1...	98	99	9A	9B
1010 0...	A0	A1	A2	A3
1010 1...	A8	A9	AA	AB
1011 0...	B0	B1	B2	B3
1011 1...	B8	B9	BA	BB
1100 0...	C0	C1	C2	C3
1100 1...	C8	C9	CA	CB
1101 0...	D0	D1	D2	D3
1101 1...	D8	D9	DA	DB
1110 0...	E0	E1	E2	E3
1110 1...	E8	E9	EA	EB
1111 0...	F0	F1	F2	F3
1111 1...	F8	F9	FA	FB
Bits	.... .000	.... .001	.... .010	.... .011

INSTRUCTION SET - BIT MAPPED

4th Byte of 4 Byte DD CB displ .. Codes

.... .100	.... .101	.... .110	.... .111	Bits
04	05	06 RLC (IX+d)	07	0000 0...
0C	0D	0E RRC (IX+d)	0F	0000 1...
14	15	16 RL (IX+d)	17	0001 0...
1C	1D	1E RR (IX+d)	1F	0001 1...
24	25	26 SLA (IX+d)	27	0010 0...
2C	2D	2E SRA (IX+d)	2F	0010 1...
34	35	36 SLI (IX+d)	37	0011 0...
3C	3D	3E SRL (IX+d)	3F	0011 1...
44	45	46 BIT 0,(IX+d)	47	0100 0...
4C	4D	4E BIT 1,(IX+d)	4F	0100 1...
54	55	56 BIT 2,(IX+d)	57	0101 0...
5C	5D	5E BIT 3,(IX+d)	5F	0101 1...
64	65	66 BIT 4,(IX+d)	67	0110 0...
6C	6D	6E BIT 5,(IX+d)	6F	0110 1...
74	75	76 BIT 6,(IX+d)	77	0111 0...
7C	7D	7E BIT 7,(IX+d)	7F	0111 1...
84	85	86 RES 0,(IX+d)	87	1000 0...
8C	8D	8E RES 1,(IX+d)	8F	1000 1...
94	95	96 RES 2,(IX+d)	97	1001 0...
9C	9D	9E RES 3,(IX+d)	9F	1001 1...
A4	A5	A6 RES 4,(IX+d)	A7	1010 0...
AC	AD	AE RES 5,(IX+d)	AF	1010 1...
B4	B5	B6 RES 6,(IX+d)	B7	1011 0...
BC	BD	BE RES 7,(IX+d)	BF	1011 1...
C4	C5	C6 SET 0,(IX+d)	C7	1100 0...
CC	CD	CE SET 1,(IX+d)	CF	1100 1...
D4	D5	D6 SET 2,(IX+d)	D7	1101 0...
DC	DD	DE SET 3,(IX+d)	DF	1101 1...
E4	E5	E6 SET 4,(IX+d)	E7	1110 0...
EC	ED	EE SET 5,(IX+d)	EF	1110 1...
F4	F5	F6 SET 6,(IX+d)	F7	1111 0...
FC	FD	FE SET 7,(IX+d)	FF	1111 1...
.... .100	..... .101	.... .110	.... .111	Bits

INSTRUCTION SET - BIT MAPPED

2nd Byte of 2 Byte ED .. Codes

Bits	.... .000	.... .001	.... .010	.... .011
0000 0...	00	01	02	03
0000 1...	08	09	0A	0B
0001 0...	10	11	12	13
0001 1...	18	19	1A	1B
0010 0...	20	21	22	23
0010 1...	28	29	2A	2B
0011 0...	30	31	32	33
0011 1...	38	39	3A	3B
0100 0...	40 IN B,(C)	41 OUT (C),B	42 SBC HL,BC	43 LD (nn),B
0100 1...	48 IN C,(C)	49 OUT (C),C	4A ADC HL,BC	4B LD BC,(nn)
0101 0...	50 IN D,(C)	51 OUT (C),D	52 SBC HL,DE	53 LD (nn),D
0101 1...	58 IN E,(C)	59 OUT (C),E	5A ADC HL,DE	5B LD DE,(nn)
0110 0...	60 IN H,(C)	61 OUT (C),H	62 SBC HL,HL	63
0110 1...	68 IN L,(C)	69 OUT (C),L	6A ADC HL,HL	6B
0111 0...	70	71	72 SBC HL,SP	73 LD (nn),SP
0111 1...	78 IN A,(C)	79 OUT (C),A	7A ADC HL,SP	7B LD SP,(nn)
1000 0...	80	81	82	83
1000 1...	88	89	8A	8B
1001 0...	90	91	92	93
1001 1...	98	99	9A	9B
1010 0...	A0 LDI	A1 CPI	A2 INI	A3 OUTI
1010 1...	A8 LDD	A9 CPD	AA IND	AB OUTD
1011 0...	B0 LDIR	B1 CPIR	B2 INIR	B3 OTIR
1011 1...	B8 LDDR	B9 CPDR	BA INDR	BB OTDR
1100 0...	C0	C1	C2	C3
1100 1...	C8	C9	CA	CB
1101 0...	D0	D1	D2	D3
1101 1...	D8	D9	DA	DB
1110 0...	E0	E1	E2	E3
1110 1...	E8	E9	EA	EB
1111 0...	F0	F1	F2	F3
1111 1...	F8	F9	FA	FB
Bits	.... .000	.... .001	.... .010	.... .011

INSTRUCTION SET — BIT MAPPED

2nd Byte of 2 Byte ED .. Codes

.... .100	.... .101	.... .110	.... .111	Bits
04	05	06	07	0000 0...
0C	0D	0E	0F	0000 1...
14	15	16	17	0001 0...
1C	1D	1E	1F	0001 1...
24	25	26	27	0010 0...
2C	2D	2E	2F	0010 1...
34	35	36	37	0011 0...
3C	3D	3E	3F	0011 1...
44 NEG	45 RETN	46 IM 0	47 LD I,A	0100 0...
4C	4D RETI	4E	4F LD R,A	0100 1...
54	55	56 IM 1	57 LD A,I	0101 0...
5C	5D	5E IM 2	5F LD A,R	0101 1...
64	65	66	67 RRD	0110 0...
6C	6D	6E	6F RLD	0110 1...
74	75	76	77	0111 0...
7C	7D	7E	7F	0111 1...
84	85	86	87	1000 0...
8C	8D	8E	8F	1000 1...
94	95	96	97	1001 0...
9C	9D	9E	9F	1001 1...
A4	A5	A6	A7	1010 0...
AC	AD	AE	AF	1010 1...
B4	B5	B6	B7	1011 0...
BC	BD	BE	BF	1011 1...
C4	C5	C6	C7	1100 0...
CC	CD	CE	CF	1100 1...
D4	D5	D6	D7	1101 0...
DC	DD	DE	DF	1101 1...
E4	E5	E6	E7	1110 0...
EC	ED	EE	EF	1110 1...
F4	F5	F6	F7	1111 0...
FC	FD	FE	FF	1111 1...
.... .100	..... .101	.... .110	.... .111	Bits

INSTRUCTION SET - BIT MAPPED

2nd Byte of 2 Byte FD .. Codes
or 2nd Byte of 3 Byte FD .. displ Codes

Bits	.... .000	.... .001	.... .010	.... .011
0000 0...	00	01	02	03
0000 1...	08	09 ADD IY,BC	0A	0B
0001 0...	10	11	12	13
0001 1...	18	19 ADD IY,DE	1A	1B
0010 0...	20	21 LD IY,nn	22 LD (nn),IY	23 INC IY
0010 1...	28	29 ADD IY,IY	2A LD IY,(nn)	2B DEC IY
0011 0...	30	31	32	33
0011 1...	38	39 ADD IY,SP	3A	3B
0100 0...	40	41	42	43
0100 1...	48	49	4A	4B
0101 0...	50	51	52	53
0101 1...	58	59	5A	5B
0110 0...	60 LD HY,B	61 LD HY,C	62 LD HY,D	63 LD HY,E
0110 1...	68 LD LY,B	69 LD LY,C	6A LD LY,D	6B LD LY,E
0111 0...	70 LD (IY+d),B	71 LD (IY+d),C	72 LD (IY+d),D	73 LD (IY+d),E
0111 1...	78	79	7A	7B
1000 0...	80	81	82	83
1000 1...	88	89	8A	8B
1001 0...	90	91	92	93
1001 1...	98	99	9A	9B
1010 0...	A0	A1	A2	A3
1010 1...	A8	A9	AA	AB
1011 0...	B0	B1	B2	B3
1011 1...	B8	B9	BA	BB
1100 0...	C0	C1	C2	C3
1100 1...	C8	C9	CA	CB
1101 0...	D0	D1	D2	D3
1101 1...	D8	D9	DA	DB
1110 0...	E0	E1 POP IY	E2	E3 EX (SP),IY
1110 1...	E8	E9 JP (IY)	EA	EB
1111 0...	F0	F1	F2	F3
1111 1...	F8	F9 LD SP,IY	FA	FB
Bits	.... .000	.... .001	.... .010	.... .011

INSTRUCTION SET - BIT MAPPED

2nd Byte of 2 Byte FD .. Codes
or 2nd Byte of 3 Byte FD .. displ Codes

.... .100	.... .101	.... .110	.... .111	Bits
04	05	06	07	0000 0...
0C	0D	0E	0F	0000 1...
14	15	16	17	0001 0...
1C	1D	1E	1F	0001 1...
24 INC HY	25 DEC HY	26 LD HY,v	27	0010 0...
2C INC LY	2D DEC LY	2E LD LY,v	2F	0010 1...
34 INC (IY+d)	35 DEC (IY+d)	36 LD (IY+d),v	37	0011 0...
3C	3D	3E	3F	0011 1...
44 LD B,HY	45 LD B,LY	46 LD B,(IY+d)	47	0100 0...
4C LD C,HY	4D LD C,LY	4E LD C,(IY+d)	4F	0100 1...
54 LD D,HY	55 LD D,LY	56 LD D,(IY+d)	57	0101 0...
5C LD E,HY	5D LD E,LY	5E LD E,(IY+d)	5F	0101 1...
64 LD HY,HY	65 LD HY,LY	66 LD H,(IY+d)	67 LD HY,A	0110 0...
6C LD LY,HY	6D LD LY,LY	6E LD L,(IY+d)	6F LD LY,A	0110 1...
74 LD (IY+d),H	75 LD (IY+d),L	76	77 LD (IY+d),A	0111 0...
7C LD A,HY	7D LD A,LY	7E LD A,(IY+d)	7F	0111 1...
84 ADD A,HY	85 ADD A,LY	86 ADD A,(IY+d)	87	1000 0...
8C ADC A,HY	8D ADC A,LY	8E ADC A,(IY+d)	8F	1000 1...
94 SUB HY	95 SUB LY	96 SUB (IY+d)	97	1001 0...
9C SBC A,HY	9D SBC A,LY	9E SBC A,(IY+d)	9F	1001 1...
A4 AND HY	A5 AND LY	A6 AND (IY+d)	A7	1010 0...
AC YOR HY	AD XOR LY	AE XOR (IY+d)	AF	1010 1...
B4 OR HY	B5 OR LY	B6 OR (IY+d)	B7	1011 0...
BC CP HY	BD CP LY	BE CP (IY+d)	BF	1011 1...
C4	C5	C6	C7	1100 0...
CC	CD	CE	CF	1100 1...
D4	D5	D6	D7	1101 0...
DC	DD	DE	DF	1101 1...
E4	E5 PUSH IY	E6	E7	1110 0...
EC	ED	EE	EF	1110 1...
F4	F5	F6	F7	1111 0...
FC	FD	FE	FF	1111 1...
.... .100	..... .101	.... .110	.... .111	Bits

