7.0

Machine Language Utility

7.1

Introduction

The Utility provides a set of facilities to develop and debug programs in machine-code. It has the ability to keep a safe copy of the registers for a program being debugged. These can be displayed and modified, as can the mode of operation of the Real World Bus, and the Timer and Interrupt controller. The memory contents can also be displayed and changed, and can be stored on, or loaded from, disc or cassette. A machine code program can be debugged using breakpoints, or an instruction - by - instruction tracing facility.

7.2

User Interface

When the Utility is entered from BASIC by means of the UT command it prints its sign-on message: P.C. UTILITY V3.3
The message is followed by the prompt character ">". Whenever the Utility prints this character, it is waiting for another command. The format of commands is always a single letter followed possibly by one or more numbers. No separator is required between the letter and the first number. Numbers are always in hexadecimal, and are terminated by a space or carriage return. The utility always uses the last hex characters type d in , two or four depending on the required range of the number. So G12345678 is equivalent to G5678, because a 4 digit hex number is required
F0000 FFFF 5566 is equivalent to:
F0000 FFFF 66 as the third number is required to have 2 digits.
Any 2 or 4 digit number can be terminated early and the Utility will use the number of digits typed. So:

G0003
G003
G03
G3

} These are all equivalent.

When there is any kind of an error, the Utility prints the character "?". This is the only possible error message.
When the utility is tracing a program or printing memory contents the display can be halted by use of the BREAK key.
Some functions require the use of a terminator apart from space or carriage return. This is called an "ESCAPE", and the key used is the "cursor Left" on the far left of the keyboard.
During the description of commands, some special signs will be used. They are:

␣ for SPACE
↵ for CARRIAGE RETURN
← for ESCAPE (LEFT ARROW)

Characters typed in are underlined in the examples.

You will return to BASIC by typing "B".

7.3

Utility Commands

This section describes in detail the four classes of commands that assist the user in his program development in the utility mode. Abbreviations used in the text are defined as follows:

adr :	ADDRESS
ladr :	LOW ADDRESS
hadr :	HIGH ADDRESS
dadr :	DESTINATION ADDRESS
badr :	BASE ADDRESS of PROM Reference

The address is a string of four hexadecimal numbers. If the string is longer than four digits, the utility accepts the four rightmost digits as the address. This feature provides the advantage that if a mistake is made while entering an address, one can disregard the mistaken figures and keep entering figures until the four rightmost digits are correct. Command arguments can be separated by either space or comma.

The four classes of commands are:

Memory Commands: These commands enable the user to trace his program while it is running, or single-step it. He can also display blocks of memory bytes, and insert user's program or data.

Register Commands These commands afford the facility to examine and modify the 8080 registers, and the vector and initialization bytes. In general these commands allow the user to initialize the DCE card before transferring control to the user program.

Hexadecimal I/O Commands With these commands the user can read file, write file.

CLASS 1. MEMORY COMMANDS

7.3.1

LOOK: L adr ladr hadr

When the sequence is terminated with the "RETURN" key the command initiates transfer to the user mode. The program counter is loaded with the address specified. After each instruction execution, the contents of all the CPU registers are displayed on the console:

I = 1043 A = 02 F = 02 B = 00 C = 00 D = 00 E = 05 H = 00 L = 00
S = P = 1045

Where "I" is the address of the instruction just executed, all the instructions between the low and high address specified will be traced. To temporarily abort program execution, press and hold the "BRAK" key during the last desired trace line, until the line is completed. To continue program execution after the break, just type "L" followed by the "RETURN" key. Tracing will continue with the command whose address is equated to "P" on the last trace.

While under the control of the Utility during the break, all functions, may be used without affecting subsequent LOOK restart. The programmer is thus free to access and modify the entire register and memory area during the break.
Before restarting execution, the "trace window" can be changed from the one originally specified with this command. To alter the trace window continue program execution by typing:

L ladr hadr

followed by a return. The LOOK function restarts with the new trace limits. Whenever the LOOK function is initiated by typing all three arguments, the system is initialized as described in Section 4.1. However, when LOOK is restarted by just typing L, or L with the new trace window arguments, only the CPU registers are restored. No other states are modified. This allows normal continution of a program after the BREAK.

The BREAK key abort feature is always active, even when the program is running outside the trace window. This feature allows escape from a program loop while saving the Program Counter.

