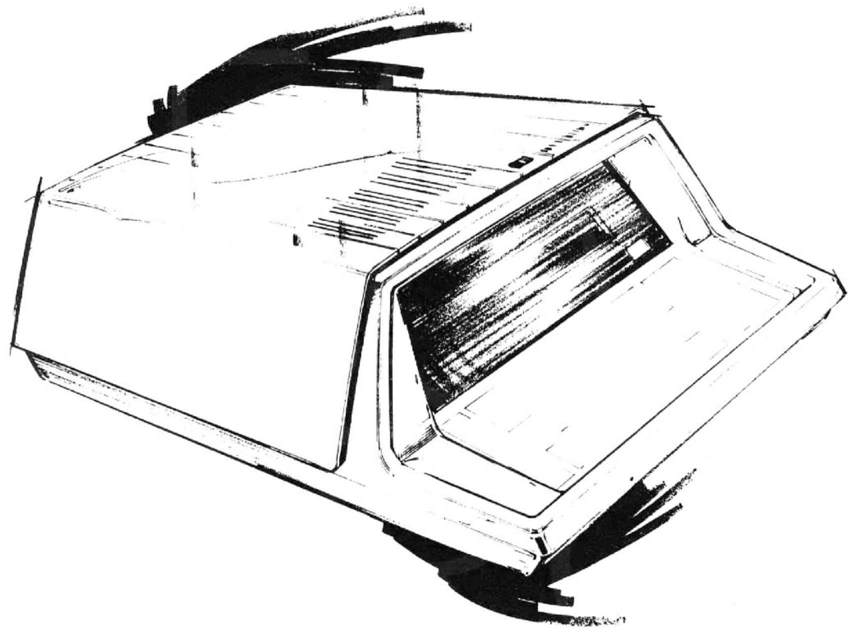
MDT . . .

MICROCOMPUTER DEVELOPMENT TERMINAL



MOS

MICROCOMPUTERS

The MDT650 is a high level system development tool which provides the user with new hardware and software techniques for verifying system designs prior to a finalized design commitment . Interactive design techniques allow the user to engage this system as a total development tool for preproduction and final production systems.

GENERAL SYSTEM DESCRIPTION

The MDT650 provides all of the hardware and software necessary to develop and assemble user programs. In addition, the unit contains several features which make it a very powerful tool for debugging both the hardware and the software for the system being designed. The unit can pay for itself very rapidly by eliminating time-sharing costs and by greatly reducing the total system development time.

MDT650...THE MICROCOMPUTER DEVELOPMENT TERMINAL

Two MCS650X series microprocessors are used to control all system functions. Interaction with the MDT650 is normally with the integral keyboard/display, however TTY or other terminal device can also be used. Expandable I/O configurations are TTL compatible.

The standard MDT650 system allows the user to assign up to 65K of memory as desired (with independent address and data bases). The ROM resident system monitor includes all necessary functions for program loading, debugging, and execution. A resident assembler may be used to assemble machine instructions. A resident editor provides source language editing capability.

HARDWARE FEATURES

. Integral Keyboard with separate function keys for control.
. Built-in 32 character display.
. Serial input for interfacing a serial terminal.
. Two address traps are provided to halt the user processor on: any address; instruction; read cycle or write cycle.
. Two scope syncs: address trap sync and instruction fetch cycle cync.
. Single instruction mode with firmware enhancement.
. Trace stack memory for storing the last 128 machine cycles.
. Seven board positions provided for user memory, bus light display, I/O or user options such as custom wire wrap boards.
. Control firmware for the assembler, disassembler, and test editor that is independent of the user's 65K memory limit.

SOFTWARE SUPPORT

The MDT650 software consists of three programs: the assembler, the disassembler and the text editor.

The MDT assembler is upwards compatible with the MCS650X Cross-Assembler. Features include:

. Can assemble from source tape or user RAM.
. Six character labels and symbols.
. Free-form entry of source statements.
. Symbol table output.
. Error flags on listing.
. Assembled program is ready for execution.