INSTRUCTION SET — BIT MAPPED

4th Byte of 4 Byte FD CB displ .. Codes

Bits	.... .000	.... .001	.... .010	.... .011
0000 0...	00	01	02	03
0000 1...	08	09	0A	0B
0001 0...	10	11	12	13
0001 1...	18	19	1A	1B
0010 0...	20	21	22	23
0010 1...	28	29	2A	2B
0011 0...	30	31	32	33
0011 1...	38	39	3A	3B
0100 0...	40	41	42	43
0100 1...	48	49	4A	4B
0101 0...	50	51	52	53
0101 1...	58	59	5A	5B
0110 0...	60	61	62	63
0110 1...	68	69	6A	6B
0111 0...	70	71	72	73
0111 1...	78	79	7A	7B
1000 0...	80	81	82	83
1000 1...	88	89	8A	8B
1001 0...	90	91	92	93
1001 1...	98	99	9A	9B
1010 0...	A0	A1	A2	A3
1010 1...	A8	A9	AA	AB
1011 0...	B0	B1	B2	B3
1011 1...	B8	B9	BA	BB
1100 0...	C0	C1	C2	C3
1100 1...	C8	C9	CA	CB
1101 0...	D0	D1	D2	D3
1101 1...	D8	D9	DA	DB
1110 0...	E0	E1	E2	E3
1110 1...	E8	E9	EA	EB
1111 0...	F0	F1	F2	F3
1111 1...	F8	F9	FA	FB
Bits	.... .000	.... .001	.... .010	.... .011

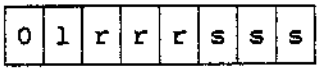
INSTRUCTION SET — BIT MAPPED

4th Byte of 4 Byte FD CB displ .. Codes

.... .100	.... .101	.... .110	.... .111	Bits
04	05	06 RLC (IY+d)	07	0000 0...
0C	0D	0E RRC (IY+d)	0F	0000 1...
14	15	16 RL (IY+d)	17	0001 0...
1C	1D	1E RR (IY+d)	1F	0001 1...
24	25	26 SLA (IY+d)	27	0010 0...
2C	2D	2E SRA (IY+d)	2F	0010 1...
34	35	36 SLI (IY+d)	37	0011 0...
3C	3D	3E SRL (IY+d)	3F	0011 1...
44	45	46 BIT 0,(IY+d)	47	0100 0...
4C	4D	4E BIT 1,(IY+d)	4F	0100 1...
54	55	56 BIT 2,(IY+d)	57	0101 0...
5C	5D	5E BIT 3,(IY+d)	5F	0101 1...
64	65	66 BIT 4,(IY+d)	67	0110 0...
6C	6D	6E BIT 5,(IY+d)	6F	0110 1...
74	75	76 BIT 6,(IY+d)	77	0111 0...
7C	7D	7E BIT 7,(IY+d)	7F	0111 1...
84	85	86 RES 0,(IY+d)	87	1000 0...
8C	8D	8E RES 1,(IY+d)	8F	1000 1...
94	95	96 RES 2,(IY+d)	97	1001 0...
9C	9D	9E RES 3,(IY+d)	9F	1001 1...
A4	A5	A6 RES 4,(IY+d)	A7	1010 0...
AC	AD	AE RES 5,(IY+d)	AF	1010 1...
B4	B5	B6 RES 6,(IY+d)	B7	1011 0...
BC	BD	BE RES 7,(IY+d)	BF	1011 1...
C4	C5	C6 SET 0,(IY+d)	C7	1100 0...
CC	CD	CE SET 1,(IY+d)	CF	1100 1...
D4	D5	D6 SET 2,(IY+d)	D7	1101 0...
DC	DD	DE SET 3,(IY+d)	DF	1101 1...
E4	E5	E6 SET 4,(IY+d)	E7	1110 0...
EC	ED	EE SET 5,(IY+d)	EF	1110 1...
F4	F5	F6 SET 6,(IY+d)	F7	1111 0...
FC	FD	FE SET 7,(IY+d)	FF	1111 1...
.... .100	..... .101	.... .110	.... .111	Bits

8 BIT LOADS

LD r,s



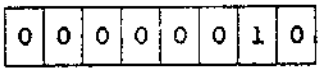
The value in Operand 's' is loaded into 'r'. No Flags affected.

LD r,v

= rr + "v" byte No flags affected
Codes in last column on right.

Reg r Bits \ Reg s	B 000	C 001	D 010	E 011	H 100	L 101	(HL) 110	A 111	(IX+d)	(IY+d)	v
B = 000	40	41	42	43	44	45	46	47	DD-46	FD-46	06-v
C = 001	48	49	4A	4B	4C	4D	4E	4F	DD-4E	FD-4E	0E-v
D = 010	50	51	52	53	54	55	56	57	DD-56	FD-56	16-v
E = 011	58	59	5A	5B	5C	5D	5E	5F	DD-5E	FD-5E	1E-v
H = 100	60	61	62	63	64	65	66	67	DD-66	FD-66	26-v
L = 101	68	69	6A	6B	6C	6D	6E	6F	DD-6E	FD-6E	2E-v
(HL)= 110	70	71	72	73	74	75		77			36-v
A = 111	78	79	7A	7B	7C	7D	7E	7F	DD-7E	FD-7E	3E-v
(IX+d)= 110	DD-70	DD-71	DD-72	DD-73	DD-74	DD-75		DD-77			DD-36-v
(IY+d)= 110	FD-70	FD-71	FD-72	FD-73	FD-74	FD-75		FD-77			FD-36-v

LD (BC),A

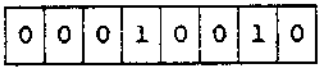


= 02 No flags affected

LD A,(BC)

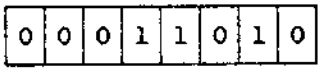
= 0A No flags affected

LD (DE),A



= 12 No flags affected

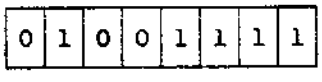
LD A,(DE)



= 1A No flags affected

LD R,A

ED byte +

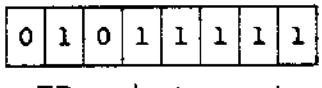


No flags affected

= ED 4F

LD I,A

ED byte +

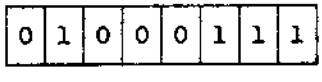


No flags affected

= ED 5F

LD A,R

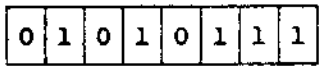
ED byte +



= ED 47

LD A,I

ED byte +



= ED 57

-- Flags --

S set if Reg is negative
Z set if Reg is zero
H & N are reset
P/V = contents of Interrupt IFF2.
C is not affected

16 BIT LOADS & EXCHANGE

LD rr,nn — 0 0 r r 0 0 0 1 — + 2 Address bytes

No flags affected on this page

rr – BC = 00 01-nn; DE = 01 11-nn; HL = 10 21-nn; SP = 11 31-nn; IX = 10 DD-21-nn; IY = 10 FD-21-nn

LD hl,(nn) — 0 0 1 0 1 0 1 0 — + 2 Byte address

hl – HL = 2A-nn; IX = DD-2A-nn; IY = FD-2A-nn

LD (nn),hl — 0 0 1 0 0 0 1 0 — + 2 Byte address

hl – HL = 22-nn; IX = DD-22-nn; IY = FD-22-nn

LD rr,(nn) — ED byte + 0 1 r r 0 0 1 1

rr – BC = 00 ED-43-nn; DE = 01 ED-53-nn; SP = 11 ED-73-nn

LD (nn),rr — ED byte + 0 1 r r 1 0 1 1

rr – BC = 00 ED-4B-nn; DE = 01 ED-5B-nn; SP = 11 ED-7B-nn

LD SP,hl — 1 1 1 1 1 0 0 1

hl – HL = F9; IX = DD-F9; IY = FD-F9

		POP	PUSH
POP rr / PUSH rr	rr – BC = 00	C1	C5
1 1 r r 0 0 0 1 / 1 1 r r 0 1 0 1	DE = 01	D1	D5
	HL = 10	E1	E5
	AF = 11	F1	F5
	IX = 10	DD-E1	DD-E5
	IY = 10	FD-E1	FD-E5

EX DE,HL — 1 1 1 0 1 0 1 1 = EB — The 2byte values in HL and DE are exchanged.

EX AF,AF' — 0 0 0 0 1 0 0 0 = 08 — The main and Primed sets of AF are exchanged.

EXX — 1 1 0 1 1 0 0 1 = D9 — The BC, DE & HL Register Pairs are exchanged for their Primed counterparts.

EX (SP),hl — 1 1 1 0 0 0 1 1

hl – HL = E3; IX = DD-E3; IY = FD-E3

The first Address on the Stack is Exchanged with the contents of (HL).

8 BIT ARITHMETIC INSTRUCTIONS

--- Flags ADD/ADC/SUB/SBC ---
S = set if result is negative
Z = set if result is zero
H = set if carry from Bit 3 - ADD/ADC
borrow from Bit 4 - SUB/SBC
P/V=set if overflow 8 bits
N = set if SUB/SBC
reset if ADD/ADC
C = set if carry from Bit 7 - ADD/ADC
borrow for Bit 7 - SUB/SBC

	Reg	Bits	ADD	ADC	SUB	SBC
rrr -	B	000	80	88	90	98
	C	001	81	89	91	99
	D	010	82	8A	92	9A
	E	011	83	8B	93	9B
	H	100	84	8C	94	9C
	L	101	85	8D	95	9D
	(HL)	110	86	8E	96	9E
	A	111	87	8F	97	9F
	(IX+d)	110	DD-86	DD-8E	DD-96	DD-9E
	(IY+d)	110	FD-86	FD-8E	FD-96	FD-9E

ADD	A,r	1 0 0 0 0 r r r		The contents of r are added to the register A. Result is in A.
ADD	A,v	1 1 0 0 0 1 1 0	=	C6H + byte to be added Flags as above
ADC	A,r	1 0 0 0 1 r r r	=	Codes as above Flags as above
ADC	A,v	1 0 0 0 1 1 1 0	=	CEH + Byte to be added Flags as above
SUB	r	1 0 0 1 0 r r r	=	Codes as above Flags as above
SUB	v	1 1 0 1 0 1 1 0	=	D6 + byte to be subtracted Flags as above
SBC	A,r	1 0 0 1 1 r r r	=	Codes as above Flags as above
SBC	A,v	1 0 0 1 1 1 1 0	=	DEH + Byte to be subtracted Flags as above
INC	r	0 0 r r r 1 0 0		
DEC	r	0 0 r r r 1 0 1		

---- Flags INC/DEC ----
S = set if negative
Z = set if result is zero
H = set if carry/borrow from Bit 3
P/V= set if r was 7FH(INC)/80H(DEC) before
N = reset
C = no effect

	Reg	Bits	Inc	Dec
rrr -	B	000	04	05
	C	001	0C	0D
	D	010	14	15
	E	011	1C	1D
	H	100	24	25
	L	101	2C	2D
	(HL)	110	34	35
	A	111	3C	3D
	(IX+d)	110	DD-34	DD-35
	(IY+d)	110	FD-34	FD-35

8 BIT LOGICAL INSTRUCTIONS

In these instructions a logical AND, OR, exclusive OR, or Compare operation is done Bit by Bit between the byte in r or v and the byte in the accumulator (A). The result is stored in (A).

	Reg	Bits	AND	XOR	OR	CP
rrr -	B =	000	A0	A8	B0	B8
	C =	001	A1	A9	B1	B9
	D =	010	A2	AA	B2	BA
	E =	011	A3	AB	B3	BB
	H =	100	A4	AC	B4	BC
	L =	101	A5	AD	B5	BD
	(HL)=	110	A6	AE	B6	BE
	A =	111	A7	AF	B7	BF
	(IX+d)=	110	DD-A6	DD-AE	DD-B6	DD-BE
	(IY+d)=	110	FD-A6	FD-AE	FD-B6	FD-BE
	v =		E6-v	EE-v	F6-v	FE-v

--- Flags ---
S = set if result is negative
Z = set if result is zero
H = set
P/V=set if parity even
N = reset
C = reset

AND r | 1 | 0 | 1 | 0 | 0 | r | r | r | = codes as above
Flags as above

AND v | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | = E6H + byte to be used to AND
Flags as above

XOR r | 1 | 0 | 1 | 0 | 1 | r | r | r | = codes as above
Flags as above

XOR v | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | = EEH + byte to be used to XOR
Flags as above

OR r | 1 | 0 | 1 | 1 | 0 | r | r | r | = codes as above
Flags as above

OR v | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | = F6H + byte to be used to OR
Flags as above

CP r | 1 | 0 | 1 | 1 | 1 | r | r | r | = codes as above

CP v | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

= FE +
+ Byte to be used for Compare

---- Flags ----
S = set if result negative
Z = set if result is zero
H = set if no borrow from Bit 4
P/V= set if overflow
N = set
C = set if borrow.