7.3.2

DISPLAY: D ladr hadr

When terminating the sequence by the "RETURN" key, the console displays consecutive memory bytes in hexadecimal starting with the one specified by the low address and ending with the one specified by the high address. Each line is preceeded by the memory address of the first byte on the line.

Example: D1000, 110A

Pressing and releasing the BREAK key aborts printout.

7.3.3

GO: G adr

When the sequence is terminated with the "RETURN" key, the command initiates transfer to the user mode. The system is initialized, and program execution starts. The user program stored in the memory controls the CPU until control is returned to the utility. The address in the command is optional; if no address is given, only the 8080 registers are restored from the save area, and not the GIC and TICC initialization bytes. Execution starts with the saved P (program counter) value. Entering "G" without address allows restarting the system after a breakpoint without reinitializing.

Example: G1040

This command transfers control to the program segment starting at the memory location 1040H.

7.3.4

FILL: F ladr hadr byte

When terminating the sequence with the "RETURN" key, the memory space defined by and including the low and high addresses is filled with the constant byte given. If no constant value is given the memory space will be filled with zeroes.

Example:

F1010	101A	FF	fill area from 1010 to 101A with FF
F1010	101A		fill area from 1010 to 101A with ØØ

7.3.5
SUBSTITUTE: S adr

When terminating the sequence with space, or the "RETURN" key, the screen displays the content of the byte specified by the address given. A new value can now be typed in. This value will replace the current content of the addressed byte when the next separator, space or comma or "RETURN", is entered. At the same time, the content of the next higher order byte is displayed for substitution. To leave a byte unchanged the space bar or "RETURN" is used after the display of the byte.

Example: S1000 3D-8F 1A = CB-3F 81-AE 78-FA

In the example above, digits entered by the user are underlined, and the space bar was used as separator. To return to the utility, press the "LEFTCURSOR" key. After escaping the sequence, the memory locations starting from address 1000 to 1004 will have the following contents:

1000: 8F, 1001: 1A, 1002: 3F, 1003: AE, 1004: FA

7.3.6
MOVE: M ladr hadr dadr

The MOVE command, when terminating the sequence with the "RETURN" key, moves a block of memory specified by the low and high addresses to a destination beginning with the destination address.

Example: M1000, 100A, 1100

After executing the above command, the program segment starting at address 1000 and ending at address 100A has been moved to a starting address at 1100, and it will occupy all the bytes up to and including address 110A. The original program segment at location 1000 is not destroyed.

The MOVE command is useful during program development when an instruction must be inserted into the program already stored in the RAM memory. For example, assume that three bytes must be inserted into a program field ranging from RAM location 1040 through 1075. The new bytes must occupy locations 1046, 1047, and 1048.
Using the MOVE command, the program segment ranging from 1046 through 1075 can be shifted right three bytes:

M1046 1075 1049

The three new bytes can now be inserted. Caution: the MOVE command does not adjust reference addresses within instructions.

CLASS 2. USER REGISTER COMMANDS

7.3.7
EXAMINE: X

When the above command is terminated by pressing the "RETURN" key, the screen displays the following CPU registers: Accumulator, Flags, Registers B through L, Stack Pointer, and the Program Counter.

Example:

X
A = 00 F = 46 B = 20 C = 44 D = 10 E = BF H = 11 L = 7A S = 11BE
P = 1040

The bit assignment of the flag-byte is as follows:

B7	SIGN
B6	ZERO
B5	ALWAYS ZERO
B4	AUXILIARY CARRY
B3	ALWAYS ZERO
B2	PARITY
B1	ALWAYS ONE
B0	CARRY

7.3.8
EXAMINE REGISTER: X reg

This command is exactly like the substitute command except that it allows substitution or initialization of the user-register copy area.

Example: Suppose we wish to initialize the accumulator to the value of 35 and register B to the value of FF. We can do this task in either of the following ways:

XA 00-35 46- 20-FF
or
XA 00-35
XB 20-FF

The digits entered by the user are underlined. In the first example the space bar was used as separator, and the value of the flags remained unchanged, since no replacement value was entered. In the second example the first substitution was terminated by the "LEFT ARROW" key.

7.3.9
VECTOR EXAMINE: V

When the "RETURN" key is pressed after the command, the console displays the contents of the user initialization and interrupt-transfer vector bytes.