The MDT disassembler commands include:

. Load interface file (symbols and code).
. Go to address. Execute one instruction and automatically disassemble.
. Run. Executes user program.
. Trap address and mode. Sets appropriate instruction.
. Forward one instruction from current location in stack.
. Backward one instruction.
. Show last/next cycle. Address, label at this address (if any), data, and decoded flags are displayed.

The MDT text editor program saves considerable money and frustration during the process of generating the system program.

. Load text buffer
. Insert or delete line
. Insert or delete character at current position in line
. Forward/backward one character
. Step forward or backward to next blank (Very useful for jumping from label field to OP Code field, etc.)
. Go to top or bottom of text
. Index up or down one line
. Find a specified string of characters
. Provide output text to printer/punch

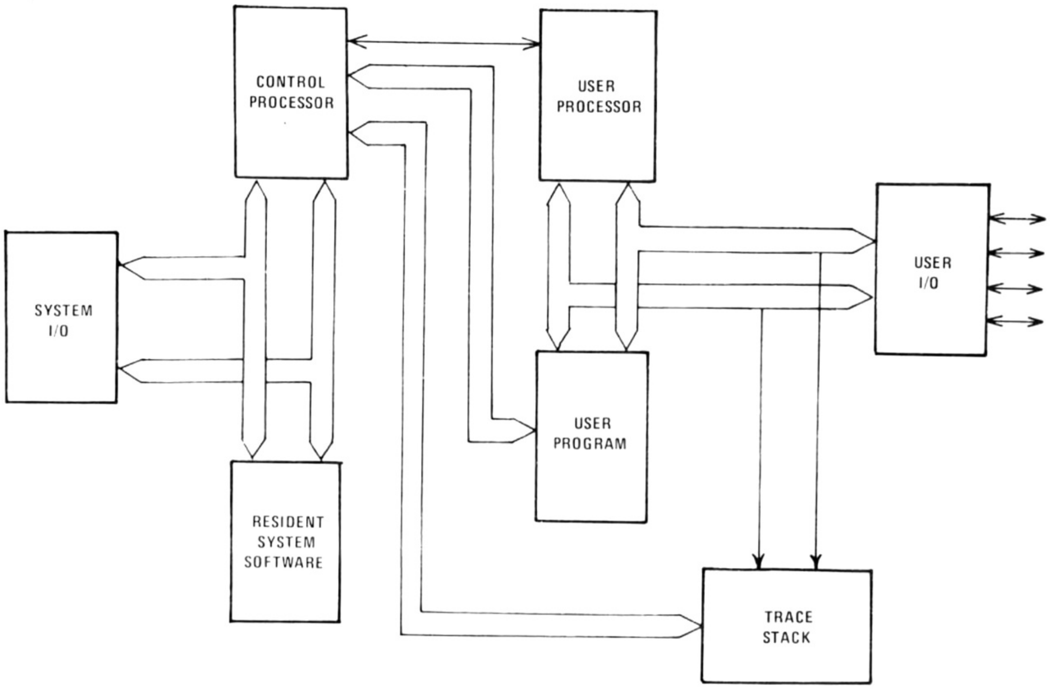
STANDARD EQUIPMENT (BASE SYSTEM)

Dual Micro Processor Module
RAM Memory Module
Program trace and address trap board
I/O board for Keyboard, Display and Peripherals
Resident Monitor ROM Module
Chassis with 14 Board Slots
Power Supplies
Finished Cabinet
Keyboard and Display
Assembler
Text Editor
User's Manual
MCS650 Assembly Language Programming Manual
MCS650 Assembly Language Reference Card

OPTIONS

PROM programmer available for 82S115, 2708 or 1702A.
Wire-wrap boards for custom designs.
Extender board module

Light display board to display address and data busses
4K RAM board
8K PROM board
2K RAM/4K PROM board
I/O boards...allows interfacing system to various peripheral devices and terminals
Floppy disc
ICE (In-Circuit Emulator)



BLOCK DIAGRAM OF MDT 650

MOS TECHNOLOGY, INC.