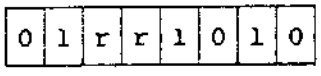
16 BIT ARITHMETIC INSTRUCTIONS

ADD HL,rr

Regs	Bits	ADD	ADC
rr - BC	= 00	09	ED-4A
DE	= 01	19	ED-5A
HL	= 10	29	ED-6A
SP	= 11	39	ED-7A

---- Flags- ADD ----
S, Z and P/V not affected
H = set if carry out of bit 11
N = reset
C = set if carry out of bit 15

ADC HL,rr ED Byte +



Codes as above

---- Flags ADC ----
S = set if negative
Z = set if zero
H = set if carry from Bit 11
P/V = set if overflow
N = reset
C = set if carry from Bit 15

ADD IX,rr DD byte +
ADD IY,rr FD byte +

Flags as with ADD above

		IX	IY
rr -	BC = 00	DD-09	FD-09
	DE = 01	DD-19	FD-19
	IX = 10	DD-29	
	IY = 10		FD-29
	SP = 11	DD-39	FD-39

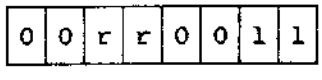
SBC HL,rr ED byte +

'rr' is subtracted from HL

		SBC
rr -	BC = 00	ED-42
	DE = 01	ED-52
	HL = 10	ED-62
	SP = 11	ED-72

---- Flags ----
S = set if result negative
Z = set if result zero
H = set if borrow from bit 12
P/V = set if overflow
N = reset
C = set if borrow (rr > HL)

INC rr



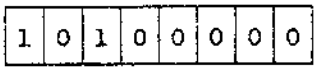
DEC rr

No flags affected

		INC	DEC
rr -	BC = 00	03	0B
	DE = 01	13	1B
	HL = 10	23	2B
	SP = 11	33	3B
	IX = 10	DD-23	FD-2B
	IY = 10	FD-23	FD-2B

BLOCK TRANSFER & SEARCH INSTRUCTIONS

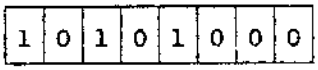
LDI — ED byte +



= A0

Flags
S, Z and C are not affected
H and N are reset
P/V reset if B=0 else set

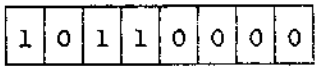
LDD — ED byte +



= A8

S, Z and C are not affected
H and N are reset
P/V reset if B=0 else set

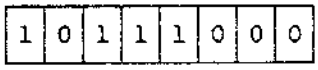
LDIR — ED byte +



= B0

S, Z and C are not affected
H, P/V and N are reset

LDDR — ED byte +

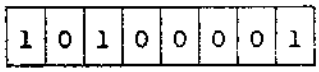


= B8

S, Z and C are not affected
H, P/V and N are reset

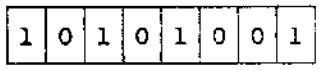
The contents of (HL) is transferred to (DE).
HL & DE are incremented, BC is decremented (LDI & LDIR)
HL, DE & BC are decremented (LDD & LDDR)
If BC <> 0 the instruction is repeated (LDIR & LDDR).

CPI — ED byte +



= A1

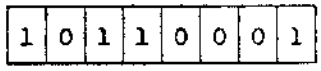
CPD — ED byte +



= A9

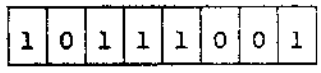
S = set if is negative
Z = set if (HL)=A else reset
H = reset if borrow from bit 4
P/V = reset if BC=0 else set
N = set
C = no effect
Flags as above

CPIR — ED byte +



= B1

CPDR — ED byte +



= B9

S = set if is negative
Z = set if (HL)=A else reset
H = reset if borrow from bit 4
P/V = reset if BC=0 else set
N = set
C = no effect
Flags as above

The contents of (HL) is compared with A.
BC is decremented, HL is incremented (CPI & CPIR)
BC and HL are decremented (CPD & CPDR).
If BC <> 0 the instruction is repeated (CPIR & CPDR)

BIT RESET & SET INSTRUCTIONS

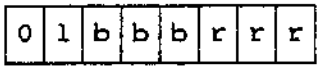
The instructions on these pages are bit manipulations.

BIT is the only one of the 3 bit manipulation codes that affects a flag.

After execution the Zero flag will contain the complement of the tested bit in the relevant register or byte to which the HL, IX+d or IY+d registers are pointing.

It sets the Zero flag if the bit tested is reset.

BIT b,r CB byte +



Codes as below

--- Flags ---

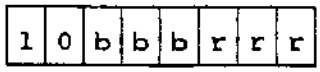
S - unknown
Z - set if bit tested is Zero else reset
H - set
P/V - unknown
N - reset
C - Not affected

bbb Bits	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
Reg Bits								
rrr-B =000	CB-40	CB-48	CB-50	CB-58	CB-60	CB-68	CB-70	CB-78
C =001	CB-41	CB-49	CB-51	CB-59	CB-61	CB-69	CB-71	CB-79
D =010	CB-42	CB-4A	CB-52	CB-5A	CB-62	CB-6A	CB-72	CB-7A
E =011	CB-43	CB-4B	CB-53	CB-5B	CB-63	CB-6B	CB-73	CB-7B
H =100	CB-44	CB-4C	CB-54	CB-5C	CB-64	CB-6C	CB-74	CB-7C
L =101	CB-45	CB-4D	CB-55	CB-5D	CB-65	CB-6D	CB-75	CB-7D
(HL)=110	CB-46	CB-4E	CB-56	CB-5E	CB-66	CB-6E	CB-76	CB-7E
A =111	CB-47	CB-4F	CB-57	CB-5F	CB-67	CB-6F	CB-77	CB-7F
(IX+d)= DD+	-CB-46	-CB-4E	-CB-56	-CB-5E	-CB-66	-CB-6E	-CB-76	-CB-7E
(IY+d)= FD+	-CB-46	-CB-4E	-CB-56	-CB-5E	-CB-66	-CB-6E	-CB-76	-CB-7E

BIT RESET & SET INSTRUCTIONS

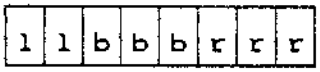
The 2 Instructions on this page are bit manipulations.
They do not affect any flags.
In RESET the code makes the Bit specified 0 in the Byte specified.
In SET the code makes the Bit specified 1 in the Byte specified.

RES b,r CB byte + Codes as below



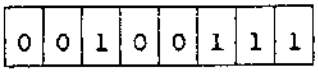
bbb Bits / Reg Bits	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
Rrr-B= 000	CB-80	CB-88	CB-90	CB-98	CB-A0	CB-A8	CB-B0	CB-B8
C= 001	CB-81	CB-89	CB-91	CB-99	CB-A1	CB-A9	CB-B1	CB-B9
D= 010	CB-82	CB-8A	CB-92	CB-9A	CB-A2	CB-AA	CB-B2	CB-BA
E= 011	CB-83	CB-8B	CB-93	CB-9B	CB-A3	CB-AB	CB-B3	CB-BB
H= 100	CB-84	CB-8C	CB-94	CB-9C	CB-A4	CB-AC	CB-B4	CB-BC
L= 101	CB-85	CB-8D	CB-95	CB-9D	CB-A5	CB-AD	CB-B5	CB-BD
(HL)= 110	CB-86	CB-8E	CB-96	CB-9E	CB-A6	CB-AE	CB-B6	CB-BE
A= 111	CB-87	CB-8F	CB-97	CB-9F	CB-A7	CB-AF	CB-B7	CB-BF
(IX+d)= DD+	-CB-86	-CB-8E	-CB-96	-CB-9E	-CB-A6	-CB-AE	-CB-B6	-CB-BE
(IY+d)= FD+	-CB-86	-CB-8E	-CB-96	-CB-9E	-CB-A6	-CB-AE	-CB-B6	-CB-BE

SET b,r CB byte + Codes as below



bbb Bits / Reg Bits	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
Rrr-B= 000	CB-C0	CB-C8	CB-D0	CB-D8	CB-E0	CB-E8	CB-F0	CB-F8
C= 001	CB-C1	CB-C9	CB-D1	CB-D9	CB-E1	CB-E9	CB-F1	CB-F9
D= 010	CB-C2	CB-CA	CB-D2	CB-DA	CB-E2	CB-EA	CB-F2	CB-FA
E= 011	CB-C3	CB-CB	CB-D3	CB-DB	CB-E3	CB-EB	CB-F3	CB-FB
H= 100	CB-C4	CB-CC	CB-D4	CB-DC	CB-E4	CB-EC	CB-F4	CB-FC
L= 101	CB-C5	CB-CD	CB-D5	CB-DD	CB-E5	CB-ED	CB-F5	CB-FD
(HL)= 110	CB-C6	CB-CE	CB-D6	CB-DE	CB-E6	CB-EE	CB-F6	CB-FE
A= 111	CB-C7	CB-CF	CB-D7	CB-DF	CB-E7	CB-EF	CB-F7	CB-FF
(IX+d)= DD+	-CB-C6	-CB-CE	-CB-D6	-CB-DE	-CB-E6	-CB-EE	-CB-F6	-CB-FE
(IY+d)= FD+	-CB-C6	-CB-CE	-CB-D6	-CB-DE	-CB-E6	-CB-EE	-CB-F6	-CB-FE

THE DAA INSTRUCTION

DAA  = 27

This instruction adjusts the accumulator to the Binary Coded Decimal condition after the following operations :-

Addition - ADD, ADC and INC
Subtraction - SUB, SBC, DEC and NEG

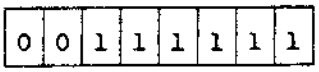
Binary Coded Decimal is the storage of a decimal value in the lower and the upper halves (nybbles) of a byte in binary. For instance, if you stored the decimal number 36 in BCD format there would be a upper nybble of 3 and a lower nybble of 6. The number if interpreted as a hex value would contain 54 decimal but in BCD it contains 36 decimal.

Now if you were to try and add another BCD value to this, say 17 BCD, then the accumulator would contain 4DH or 77 decimal. The DAA instruction adjusts the result to give the correct decimal value for a BCD add. During the addition to give the hex 4D result the Carry flag and the Half Carry flag were reset so neither has a value. Inspecting the table below it can be seen that the 3rd row meets these conditions and so we would add 06H to the result giving 53 as the answer. This is the correct addition of 36 and 17 in decimal in BCD format.

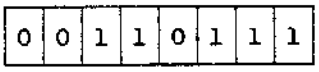
It becomes obvious on examination that all the instruction does is converts the product from hex (base 16) to decimal (base 10) in each nybble.

Operation	Carry before	Bits 7-4 Upper Nyb.	Half Carry before	Bits 3-0 before	Value added	Carry after
ADD or ADC or INC.	0	0 - 9	0	0 - 9	00	0
	0	0 - 8	0	A - F	06	0
	0	0 - 9	1	0 - 3	06	0
	0	A - F	0	0 - 9	60	1
	0	9 - F	0	A - F	66	1
	0	A - F	1	0 - 3	66	1
	1	0 - 2	0	0 - 9	60	1
	1	0 - 2	0	A - F	66	1
	1	0 - 3	1	0 - 3	66	1
SUB or SBC or DEC or NEG.	0	0 - 9	0	0 - 9	00	0
	0	0 - 8	1	6 - F	FA	0
	1	7 - F	0	0 - 9	A0	1
	1	6 - F	1	6 - F	9A	1

GENERAL PURPOSE INSTRUCTIONS

CCF 

= 3F The Carry flag is inverted
The only other flag affected is the
Add/Subtract flag which is reset.

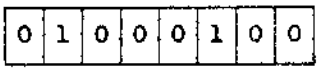
SCF 

= 37 The Carry flag is set and the
Half Carry and Add/Subtract flags are reset.

CPL 

= 2F The contents of each bit in the
Accumulator is inverted (1's complement).