Example:
V
0 = 00 M = 00 T = 10 G = 20 1 = 106F 2 = 1089 3 = 0040 4 = 0040
5 = 0040 6 = 0040 7 = 106F.

7.3.10
VECTOR EXAMINE BYTES: V byte

The function of this command is the same as that of the substitut or examine register commands. It allows changing the contents of the transfer vector or initialization bytes.

Example: V2 1089-1100

When the "CURSORLEFT" key is pressed after the sequence above, the interrupt 2 vector address is changed from 1089 to 1100.

CLASS 3 HEXADECIMAL I/O COMMANDS

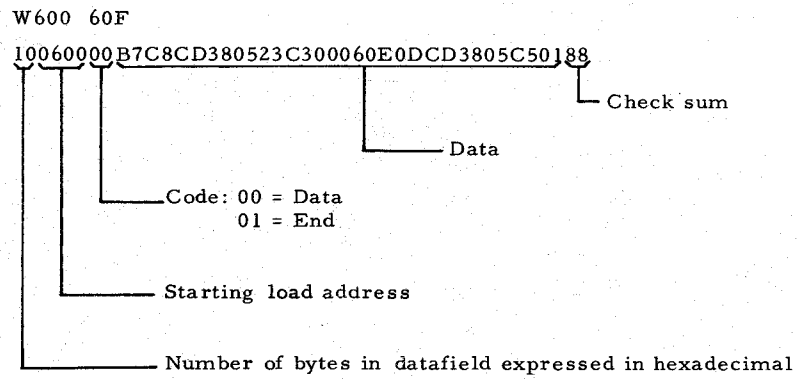
7.3.11
READ: R adr

The address in the command is optional.
Pressing the "RETURN" key after the command, initiates action. The READ function will start reading the binary file from tape or disc as soon as the tape recorder or disc drive is turned on. While reading the tape, the utility checksums each record. If a read error occurs, the error exit is taken, the reading stops, and the control is returned to the user. In this case the tape may be read again by backing it up at least one record. The reading continues until the end of file record is read.

7.3.12

WRITE: W ladr hadr

After pressing the "RETURN" key the hexadecimal content of the memory range specified by the low and high addresses is output to the tape or disc. The format of this output is the packed hexadecimal format described below.



W0␣FFF␣GEORGE↵

Writes the area of memory from 0 to FFF to disc or cassette under the name "GEORGE".

W0␣1F↵

Writes the area 0 to 1F on cassette with no name. Unnamed files should not be used on disc. It is loaded back into exactly the same addresses as it was written from.

R1000␣FRED↵

As above, but the data is read into addresses 1000 hex bytes higher than it was written from.

R↵

The next binary file on the cassette is read into memory. No offset is used. Note that unnamed files should not be used with discs.

The files created by the W and read in by the R command have a file type of 1. They cannot be accessed by, and will be ignored entirely by the LOAD, LOADA commands of BASIC. Similarily R will not read in files of types other than 1.

File names include every character typed between the space and the carriage return. There is no "character delete" facility, so great care should be taken.

Decimal	Character	Decimal	Character	Decimal	Character
000	NUL	031	US	062	>
001	SOH	032	SPACE	063	?
002	STX	033	!	064	@
003	ETX	034	"	065	A
004	EOT	035	#	066	B
005	ENQ	036	$	067	C
006	ACK	037	%	068	D
007	BEL	038	&	069	E
008	CH DEL	039	'	070	F
009	TAB	040	(	071	G
010	LF	041	)	072	H
011	VT	042	*	073	I
012	FF	043	+	074	J
013	CR	044	,	075	K
014	SO	045	-	076	L
015	SI	046	.	077	M
016	↑ CURS	047	/	078	N
017	↓ CURS	048	0	079	O
018	← CURS	049	1	080	P
019	→ CURS	050	2	081	Q
020	Shift+↑	051	3	082	R
021	Shift+↓	052	4	083	S
022	Shift+←	053	5	084	T
023	Shift+→	054	6	085	U
024	CAN	055	7	086	V
025	EM	056	8	087	W
026	SUB	057	9	088	X
027	£	058	:	089	Y
028	¢	059	;	090	Z
029	GS	060	<	091	(
030	RS	061	=	092	\