VALLEY FORGE CORPORATE CENTER
950 RITTENHOUSE ROAD NORRISTOWN, PA. 19401
TEL: (215) 666-7950 TWX:510/660/4033

MICROCOMPUTER DIGEST

Volume 2, Number 10 April, 1976

MICROCOMPUTER DEVELOPMENT TERMINAL

MOS Technology has announced the MDT650 Microcomputer Development Terminal for modeling new 650X designs. The MDT650 terminal is used to evaluate and debug the user's programs and system hardware. The unit can be configured to a wide range of design applications through user-system emulation. The MDT650 incorporates a completely separate processor and bus structure for application emulations, thus eliminating emulator executive overhead time during real time execution.

Interaction with the MDT650 is normally with the integral keyboard/display. However, a TTY or other terminal device can be used. The expandable port configurations are TTL compatible.

The standard MDT650 system allows the user to assign up to 65K of memory as desired (with independent address and data bases). The ROM resident system monitor includes all necessary functions for program loading, debugging, and execution. A resident assembler may be used to assemble machine instructions. Interpretation of machine codes is linked to original op-codes, labels and mnemonics. A resident editor provides source language editing capability.

The MDT650's bare system price of $3950 includes dual microprocessor module, RAM memory module, program trace and address trap board, I/O board, resident monitor ROM module; chassis with 14 board slots, power supplies, cabinet, keyboard and display, system monitor, assembler text editor, user's manual, MCS650 assembly language programming manual, and MCS650 assembly language reference card. Options planned include a PROM programmer available for 82S115, 2708 or 1702A, wire-wrap board for custom designs, extender board module, address and data bus display board, 4K RAM board, 8K PROM board, 2K RAM/4K PROM board, I/O board, user RAM write protect option, high speed ports for printer, card readers, floppy disc interface, In-Circuit Emulator, and system software source listings.

MOS Technology reports availability of the base system in the second quarter, 1976 with the various hardware options becoming available in the second and third quarters, 1976.

A

USER'S GUIDE TO THE MDT 650

A.1 INTRODUCTION—WHAT'S AN MDT 650?

The Microcomputer Development Terminal (MDT) 650 is a *special purpose* microcomputer which is intended to serve as a software development aid in the design of systems which use the 650X series of microprocessors currently being manufactured by MOS Technology and the Rockwell International Corporation. We will primarily be concerned with the 6502 microprocessor in the 650X series.

In terms of size, shape, and weight, the MDT 650 resembles an electric office typewriter. The keyboard has additional keys for control and there is a single line alphanumeric display consisting of 32 (20_{16}) characters. It is also possible to connect peripheral equipment such as cathode ray tube terminals (CRTs) and teleprinters such as the model 33ASR Teletype with paper tape reader and punch.

For purposes of this discussion, the MDT is much more than just another "piece of hardware." Rather, it is the vehicle for discussing the general area known as microcomputer systems design aids. The reader should ponder (maybe even meditate!) on each of the following:

a) what functions a software aid can and cannot do for the user.

b) the relative advantages and disadvantages of including automatic features and defaults in the design of software development aids themselves.

c) software development aids, like the microcomputer digital computers they really are, do not lessen in any way the need for the

[Software for the MDT650 was done by COMPAS, Computer Applications Corporation, Ames, Iowa. Information relating to the MDT650 software is used with the kind permission of COMPAS.]

user to be meticulously careful about what appears to be an infinite number of "nit picking" details. All digital computers are "impartial" graders, which seldom, if ever, give "partial credit" to the user for work submitted which is only slightly in error.

Figure A.0 presents a functional block diagram of the MDT as viewed by the user. It shows the two primary modes designated as EDITOR and MONITOR; as well as the key stroke sequences to enter each of the five secondary modes: LOAD, EDIT, ASSEMBLE, EXECUTE, and OUTPUT MODES.

A.2 SUMMARY: HOW TO USE THE MDT 650

This entire section of the appendix describes in detail how to use the MDT 650. It may best be summarized by listing and briefly explaining the five modes of operation.

1.LOAD MODE

In the LOAD MODE, TEXT is stored into the MDT 650 memory. TEXT includes the source program, comments, and assembler directives, such as .END statements. It could also be poetry.