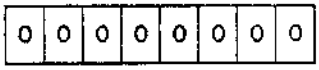
NEG ED byte +



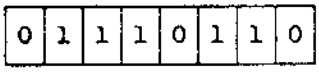
= ED 44

The contents of the accumulator
Is negated (2's complement)
I.E. subtracted from 256 (0).
Note 80H remains 80H.

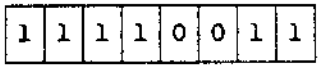
---- Flags ----
S = set if result negative
Z = set if result is zero
H = reset if borrow from Bit 4
P/V = set if A = 80H before
N = set
C = reset if A = 0 before

NOP 

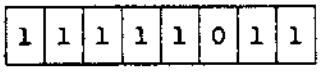
= 00 No operation.

HALT 

= 76 This instruction suspends CPU
operation until an interrupt
or reset is sent.

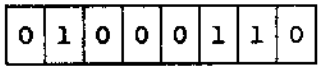
DI 

= F3 This instruction disables the
Maskable interrupt by resetting the
Interrupt enables IFF1 & IFF2.

EI 

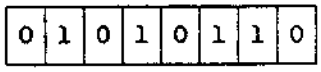
= FB This instruction sets the interrupt
Enables so enabling the maskable interrupt.

IM 0 ED byte +



= ED 46 In this interrupt mode the
Interrupting device can call on the
CPU to execute any instruction.

IM 1 ED byte +



= ED 56 In this mode the CPU will react
To any interrupt by executing an
RST 38H (the interrupt restart).

IM 2 ED byte +

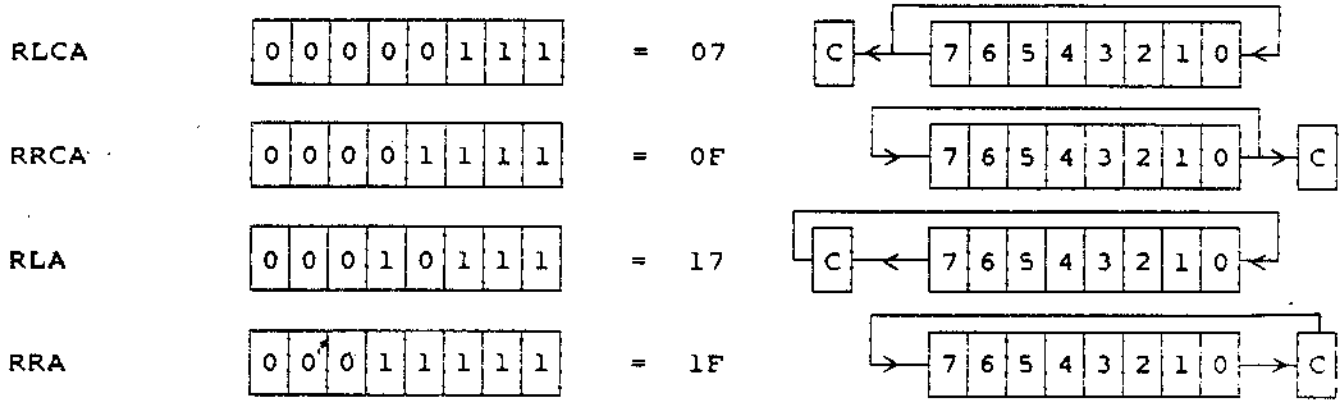
= ED 5E This mode allows an indirect
Call to a memory address which is formed
By the high byte from the I register and the
Low byte from the interrupting device.

ROTATE AND SHIFT INSTRUCTIONS

These 4 instructions apply to the accumulator (A register) only

RLCA - Rotate Accumulator left with branch carry
RRCA - Rotate Accumulator right with branch carry
RLA - Rotate Accumulator left thru carry
RRA - Rotate Accumulator right thru carry

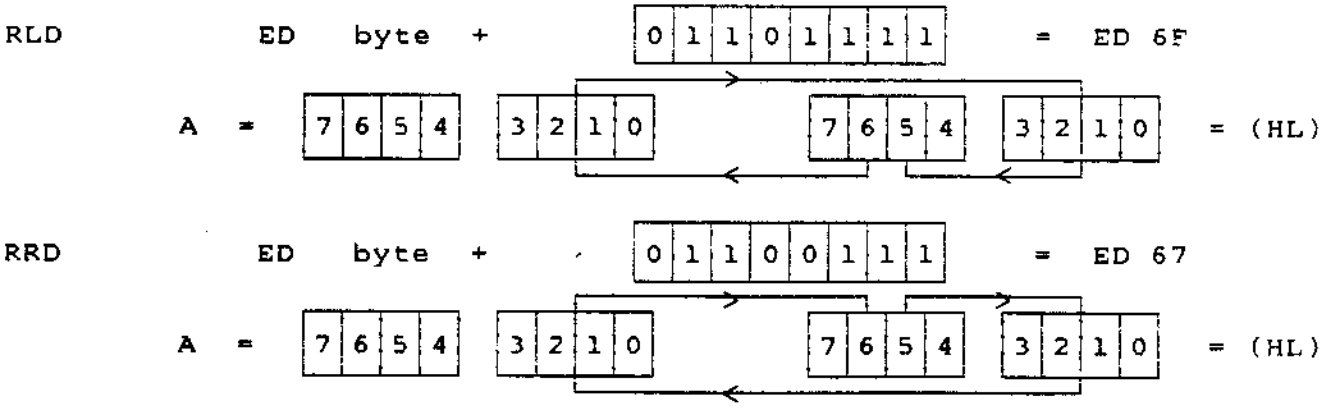
Flags - S,Z & P/V are not affected : H & N are reset.
The carry flag is involved in each case. The value it receives comes from bit 7 (RLCA & RLA) and bit 0 (RRCA & RRA).



These 2 little cuties are very powerful and useful in some situations. With RLD the low nybble of the contents of the memory location pointed to by HL are transferred to the high nybble of (HL) and those of the high nybble are transferred to the low nybble of the A register. The contents of the low nybble of A are transferred back to the low nybble of (HL). This instruction if executed three times restores original values. The shift is to the right instead with RRD.

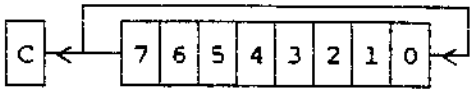
Flags for both RLD & RRD

S - set if A is negative else reset
Z - set if A is zero after operation else reset
P/V - set if parity is even after operation
H & N are reset : C is not affected



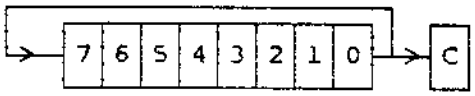
ROTATE AND SHIFT INSTRUCTIONS

RLC CB Byte + | 0 | 0 | 0 | 0 | 0 | r | r | r |



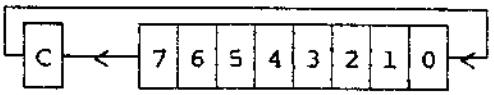
Rotate left with branch carry

RRC CB Byte + | 0 | 0 | 0 | 0 | 1 | r | r | r |



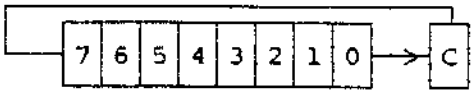
Rotate right with branch carry

RL CB Byte + | 0 | 0 | 0 | 1 | 0 | r | r | r |



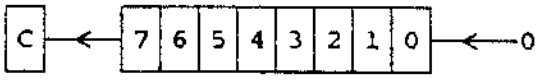
Rotate left thru carry

RR CB Byte + | 0 | 0 | 0 | 1 | 1 | r | r | r |



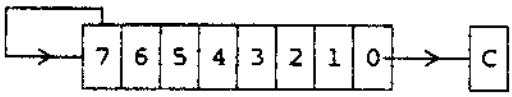
Rotate right thru carry

SLA CB Byte + | 0 | 0 | 1 | 0 | 0 | r | r | r |



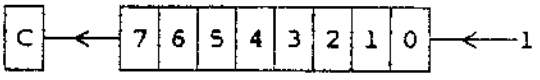
Arithmetic shift left

SRA CB Byte + | 0 | 0 | 1 | 0 | 1 | r | r | r |



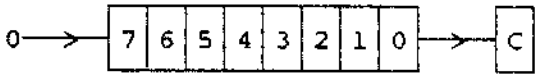
Arithmetic shift right

SLI CB Byte + | 0 | 0 | 1 | 1 | 0 | r | r | r |



Shift left & increment

SRL CB Byte + | 0 | 0 | 1 | 1 | 1 | r | r | r |



Logical shift right

Flags for all 8 instructions same as RLD & RRD except Carry which is affected by data from Bit 7 (RLC,RL,SLA,SLI), Bit 0 (RRC,RR,SRA,SRL).

Reg	Bits	RLC 000	RRC 001	RL 010	RR 011	SLA 100	SRA 101	SLI 110	SRL 111
B	= 000	CB-00	CB-08	CB-10	CB-18	CB-20	CB-28	CB-30	CB-38
C	= 001	CB-01	CB-09	CB-11	CB-19	CB-21	CB-29	CB-31	CB-39
D	= 010	CB-02	CB-0A	CB-12	CB-1A	CB-22	CB-2A	CB-32	CB-3A
E	= 011	CB-03	CB-0B	CB-13	CB-1B	CB-23	CB-2B	CB-33	CB-3B
H	= 100	CB-04	CB-0C	CB-14	CB-1C	CB-24	CB-2C	CB-34	CB-3C
L	= 101	CB-05	CB-0D	CB-15	CB-1D	CB-25	CB-2D	CB-35	CB-3D
(HL)	= 110	CB-06	CB-0E	CB-16	CB-1E	CB-26	CB-2E	CB-36	CB-3E
A	= 111	CB-07	CB-0F	CB-17	CB-1F	CB-27	CB-2F	CB-37	CB-3F
(IX+d)	= DD+	-CB-06	-CB-0E	-CB-16	-CB-1E	-CB-26	-CB-2E	-CB-36	-CB-3E
(IY+d)	= FD+	-CB-06	-CB-0E	-CB-16	-CB-1E	-CB-26	-CB-2E	-CB-36	-CB-3E

JUMP & JUMP RELATIVE INSTRUCTIONS

JP nn | 1 1 0 0 0 0 1 1 | = C3 + 2 byte address

No flags affected on this Page

JP (hl) | 1 1 1 0 1 0 0 1 |

No flags affected

hl - HL = E9
IX = DD-E9
IY = FD-E9

JP cc,nn | 1 1 c c c 0 1 0 |

+ 2 byte address

The jump occurs according to the condition of the flag used for the test.

ccc - non zero NZ = 000 C2+addr
zero Z = 001 CA+addr
no carry NC = 010 D2+addr
carry C = 011 DA+addr
parity odd PO = 100 E2+addr
parity even PE = 101 EA+addr
sign positive P = 110 F2+addr
sign negative M = 111 FA+addr

JR j | 0 0 0 1 1 0 0 0 | = 18 + jump displacement byte

JR NZ,j | 0 0 1 0 0 0 0 0 | = 20 + jump displacement byte

JR Z,j | 0 0 1 0 1 0 0 0 | = 28 + jump displacement byte

JR NC,j | 0 0 1 1 0 0 0 0 | = 30 + jump displacement byte

JR C,j | 0 0 1 1 1 0 0 0 | = 38 + jump displacement byte

If the condition required is met, the Jump Relative occurs by the Program Counter having the 8 Bit signed displacement value of j added.

DJNZ j | 0 0 0 1 0 0 0 0 | = 10 + jump displacement byte

In this case the value in the register B is used to determine the number of times the program loops to repeat the execution of the same portion of code. This is accomplished by decrementing the B register each loop until zero when it sets the Z flag and the execution falls through to the next instruction.

CALL & RETURN INSTRUCTIONS

CALL nn | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | = CD + 2 Byte address

The contents of the Program Counter is pushed onto the stack. The 16 bit address nn is loaded into the PC. Execution resumes at this address. No flags are affected.