Decimal	Character	Decimal	Character	Decimal	Character
093	)	123	{		
094	↑	124	|		
095	←	125	}		
096	`	126	~		
097	a	127	DEL		
098	b				
099	c				
100	d				
101	e				
102	f				
103	g				
104	h				
105	i				
106	j				
107	k				
108	l				
109	m				
110	n				
111	o				
112	p				
113	q				
114	r				
115	s				
116	t				
117	u				
118	v				
119	w				
120	x				
121	y				
122	z				

LIST OF SOME USEFUL POKES

POKE #2C4,#FF FORCE A BREAK

OUTPUT

POKE #131,0 OUTPUT TO SCREEN + RS 232
,1 OUTPUT TO SCREEN
,2 TO EDIT BUFFER
,3 TO DISC — inc 5

INPUT

POKE #135,0 INPUT FROM K. B./SCREEN
,1 INPUT FROM STRING
2 INPUT FROM EDIT BUFFER TO PROGRAM AREA

TAPE CONTROL

POKE #40,#28 TAPE 1 ON
#40,#18 TAPE 2 ON
#40,#30 TAPE 1 AND 2 OFF

PCKE #13D,#10 CASSETTE PORT 1 ACTIVATED
#13D,#20 " " 2 "

SWITCH FLOPPY DRIVE

POKE #730,#30 FLOPPY DRIVE 0 ACTIVATED
#730,#31 FLOPPY DRIVE 1 ACTIVATED

AM 9511

UT
>SFBØØ
>
>B

UNIT FLOPPY DISK

UT
>Z3
>XA 30 USE DRIVE N° Ø
31 " " " 1
>G B6
>B

TOP OF STACK #F900
BOTTOM OF STACK #F800

POKE #2C4,#FF : FORCE A BREAK IN PROGRAM

ON TAPE "ACTIVATE"

TO ACTIVATE FLOPPY (2C5 TO 2E2)

2C5 C3 58 Ø5 C3 F2 Ø5 C3 12 Ø6 C3 A1
2DØ Ø5 C3 FB Ø5 C3 FC Ø6 C9 ØØ ØØ C3 75 Ø6 C3 29 Ø6
2EØ C3 5C Ø6 (2E2)
2A0 08 5D 08 5E 08

TO ACTIVATE CASSETTE (2C5 TO 2E2)

2C5 C3 B8 D2 C3 F1 D2 C3 27 D4 C3 25
2DØ D3 C3 40 D3 C3 45 D4 C3 A2 D3 C9 ØØ ØØ C9 ØØ ØØ
2EØ C3 B4 DD (2E2)

2AØ 33 ED 03 F6 03 50 B3 C5 E8

SOFTWARE PROTECTION

1. Write program in BASIC (Avoid putting REM)

2. UT

3. D2A1 2A4 (Pointers) ↵
   2A1 # # # #
   Low High Low High
   VAL 1 VAL 2

4. SAVE ON CASSETTE BY
   W (VAL 1 + 1) (VAL 2) FILE NAME (without double quote)

5. Protect by
   F(VAL 1+1) (VAL 2) C↵ C(C = Hex code for form feed)

6. B (return to BASIC)

7. SAVE ON CASSETTE (SAVE "FILENAME")
   When loading from cassette you cannot LIST nor EDIT anymore as all information is scrambled.

WHAT TO DO IF AN ACCIDENTAL RESET HAPPENED DURING PROGRAM KEYING OR AT END OF PROGRAM

1. Push on BREAK
2. Type UT return
3. Type S29F and 6 x Space bar, result is b a x x x x
4. Note b a x x x x
5. Cursor (←)
6. Type S a b space bar, result is x x
7. Note x x
8. Cursor (←)
9. Press B (BASIC)

If you accidentally RESET

1. Type UT return
2. Type S29F press 6 times space bar; result is x y &&&&
3. Change the 6 positions if different to what you noted.
4. S a b change the 2 " " " " " " "
   Cursor
5. Press B
6. Type EDIT press and BREAK Space

SAVING AND RELOADING A DRAWING

After you draw the picture for saving
Press on BREAK
Type MODE ? A (? being the mode in which you draw the picture)
Type UT Return
Type W XXXX BFFF PICTURE 1

To reload the picture
Type MODE ?A (? being the mode in which the picture was drawn)
Press UT Return
Type R

MODE 1
2 A B350 TO BFFF
3A A440 TO BFFF
4
5 5670 TO BFFF
6