CALL cc,nn | 1 | 1 | c | c | c | 1 | 0 | 0 |

RET cc | 1 | 1 | c | c | c | 0 | 0 | 0 |

No flags affected

cc -			RET	CALL
	Non zero	NZ = 000	C0	C4
	Zero	Z = 001	C8	CC
	Non carry	NC = 010	D0	D4
	Carry	C = 011	D8	DC
	Parity odd	PO = 100	E0	E4
	Parity even	PE = 101	E8	EC
	Sign pos.	P = 110	F0	F4
	Sign neg.	M = 111	F8	FC

RET | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | = C9 No flags affected

The current contents of the stack are POPed into the Program Counter and execution resumes at that address. RET = POP PC.

RETI ED byte + = ED 4D Return from Interrupt

0	1	0	0	1	1	0	1	No flags affected

This instruction is used after an interrupt routine to restore control to the interrupted program by POPing the stack contents into the Program Counter.

RETN ED byte + = ED 45 Return from non maskable int.

0	1	0	0	0	1	0	1	No flags affected

This is the return signal from a service routine for a non-maskable interrupt. This instruction executes an unconditional return to the interrupted program. The previous value in IFF1 is restored from IFF2.

RST tt | 1 | 1 | t | t | t | 1 | 1 | 1 |

ttt -			
000	=	C7	= 0000H
001	=	CF	= 0008H
010	=	D7	= 0010H
011	=	DF	= 0018H
100	=	E7	= 0020H
101	=	EF	= 0028H
110	=	F7	= 0030H
111	=	FF	= 0038H

The contents of the Program Counter is pushed onto the stack. Then the high byte of the PC is loaded with 00H and the low byte is loaded with the RST value ttH ANDed with 38H. execution continues at this address.

INPUT & OUTPUT INSTRUCTIONS

IN A,(n)

1	1	0	1	1	0	1	1

= DB + Port number

No flags affected

The value "n" selects the I/O device
at one of the 256 possible Ports.
The A register (accumulator) returns the value.

OUT (n),A

1	1	0	1	0	0	1	1

= D3 + Port number

No flags affected

The value "n" selects the I/O device
at one of the 256 possible Ports.
The value in the A register is sent to the Port.

IN r,(C)

ED Byte +

0	1	r	r	r	0	0	0

		IN	OUT
rrr -	B - 000	ED-40	ED-41
	C - 001	ED-48	ED-49
	D - 010	ED-50	ED-51
	E - 011	ED-58	ED-59
	H - 100	ED-60	ED-61
	L - 101	ED-68	ED-69
	A - 111	ED-78	ED-79

--- Flags ---
S - set if input data negative
Z - set if input data zero
H - reset
P/V - set if parity even
N - reset
C - not affected

The Port whose value is in the C register
sends a byte which is placed in the register r.

OUT (C),r

ED Byte +

0	1	r	r	r	0	0	1

Flags not affected

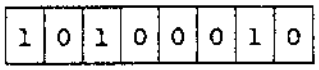
Code as above

The Port whose value is in the C register
has the value of the register r sent.

INPUT & OUTPUT INSTRUCTIONS

INI

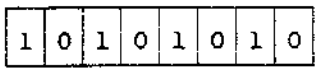
ED Byte +



= A2

IND

ED Byte +

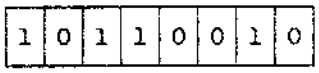


= AA

--- Flags ---
S, H & P/V are unknown
Z - set if B decr'ts to 0
N - is set
C - not affected

INIR

ED Byte +



= B2

INDR

ED Byte +

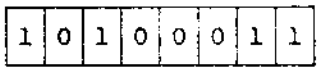
= BA

--- Flags ---
S, H & P/V are unknown
Z & N are set
C - not affected

The contents of the C register selects the Port number.
A byte from the Port is then placed in memory at the location pointed to by the HL register pair.
Then HL is incremented (INI,INIR) or decremented (IND,INDR).
The Register B is decremented and the program continues (INI,IND)
Or the instruction is repeated until B is zero (INIR,INDR).

OUTI

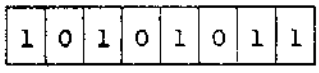
ED Byte +



= A3

OUTD

ED Byte +

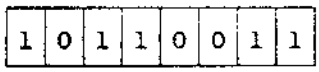


= AB

--- Flags ---
S, H & P/V are unknown
Z - set if B decr'ts to 0
N - is set
C - not affected

OTIR

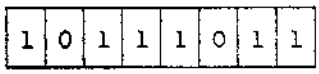
ED Byte +



= B3

OTDR

ED Byte +



= BB

--- Flags ---
S, H & P/V are unknown
Z & N are set
C - not affected

The contents of the C register selects the Port number.
A byte from the memory location pointed to by the HL register pair is written to the Port.
Then HL is incremented (OUTI,OTIR) or decremented (OUTD,OTDR).
The Register B is decremented and the program continues (OUTI,OUTD)
Or the instruction is repeated until B is zero (OTIR,OTDR).

UNLISTED HX LX HY LY CODES

These Pages have the unlisted codes for manipulating separately the high and low bytes of the Index Registers IX & IY. The operation and flags are identical to other 8 bit Arithmetic and Logical operations.

The use of these instructions is up to the imagination of the programmer. However, with a little thought, some ideas are very helpful. In a situation using IX as an Index register to tables or screens of ASCII or Data there can be, for instance, tables of 256 (or multiple) byte length that have corresponding ASCII and Data at a 256 (or multiple) byte offset. By incrementing or decrementing HX the corresponding lot of information can be easily accessed.

There is probably no disassembler available that handles these codes. When using an assembler with them enter each byte with the DEFB statement and the CPU will execute them as the operations listed.

LD r,x DD or FD Byte + | 0 | 1 | r | r | r | 1 | 0 | x |

			HX	LX	HY	LY
		x =	0	1	0	1
	rrr = B = 000		DD-44	DD-45	FD-44	FD-45
	C = 001		DD-4C	DD-4D	FD-4C	FD-4D
	D = 010		DD-54	DD-55	FD-54	FD-55
	E = 011		DD-5C	DD-5D	FD-5C	FD-5D
	HX = 100		DD-64	DD-65	FD-64	FD-65
No flags	LX = 101		DD-6C	DD-6D	FD-6C	FD-6D
affected	A = 111		DD-7C	DD-7D	FD-7C	FD-7D

LD x,r DD or FD Byte + | 0 | 1 | 1 | 0 | x | r | r | r |

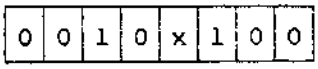
			HX	LX	HY	LY
		x =	0	1	0	1
	rrr = B = 000		DD-60	DD-68	FD-60	FD-68
	C = 001		DD-61	DD-69	FD-61	FD-69
	D = 010		DD-62	DD-6A	FD-62	FD-6A
	E = 011		DD-63	DD-6B	FD-63	FD-6B
	HX = 100		DD-64	DD-6C	FD-64	FD-6C
No flags	LX = 101		DD-65	DD-6D	FD-65	FD-6D
affected	A = 111		DD-67	DD-6F	FD-67	FD-6F

LD x,v DD or FD Byte + | 0 | 0 | 1 | 0 | x | 1 | 1 | 0 | + byte

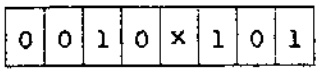
HX	DD-26-v	HY	FD-26-v
LX	DD-2E-v	LY	FD-2E-v

UNLISTED HX, LX, HY, LY CODES

INC Hx — DD or FD Byte +
HX = DD-24 HY = FD-24
LX = DD-2C LY = FD-2C



DEC Hx — DD or FD Byte + 25 =
HX = DD-25 HY = FD-25
LX = DD-2D LY = FD-2D



-- Flags INC/DEC --
S - set if result negative
Z - set if result zero
H - set if carry from Bit 3 (INC), no borrow from Bit 4 (DEC)
P/V - set if byte was 7FH before (INC), was 80H before (DEC)
N - reset (INC), set (DEC)
C - not affected

ADD	x	DD or FD +	1 0 0 0 0 1 0 x		
ADC	x	DD or FD +		1 0 0 0 1 1 0 x	
SUB	x	DD or FD +	1 0 0 1 0 1 0 x		
SBC	x	DD or FD +		1 0 0 1 1 1 0 x	
AND	x	DD or FD +	1 0 1 0 0 1 0 x		
XOR	x	DD or FD +		1 0 1 0 1 1 0 x	
OR	x	DD or FD +	1 0 1 1 0 1 0 x		
CP	x	DD or FD +		1 0 1 1 1 1 0 x	

	ADD	ADC	SUB	SBC	AND	XOR	OR	CP
HX	DD-84	DD-8C	DD-94	DD-9C	DD-A4	DD-AC	DD-B4	DD-BC
LX	DD-85	DD-8D	DD-95	DD-9D	DD-A5	DD-AD	DD-B5	DD-BD
HY	FD-84	FD-8C	FD-94	FD-9C	FD-A4	FD-AC	FD-B4	FD-BC
LY	FD-85	FD-8D	FD-95	FD-9D	FD-A5	FD-AD	FD-B5	FD-BD

FLAGS --- ADD/ADC/SUB/SBC --- Common --- AND/XOR/OR/CP ---
S = set if result negative
Z = set if result zero
H = set if carry from Bit 3 - ADD/ADC
borrow from Bit 4 - SUB/SBC
P/V=set if overflow
N = set ADD/ADC : reset SUB/SBC
C = set if carry ADD/ADC: borrow SUB/SBC
H = AND/XOR/OR - set
CP - set if overflow
P/V= set if Parity even
CP - set if Overflow
N = reset:CP -set
C = reset:CP -set if borrow

A P P E N D I X

2 EXTRA "&" FUNCTIONS

7.23. & Extra uses for the "&" Function with NEWDOS/80 Models I & III.

(1) &ddddd - returns the Hex equivalent of Decimal Input (hhhhH)
(2) &%ddddd - returns the Integer signed decimal of ddddd.

e.g. &37654 returns 9316H and &%37654 returns -27882.

While the "&" function (No.1) can only be used as a String, the "&%" function (No.2) can be used inside a Program for any value (0 to 65535) for PEEKs, POKEs, or any other uses requiring an Integer for operations. As well as expressing -32768 as 8000H, both functions convert 32768 to 8000H as well, so as to give a Hex conversion of addresses between 32767 and 65536 (No.1), and an Integer value (No.2).

Naturally, in Function (2) this conversion stops a FOR-NEXT loop from crossing between 32767 and 32768 as the 32768 is now -32768.
i.e. (FOR I = &%32767 TO &%32768 : POKEI,n : NEXT) will not work
but (FOR I = 32767 TO 32768 : POKE&%I,n : NEXT) will work.

The Zaps to be applied :-

From DOS load SUPERZAP
DFS - SYS20/SYS - Sector 3 - MOD 00
change 28 01 2B to CZ AE 56.
<ENTER> - Y - <ENTER> - K - 4 - MOD C2
change all 00's from C2 to FB to

```
C0 -  18 EC  FE 25  28 14  2B CD  C6 56  E5 21  30 41  E5 CD
D0 -  D2 44  36 48  3E 05  01 C3  81 28  CD 38  23 E5  01 06
E0 -  2B C5  E7 E8  3A 24  41 31  80 90  11 00  00 88  C0 3A
F0 -  23 41  B9 D0  04 00  16 07  00 00  00 00  32 02  00 52
```

This is how it should look when you are finished. N.B.- The underlined Bytes at D0 are the address of a routine in the Model III for converting the Hex contents of the DE Register pair to ASCII. The corresponding address in the Model I is 4063H.
Thus in the Model I zap at D0 above 63 40 instead of D2 44.

2 EXTRA "&" FUNCTIONS

The most use of the second function will be in the POKE, PEEK & USR usage where the value might cross the invisible barrier between 32767 and 32768. The continual challenge of quickly figuring the Hexadecimal equivalent of 41550 is also made a thing of the past.

Below is a commented disassembly of the code used :-

```
  56AC  18 EC       JR    569AH      ; End of existing code

1 56AE  FE 25       CP    25H        ; Test for " % "
  56B0  28 14       JR    Z,56C6H    ; If so go to Subr & Ret to caller
  56B2  2B          DEC   HL         ; Go back for Incr in 2338H
1 56B3  CD C6 56    CALL  56C6H      ; Call Subroutine
  56B6  E5          PUSH  HL         ; Store Addr of start of Entry
  56B7  21 30 41    LD    HL,4130H   ; HL points to Print Buffer
  56BA  E5          PUSH  HL         ; Store for DE later
  56BB  CD 02 44    CALL  4402H      ; Convert DE to $, store in (HL)
  56BE  36 48       LD    (HL),48H   ; Add "H" to end of string
  56C0  3E 05       LD    A,05H      ; Load A with length of String
  56C2  D1          POP   DE         ; DE = Address of Print Buffer
  56C3  C3 81 28    JP    2881H      ; Move to String area & Return

;                 Conversion Subroutine

1 56C6  CD 38 23    CALL  2338H      ; Evaluate entry - Result in ACC
  56C9  E5          PUSH  HL         ; Store Addr of start of Entry
  56CA  01 06 2B    LD    BC,2B06H   ; BC has Routine Addr.
  56CD  C5          PUSH  BC         ;  onto Stack for Return
  56CE  E7          RST   20H        ; Test Number Type Flag
  56CF  E8          RET   PE         ; Return if not Single Precision
  56D0  3A 24 41    LD    A,(4124H)  ; Exponent of SP No ) If value
  56D3  01 80 90    LD    BC,9080H   ; BCDE gets SP       ) is between
  56D6  11 00 00    LD    DE,0000H   ; value of -32768    ) 32767 &
  56D9  B8          CP    B          ; Is it 90H ?       ) 65536
  56DA  C0          RET   NZ         ; If not RETurn     ) convert to
  56DB  3A 23 41    LD    A,(4123H)  ; MSB of SP No.     ) Negative
  56DE  B9          CP    C          ; If negative       ) by adding
  56DF  D0          RET   NC         ;   Return          ) -65536.
  56E0  04          INC   B          ; BCDE = -65536     ) Result
  56E1  C3 16 07    JP    0716H      ; Add to ACC & Ret  ) = Integer
```

The 'Result=Integer' of the bottom line means that the result of the addition is still Single Precision (32768 to 65535) but is capable of expression as an Integer (-32768 to -1).

SYSTEM OPTION PRINT ENHANCEMENT

This enhancement lists up to 12 bytes of ASCII explanation with each option of the SYSTEM command of NEWDOS/80 Version 2. The SYS17/SYS module handles the printing of the SYSTEM options.

This Zap clashes with an Apparat Zap (#057) which corrects an error occurring when the number of lumps exceeds 96, overwriting the lock-out table. By increasing the value of Granules per Lump (GPL PDRIVE parameter) this is obviated (See the table on Page 63). Nor is this modification available with NEWDOS/80 Version 2.5 (Hard Drive) as this area is used for Hard Drive routines.

A Superzap read of the sectors is shown below. The first sector, File Relative Sector 0, gives the chance to initialise the new code and the Call to execute it. The second sector has a change of one byte from a comma (the previous separator) to a dash (the new separator). The fourth sector has the extra code involved as well as the start of the ASCII. The last sector is all ASCII. Note carefully that the bytes at 08H to 0BH in the last sector are the load instructions and must not be changed. They tell the DOS to load (01), how many bytes (F2H) and the address into which to load (50F8H). The C0H bytes are filler and print nothing.

Disassembly for the Model I :-

```
501CH  D9              EXX             ; Alternate Registers
501DH  21 74 50        LD   HL,5044H   ; Start of ASCII
5020H  0E 00           LD   C,0        ; First gap between Options
5022H  D9              EXX             ; Store for Print
5023H  11 00 00        LD   DE,0000H   ; Replace call line
5026H  C9              RET             ; Go back

;      Print Subroutine
5027H  D9              EXX             ; Stored Registers
5028H  3A 20 40        LD   A,(4020H)  ; Get LSB of Cursor Position
502BH  E6 FC           AND  0FCH       ; Lose bottom 2 bits
502DH  81              ADD  A,C        ; Add 0 (first time) or 4
502EH  0E 04           LD   C,4        ; Set for gap after first
5030H  32 20 40        LD   (4020H),A  ; Set cursor to new position
5033H  06 0C     PRINT LD   B,12       ; Number of characters
5035H  7E        LOOP  LD   A,(HL)     ; Get character
5036H  CD 6B 4E        CALL 4E6BH      ; Print it
5039H  23              INC  HL         ; To next character
503AH  10 F9           DJNZ LOOP       ; Do it 12 times
503CH  D9              EXX             ; Restore regs
503DH  78              LD   A,B        ; Get separator
503EH  CD 6B 4E        CALL 4E6BH      ; Print it
5041H  C9              RET             ; Finished, go back
```

SYSTEM PRINT ENHANCEMENT : Model I

The following changes are for the Model I :-

From SUPERZAP : DFS : SYS17/SYS : 0 : MOD C7
Change C7 from 11 00 00 06 00 to CD 1C 50 06 2D
Change CF from CD 6B 4E to CD 27 50

Then move on to the next Sector : ; : MOD 18
Change 18 from 2C to 2D

Then move on to File Relative Sector 3 : ; ; : MOD 2C
Change 2C from all 00's as per Sector Printout

```
DRV 00   0607 1ABE 1323 0100 FA4F 20CA 10F8 C942 .....".#..O ....B
0   10   4F4F 5444 4952 2020 2020 2053 5953 8360 OOTDIR.....SYS.`
0H  20   0000 4200 00FF 0000 0000 FFFF 0921 4450 ..B..........!DP
     30  0E00 0911 0000 C9D9 3A20 40E6 FC81 0E04 ........:.@.....
DRS 40   3220 4006 0C7E CD6B 4E23 10F9 0978 CD6B 2.@..~.kN#...x.k
108 50   4EC9 0000 456E 2050 6173 7377 6F72 6473 N...En.Passwords
6CH 60   5255 4E2D 4F6E 6C79 4D6F 6465 454E 2044 RUN-OnlyModeEN.D
     70  6562 6F75 6E63 6520 0D45 6E61 626C 6522 ebounce..Enable"
     80  4A4B 4C22 456E 6162 6C65 2022 3132 3322 JKL"Enable."123"
     90  456E 6162 6C65 2022 4446 4722 0D45 6E42 Enable."DFG".EnB
     A0  5245 414B 2048 6579 4C2F 4320 4D6F 6420 REAK.KeyL/C.Mod.
     B0  496E 7374 4862 6420 496E 7427 6365 7074 InstKbd.Int'cept
FRS C0   0D4E 6F44 7269 7665 73C0 C0C0 5265 6164 .NoDrives...Read
3   D0   732F 4572 72C0 C0C0 4449 5220 4472 6976 s/Err...DIR.Driv
3H  E0   65C0 C0C0 0D4C 6F77 6573 7444 7269 7665 e....LowestDrive
     F0  4849 4D45 4DC0 C0C0 C0C0 C0C0 456E 436C HIMEM.......EnCl


DRV 00   6561 7248 6579 C0C0 01F2 F850 0D43 6F70 earKey.....P.Cop
0   10   794E 6F20 5073 7764 4261 7369 6349 6E70 yNo.PswdBasicInp
0H  20   7455 2F43 4348 4149 4E20 496E 7075 743F tU/CCHAIN.Input?
     30  0D4B 6272 6420 5265 7065 6174 3235 6D73 .Kbrd.Repeat25ms
DRS 40   4465 6C61 79C0 C0C0 5772 6974 6573 2F45 Delay...Writes/E
109 50   7272 C0C0 0D50 7472 4869 6768 43C0 C0C0 rr...PtrHighC...
6DH 60   4461 7465 5469 6D65 436F 6C64 4461 7465 DateTimeColdDate
     70  5469 6D52 6573 6574 0D52 4F55 5445 2044 TimReset.ROUTE.D
     80  4F2C 4E4C 4348 4149 4E20 5061 7573 653F O,NLCHAIN.Pause?
     90  4F76 6572 7269 6465 4155 544F 0D52 3D43 OverrideAUTO.R=C
     A0  4D44 5265 7065 6174 466F 7263 654C 4344 MDRepeatForceLCD
     B0  5652 2C4E 466F 7263 6520 4C43 2C4E 2020 VR,NForce.LC,N..
FRS C0   0D42 6C69 6E6B 4375 7273 6F72 4375 7273 .BlinkCursorCurs
4   D0   72C0 C056 616C C0C0 5469 6D4C 6F6F 704D r..Val..TimLoopM
4H  E0   756C 74C0 0D45 6E61 622E 5752 4449 5250 ult..Enab.WRDIRP
     F0  4449 5220 4D4F 4420 332F 3120 0202 004D DIR.MOD.3/1....M
```

SYSTEM OPTION PRINT ENHANCEMENT

The only Option which is not self-explanatory is the one to choose the Lowest Drive Number for new file creation. The ASCII for this option (AO) shows only "Lowest Dr".

The Model I has 5 SYSTEM Options not in the Model III, (AC, AI, AS, BF & BN) whereas the Model III has only one extra, (BB). The user has eliminated the old BM option which was rendered unnecessary by ZAP 44.

Disassembly for the Model III :-

```
5020H  D9              EXX             ; Alternate Registers
5021H  21 74 50        LD   HL,5074H   ; Start of ASCII
5024H  AF              XOR  A          ; Zero A
5025H  4F              LD   C,A        ; First gap between Options
5026H  D9              EXX             ; Store for Print
5027H  32 24 40        LD   (4024H),A  ; Space Comp. Code toggle
502AH  11 00 00        LD   DE,0000H   ; Replace call line
502DH  C9              RET             ; Go back

;      Print Subroutine

502EH  D9              EXX             ; Stored Registers
502FH  3A 20 40        LD   A,(4020H)  ; Get LSB of Cursor Position
5032H  47              LD   B,A        ; Store in B
5033H  E6 3C           AND  3CH        ; See if end of line
5035H  FE 34           CP   34H        ; Is it end ?
5037H  38 07           JR   C,SPACE    ; If less go over
5039H  3E 0D           LD   A,0DH      ; Carr. Return & Line Feed
503BH  CD 6B 4E        CALL 4E6BH      ; Call Print routine
503EH  18 09           JR   PRINT      ;  & jump across
5040H  78        SPACE LD   A,B        ; Restore value
5041H  E6 FC           AND  0FCH       ; Lose bottom 2 bits
5043H  81              ADD  A,C        ; Add 0 (first time) or 4
5044H  0E 04           LD   C,4        ; Set for gap after first
5046H  32 20 40        LD   (4020H),A  ; Set cursor to new position
5049H  06 0C     PRINT LD   B,12       ; Number of characters
504BH  7E        LOOP  LD   A,(HL)     ; Get character
504CH  CD 6B 4E        CALL 4E6BH      ; Print it
504FH  23              INC  HL         ; To next character
5050H  10 F9           DJNZ LOOP       ; Do it 12 times
5052H  D9              EXX             ; Restore regs
5053H  78              LD   A,B        ; Get separator
5054H  CD 6B 4E        CALL 4E6BH      ; Print it
5057H  C9              RET             ; Finished, go back
```

SYSTEM PRINT ENHANCEMENT : Model III

The following changes are for Model III :-

From SUPERZAP : DFS : SYS17/SYS : 0 : MOD C7
Change C7 from 11 00 00 06 00 to CD 2C 50 06 2D
Change D0 from CD 6B 4E to CD 3C 50

Then move on to the next Sector : ; : MOD 18
Change 18 from 2C to 2D

Then move on to File Relative Sector 3 : ; ; : MOD 2C
Change 2C from all 00's as per Sector Printout

```
DRV 00   1643 8920 0011 0100 FA4F 0543 0604 CD06 .C.......O.C....
0   10   5021 1750 0607 1ABE 1323 20CA 10F8 C942 P!.P.....#.....B
0H  20   4F4F 5444 4952 2020 2020 2053 5953 8360 OOTDIR.....SYS.`
     30  0000 4300 00FF 0000 0000 FFFF D921 7450 ..C..........!tP
DRS 40   AF4F D932 2440 1100 00C9 D93A 2040 47E6 .O.2$@.....: @G.
108 50   3CFE 3438 073E 0DCD 6B4E 1809 78E6 FC81 <.48.>..kN..x...
6CH 60   0E04 3220 4006 0C7E CD6B 4E23 10F9 D978 ..2.@..~.kN#...x
     70  CD6B 4EC9 0000 0000 0000 0000 0000 0000 .kN.............
     80  0000 0000 456E 2050 6173 7377 6F72 6473 ....En.Passwords
     90  5255 4E2D 4F6E 6C79 4D6F 6465 456E 6162 RUN-OnlyModeEnab
     A0  6C65 2022 4A4B 4C22 456E 6162 6C65 2022 le."JKL"Enable."
     B0  3132 3322 456E 6162 6C65 2022 4446 4722 123"Enable."DFG"
FRS C0   456E 2042 5245 414B 206B 6579 4B62 6449 En.BREAK.keyKbdI
3   D0   6E74 6572 6365 7074 4E6F 2044 7269 7665 nterceptNo.Drive
3H  E0   7320 20C0 5265 6164 732F 4572 726F 7220 s...Reads/Error.
     F0  4449 5220 4472 6976 654E 6FC0 4C6F 7765 DIR.DriveNo.Lowe

DRV 00   7374 2044 7269 76C0 01F2 F850 C0C0 C0C0 st.Driv....P....
0   10   C0C0 4849 4D45 4DC0 456E 2043 6C65 6172 ..HIMEM.En.Clear
0H  20   204B 6579 436F 7079 204E 6F50 7377 7264 .KeyCopy.NoPswrd
     30  4368 6169 6E20 496E 7075 743F 4B79 6272 Chain.Input?Kybr
DRS 40   6420 5265 7065 6174 3235 6D73 446C 6179 d.Repeat25msDlay
109 50   73C0 C0C0 5772 6974 6573 2F45 7272 6F72 s...Writes/Error
6DH 60   5072 7472 4869 6768 7374 4368 4461 7465 PrtrHighstChDate
     70  5469 6D65 436F 6C64 4461 7465 5469 6D52 TimeColdDateTimR
     80  6573 6574 524F 5554 4520 444F 2C4E 4C20 esetROUTE.DO,NL.
     90  436C 6F63 6B35 302F 3630 6379 4368 6169 Clock50/60cyChai
     A0  6E20 5061 7573 653F 4F76 6572 7269 6465 n.Pause?Override
     B0  4155 544F 5230 434D 4420 5265 7065 6174 AUTOR=CMD.Repeat
FRS C0   466F 7263 6520 4C2F 432C 5920 426C 696E Force.L/C,Y.Blin
4   D0   6B20 4375 7273 6F72 4375 7273 7256 616C k.CursorCursrVal
4H  E0   C0C0 C0C0 5469 6D65 4C6F 6F70 4075 6C74 ....TimeLoopMult
     F0  456E 6162 6C65 5752 4449 5250 0202 004D EnableWRDIRP...M
```

ABOUT NEWDOS/80 PDRIVES

First some things to presume :-

1. NEWDOS/80 has a standard of 5 sectors per Granule (SPG)

2. 1 Granule is the smallest amount of space that can be allocated on a Disk Drive i.e. a program only 10 bytes long takes up 5 sectors or 1280 bytes of disk storage.

3. A lump is a measure of disk storage. Each lump is represented by 1 byte in the GAT sector (1st sector of Directory). One bit of this byte is used for each gran taken. The total number of bits used in each byte is the same as the GPL value (i.e. the number of Granules per Lump). The bits are allocated starting from bit zero.

4. In the lockout table each lump is represented by 1 byte just as in the GAT sector. Similarly, one bit of this byte is used for each gran not locked out. This lockout table is tested as to whether it is in use or not by reading byte 60H of the GAT sector. If not in use this byte should be FFH. As this is the lockout byte for the Boot sector it is not surprising that it should not be locked out. The DOS tests this to see if the lockout table has been over-run by the number of bytes used by the allocation table above it i.e. the number of lumps on the disk. In NEWDOS/80, whenever the number of lumps is greater than 96, the lockout table is used by the Granule Allocation Table. This happens, for example, when the number of sectors used in the drive amounts to more than 960 and GPL=2.

- : - : - : - : - : - : - : - : - : -

To begin, the simple :-

TI=A : TD=A : TC=40 : SPT=10 : TSR=3 : GPL=2 : DDSL=17 : DDGA=2

This means that we have :-

TI=A a standard Tandy Interface

TD=A Single Density, Single Sided

(only covers 5 inch)

Type	Dens	Side	Sect/Tr
A	S/D	S/S	10
C	S/D	D/S	20
E	D/D	S/S	18
G	D/D	D/S	36

TC = 40 Track Count - 40 Track Drive (48 Tracks per Inch)
SPT = 10 Sectors per Track (Standard IBM format 256byte Sector)
TSR = 3 Stepping rate between Tracks (0=fastest : 3=slowest)
GPL = 2 Number of min. units of space (granules) per lump
DDSL = 17 Disk Directory starting Lump
DDGA = 2 Allocation of space units to Directory

ABOUT NEWDOS/80 PDRIVES

Now to get into it :-

If you have a 40 Track, Single Sided, Single Density (GPL=2) Drive then 1 Lump = 1 Track, so all is very simple. To determine the status of any track (lump) just count the required number of bytes into the GAT sector and look at which bits are set. There is another table which looks like almost a copy of the Granule Allocation Table below it starting at byte 60H. This is the lockout table and is used sometimes to tell the DOS not to use the space represented here as some trouble had occurred during formatting.

You might have noticed that in this situation the GAT sector shows a completely unused lump as FC. This means that all that is used in each byte are bits 0 & 1 (GPL=2). Now each byte refers to a Lump and in the simplest case this is also 1 Track (as above). So you can see what track a byte refers to by counting each byte.

Now pull your socks up :-

In double sided drives the SPT changes to twice as many because tracks span sides. Then the track-lump nexus is broken.

TI=A : TD=C : TC=40 : SPT=20 : TSR=3 : GPL=2 : DDSL=17 : DDGA=2

The SPT has changed to 20, TC remains at 40, GPL at 2, etc.

Now the total number of sectors = TC * SPT = 40 * 20 = 800

And the Directory is still at SPG * GPL * DDSL = 5 * 2 * 17 = 170. You have changed the relative position of the Directory on the disk, haven't you. Also the number of bytes used by the system in the GAT sector has climbed to 80 (50H) instead of the old 40 (28H). Well why is this so ... because each lump is still 2 grans and considering the sectors per gran does not change, there are twice the number of sectors.

So we take the big step :-

We now make GPL=4. Now the Directory is back to the same relative position (Sector 340 in 800 Sectors - the same as 170 in 400) and only 40 bytes are used in the GAT sector and the Track-Lump nexus is restored. However in the GAT an unused lump is now F0, quite a step from FC. Bananas. FC has bits 0 & 1 both 0 catering for 2 granules per lump while F0 has bits 0,1,2 & 3 all 0 thus catering for 4 granules per lump.

A conclusion can be drawn here as to how many grans to a lump are possible. You guessed it, 8. For a similar reason the number of sectors per granule is limited to 8. Thus we have a maximum capability of 8 * 8 * (202bytes possible in the GAT sector) * 256 bytes per sector. This comes to 3,309,568 bytes (over 3 Megabytes) in 1 PDRIVE slot or 26,476,544 bytes (over 25 Megabytes) for all 8 slots capable. Well, this is how they get on in hard drives with all those thousands of sectors.

ABOUT NEWDOS/80 PDRIVES

Now pull your proverbial socks up further :-

How about double density ? (expletive deleted) 18 sectors per track. How the hell ?? 5 doesn't go into 18. So we have permanently broken the Track-Lump nexus so we have to learn what a lump means and what APPARAT says about Directory start sectors.

Now you all know 17. In 40 Tracks this is the standard DDSL and in single density with GPL=2 also the Directory starting Track, but in 80 Tracks APPARAT recommends 35 as the DDSL. In NEWDOS/80 we place the Directory at the start of a lump not a track. Obviously in double density the lump must span tracks as well as sides (5 does not go into 18 again).

So in the standard Model III, the Type of Drive is TD=E ; if DDSL = 17 then the Directory starting sector is still 170 (if GPL=2) in a 720 track drive ; not even a quarter way through the disk. Although, say, with DDSL=24 the Directory would start about a third way through, up to now GPL=2 and DDSL=17 are standard in all Model III disks emanating from Apparat.

However, with 40 track double sided there is no established precedent. The most obvious thing to do is to balance the number of sectors with the GPL or make the DDSL suit. There is a precedent for 80 track double sided in that Apparat suggests DDSL=35 and GPL=8 which does balance the amounts quite well.

What a lot of pedantic nonsense you say !!

Well, as an extreme case, on an 80 track D/S D/D disk there are 80 * 36 = 2880 sectors and if GPL=2 and DDSL=17 the Directory would start at Drive sector 170. How bloody ridiculous. There is actually no disadvantage in this situation except that you will hear the poor 80 tracker rattle over 40 to 60 tracks nearly every time it accesses a program (the program must be found in the DIR) and the physical wear this might precipitate. With judicious use of the GPL and DDSL values the normal head movement can be cut to a minimum.

To keep the lockout table in use, and save the complications caused by it not being so, it seems wise to balance the GPL against the number of granules on the disk as well as keeping the Directory in a sensible place on the disk. A zap (23) to the COPY function explains that DIRCHECK sees an error where none exists when the lockout table is used as part of the Granule allocation table. Zap 57 of NEWDOS/80 corrects an error occurring when the number of lumps exceed 96.

ABOUT NEWDOS/80 PDRIVES

Now some theory :-

In a drive the most common access is to the SYS files. These are all placed on a system disk in very particular places. The RST 28H Restart which loads these SYS files does not access the **HIT Sector** at all. The contents of the A register dictates what DEC (location in the Directory) will be loaded. BOOT/SYS and the main resident DOS module SYS0/SYS reside at the start of a disk. All other SYS files are clustered around the Directory from 65 sectors before to 70 sectors after. This is the most efficient place for them no matter where the Directory is or how long it is.

These SYS files are used to perform functions and to manipulate Basic programs etc. In the simplest case where SPT=10 the SYStem files take 145 sectors which is 15 tracks. If SPT=36 then the 145 sectors use just over 4 tracks. The head, in a typical case, moves to the SYS program then to the Directory, then to the program.

Considering the fact that the start of the program is the part accessed (you will only at a maximum access the end of the disk minus the length of the program) as well as the likelyhood of some free space at the end of the disk, then just over a third of the way through the disk sounds a reasonably efficient place for the start of a Directory.

In TRSDOS Model III the number of sectors per gran is 3 and the number of grans per track 6. This keeps the track - lump nexus and the number of bytes used in the GAT sector to 40H. However there is no accomodation for double sided or 80 track disks. In LDOS it is almost the same in track-GAT byte usage and in DOSPLUS the same plan is used. Both these DOSes now cater to side and track variation.

In both DOSPLUS and LDOS a cylinder is used as a measure of space and it involves all the sectors on a given track, both sides. Thus on a hard drive, considered vertically, the number of surfaces that are used (i.e. the number of platters or disks * the number of sides used on each) on each track are all taken together as being a cylinder. This is the amount of space represented by a byte in the GAT sector. In hard drives LDOS partitions by heads so that it has a nexus which might not occur in NEWDOS/80.

We have LDOS and TRSDOS users saying they cannot understand NEWDOS usage of the word lump but as NEWDOS was first marketed when the state of the art was very rudimentary it was then a very advanced DOS and for that matter still is in some of its functions and capabilities.

ABOUT COPY

There are quite a few switches available in NEWDOS/80 in the COPY function by which very diverse configurations can be accomodated. So many that it can become quite complicated to use. This is simply a list the proceedures involved. This information is intended as a reminder not a full explanation.

FORMAT also uses some of these parameters ;namely =tc, N, Y, NDMW, DDND, ODN, KDN, DDSL, DDGA, DPDN, PFST, PFTC, and RWF, the last three being used only by format.

The DOS uses the default PDRIVE specifications if not notified otherwise. To inspect these, issue a PDRIVE 0 command . To ascertain what each of the examples is doing refer to the explanation of each switch.

There are 6 formats in the COPY command

Format 1 - Single file copy using 2 drives with a SYSTEM disk in drive 0.

Examples

```
COPY PROG:0 :1
COPY PROG:0 NEW:0-
COPY PROG:1 NEW:2
COPY PROG:1 :2 DPDN=9
```

Format 2 - Single file copy using 1 or 2 drives but neither disk necessarily being the SYSTEM disk and all disk mounts are requested when needed.

Examples

```
COPY $PROG:1 :0
COPY $PROG:0 :0
COPY $PROG:0 :3 SPDN=9
```

Format 3 - Single file copy using only 1 drive and both disks being SYSTEM disks.

Examples

```
COPY 0 PROG PROG
COPY 1 OLD/DAT NEW/DAT
COPY 2 PROG PROG DPDN=5
```

Format 4 - Single file copy using only 1 drive with neither of the disks necessarily being the SYSTEM disk and all disk mounts are requested when needed.

Examples

```
COPY 0 $PROG PROG
COPY 1 $DATA/DAT OLD/DAT DPDN=9
COPY 0 $PROG NEWVER SPDN=5
```

Format 5 - Full disk copy with both disks having the same PDRIVE specs (except track count) and the source not necessarily being the SYSTEM disk. All disk mounts are requested when needed unless NDMW is specified.

```
Examples        COPY 0 1,,FMT
                COPY 0 0 16/03/84 FMT
                COPY,0,0,16/03/84,FMT,DDND,SN=NAME1
                COPY 0 2,,NFMT,USD,KDN,NDMW
                COPY 2 3,,NFMT,BDU
                COPY 1=39 2=40,,FMT,NDMW,SPDN=5
```

Format 6 - Full disk copy with no PDRIVE compatibility necessary and source not necessarily being the SYSTEM disk. The SYSTEM files on the source disk are not transferred unless FMT is specified. A verify is done if FMT is specified. All disk mounts are requested when needed except if NDMW is specified.

```
Examples        COPY 0 1,,FMT,CBF
                COPY 0 1,,NFMT,CBF,USR
                COPY 0 1,,CBF,NFMT,USR,CFWO,NDMW
                COPY 0 1,,CBF,NFMT,USR,UPD,DFO,NDMW
                COPY 0 1,,CBF,FMT,NDMW,DPDN=9
```

The default specifications for any drive used in a COPY are the PDRIVE specs as shown in the PDRIVE 0 command. Beware of trying to copy to a drive not set as being active by the SYSTEM parameter AL.

A full disk copy cannot be executed from Basic as the SYS6/SYS file is 7 Grans because of space limitations. It would overwrite the area used by Basic. If the files in a format 6 COPY will not fit on the destination disk then the user is notified which file(s) will not fit. The files that are partially loaded can be seen in a 'DIR A' as having no length (EOF=0) but the GAT sector is still allocating their space. So make sure you kill those files to free up the space.

Pressing the up arrow will terminate the COPY function but not during disk access as the interrupts are disabled. Just keep the key pressed for a while.

ABOUT COPY

A list of the switches and their meanings follow :-

=tc - Track count.
Used immediately after either Drive number to tell the DOS to use 'tc' number of tracks on the disk during the copy.

BDU - Bypass Destination Directory Update.
This is used in format 5 copy. The 5 COPY normally transfers the sectors to be copied and then updates the data in the Boot, PDRIVE & Directory specifications. BDU inhibits the update of the data. This switch can be used to transfer from disks with a bad, a non standard or no Directory or copy a whole disk with no alterations.

CBF - Copy By File.
This is a format 6 copy. It is mostly used to transfer a whole disk contents between disks configured differently. This copy transfers files separately, not just sectors and so is useful to rearrange the files on a disk. In an 80 track disk that is not full, a CBF can take a lot less time than a full format 5 copy because it does not copy all the sectors.

CFWO - Check Files With Operator.
The files on the source disk are displayed and the user is asked if each is to be copied. The Y/N/R/Q is useful if you make a mistake. R to restart the process or Q to copy only the files previously requested.

DDGA - Disk Directory Granule Allocation. (as in PDRIVE specs)
The DOS is to use the DDGA spec for the destination drive for the duration of the copy.

DDSL - Disk Directory Starting Lump. (as in PDRIVE)
The DOS is to use this DDSL specification for the destination disk for the duration of the copy.

DDND - Display Destination old Name and Date.
The destination name and date are read and printed and the operator is asked whether or not to proceed.

ABOUT COPY

DFO - Destination Files Only.
Only those files already on the destination disk will be copied.

DPDN= - Destination PDRIVE Number.
For the duration of the copy the destination drive is assumed to have the PDRIVE specs of the requested entry in the PDRIVE table.

FMT - Format.
The destination disk is to be formatted.

ILF= - Inclusion List File.
The DOS is to copy only those files which are listed in a file specially created to be read by the DOS during the copy. Such a file can be created with a word processor or by NEWDOS's CHAINBLD/BAS.

KDD - Keep Destination Date.
The destination disk is to retain its old date.

KDN - Keep Destination Name.
The destination disk is to retain its old name.

N - No data on destination disk.
The copy will be terminated if the destination disk contains recognizable data. This is useful if it is not known what is on the disk. This will not protect the data if the PDRIVE specs for the destination drive does not match the one for that disk. The data would not be recognizable under a different PDRIVE.

NDMW - No Disk Mount Waits.
The disk to be copied is in the source drive and the destination drive contains the destination disk. No mount prompts are given and the copy proceeds uninterrupted.

NFMT - No Format.
The destination disk is not to be formatted. To copy across System files a format must be done.

ODN - Old Destination Name.
The destination disk is to retain its old name.

ABOUT COPY

ODPW - Old Disk Password.
The destination disk is to retain its old password.

PFST= - Partial Format Starting Track. --FORMAT ONLY--
To allow the formatting of a range of tracks rather than the entire disk, set this parameter to the desired starting track number. In the absence of a PFTC (below) parameter only one track is formatted.

PFTC= - Partial Format Track Count. --FORMAT ONLY--
Used in conjunction with PFST (above) if necessary, to specify how many tracks to format. Must not be used without PFST.

RWF - Raw Format. --FORMAT ONLY--
Used to obliterate any data on the disk. Each track is formatted once only whether successful or not (errors are ignored). The disk will not to have any system data initialised (BOOT or DIR).

SN= - Source disk Name
If the name of the source disk is not as specified then the operator is given the option of quitting or proceeding with the copy.

SPDN= - Source PDrive Number
Used to instruct the DOS to treat the source disk as having the characteristics of the specified PDRIVE number. Would mainly be used in a single drive copy situation where the source disk and destination disk have different PDRIVEs or when using an 80 track destination drive to copy in 40 track mode (i.e. TI=AL).

SPW= - Source PassWord
If passwords are enabled the source password must be specified to enable the copy to proceed. This is so unless SYSTEM option AR=N in which case the copy is permitted. Option AR might be set to N in circumstances where an operator must be able to make backups of disks containing confidential files but still be denied access to their contents.

UBB - Use Big Buffer
An outdated command - relates only to NEWDOS80 V1.0 and earlier.

UPD - UPDated files only
Each file in a directory contains a flag which is set when the file is written to in any way. The PROT command is used to reset these flags. If the UPD parameter is used then only those files whose update flags are set will participate in the copy. This is useful because only files that have been changed are copied and no time is spent unnecessarily copying identical files. It is necessary to issue a PROT :d,RUF command to both disks after this type of copy to reset the update flags (d=drive number in each case). This is especially so if master and backup disks are alternated.

USD - Use Source Date
The destination disk will be given the date on the source disk.

USR - USeR files only
System and invisible files are omitted from the copy.

XLF= - eXclude Listed Files
Copy all files except those listed in a file specially created to be read by the DOS during the copy.

Y - Yes overwrite destination disk
The user no longer requires any data that may be on the destination disk.

- : - : - : - : - : - : - : -

VERY IMPORTANT FOR 80 TRACK DRIVE USERS :-

In the situation where an 80 track drive is used to save programs or data onto a 40 track disk, the data has problems being read by a 40 track drive. The width of the signal laid down by an 80 track head is about two thirds of the width of the signal laid down by a 40 track head. The 40 track drive thus reads all the data written by the 80 track head as well a width of the signal laid down originally by the 40 track head. This may not cause read errors if you are very lucky, but experience tells that it cannot be reliably read by a 40 track drive. The disk, never the less, is still readable by an 80 track drive.

ABBREVIATIONS

!	Single precision declaration symbol : NTF = 4
#	Double precision declaration symbol : NTF = 8
$	String declaration symbol : NTF = 3
%	Integer precision declaration symbol : NTF = 2
123	3 keys which when pressed simultaneously execute DEBUG
AACC	The auxiliary Accumulator
ACC	The Accumulator (the A register)
ASC	ATTRIB param allowing or not automatic Disk space deallocation
ASE	ATTRIB param allowing or not automatic Disk space allocation
Bit	Smallest part of memory. 8 bits to a byte
Buffer	An area of memory set aside for I/O, typically sector read
Bus	Multi wired connection between a Device and the CPU
Byte	A unit of storage in the computer
CBF	Copy by File
Chain	To obtain keyboard Input from a file created for this purpose
Char	A character is usually a printable value (equivalent to a byte)
CPU	Central Processing Unit
CR	Carriage return
Cass	Cassette recording Device
Char	Character (in the ASCII range 33 to 127)
Cur	Cursor
DAM	Data Address mark
DCB	Device Control Block
DDGA	Default Directory Granule allocation
DDSA	Default Directory
DDSL	Default Directory Starting Lump
DEC	Directory Entry Code - refers to the position in the Directory
DFG	3 keys which when pressed together pass control to MINI-DOS
DIR	Disk Drive Directory
DOS	Disk Operating System
DP	Double Precision
DRQ	Data request (from a Device)
EOF	End of File
EOL	End of line
EOM	End of message
EOR	End of Record
EOS	End of Statement
ESF	Exatron Stringy Floppy (a small Data storage Device)
Ext	Extension to a File name (e.g. prog/BAS)
Extent	2 bytes of control data inside an File Directory Entry
FCB	File Control Block
FDC	Floppy Disk Controller
File	A collection of data comprising one entry in the Directory
FPDE	File Primary Directory Entry
FXDE	File Extended Directory Entry
GAT	Granule Allocation Table

ABBREVIATIONS

GPL	Granules per Lump
Hash	Mathematically coded sum of the ASCII values of a file
HDDN	Hard Disk Drive Number
HIMEM	Highest Memory location available for use
HIT	Hash Index Table
IFF1	Interrupt flip flop No 1 - This is the Interrupt flag
IFF2	Interrupt flip flop No 2 - This is the Interrupt storage flag
I/O	Input or Output to or from a Device
INT	Integer
INTRQ	Interrupt request from a Device or routine
IPL	Initial Program Loader
JKL	3 keys when pressed together invoke the Screen-print utility
LF	Line feed
LOC	Currnet location in the File
LOF	Length of File
Logical	Logical seems in most cases to mean counting from 0
LRL	Logical Record Length
LSB	Least significant byte
Lump	A section of disk space represented as 1 byte of the GAT sector
ms	Millisecond
MSB	Most significant byte
PC	Program Counter (Reg pair holding the next execution address)
RAM	Random access Memory
ROM	Read only Memory
RST	The Restart instruction
SCC	Space Compression Codes
Sector	Block of 256 bytes of storage on Disk
SFS	Section's First Surface
SP	Single Precision
SPC	Space Compression Codes
SPG	Sectors per Granule
SSC	Section Surface Count
TC	Track Count
TD	Type of Drive
TI	Type of Interface
TOPMEM	The highest Memroy location available for use
TPS	Tracks per Surface
Track	The unit of disk space the drive head(s) passes over in 1 revolution
TSR	Track stepping rate
Vector	An address which contains a jump to a routine
VFS	Volume's first Sector
VFS	Volume's first Sector
VSC	Volume Surface Count
WRA1	Work Area number 1 (the Accumulator)
WRA2	Work Area number 2 (the auxiliary Accumulator)
Zap	To change bytes in a program

I N D E X

I N D E X

I N D E X

I N D E X