TEXT may be entered into the MDT 650 memory via the MDT 650 keyboard, but more efficient use may be made of the MDT 650's special capabilities if off-line peripheral devices are used to prepare paper tape. A considerable number of errors (but not of every possible type) are tolerated on the off-line prepared paper tape since they may be removed easily in the EDIT MODE.

2. EDIT MODE

TEXT currently stored in the MDT 650 memory may be modified and/or corrected by inserting a character, deleting a character, inserting a line, or delating a line.

It is also possible with a single key stroke to display the top line of TEXT or the bottom line of the TEXT.

3. ASSEMBLY MODE

The source program is converted to machine language. The assembler is a two pass assembler and about 23 error code diagnostic messages are available to aid in detecting errors.

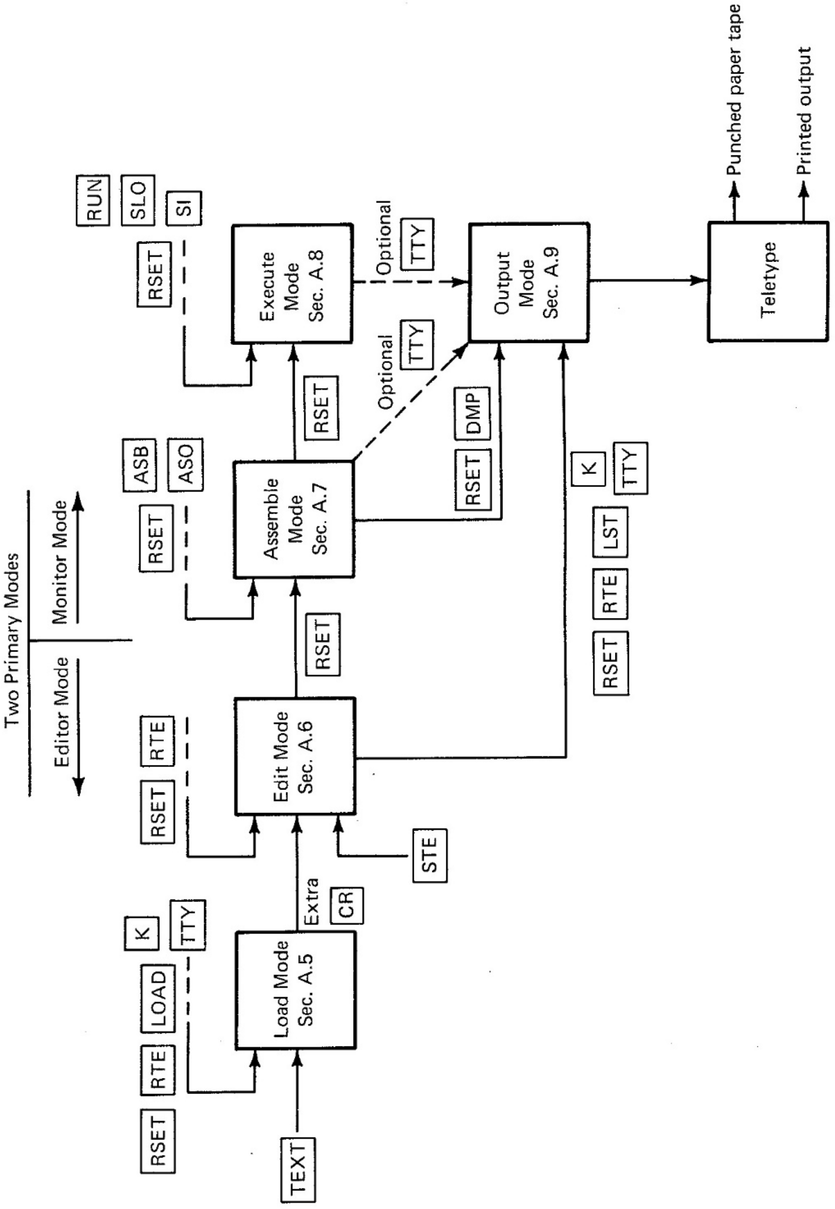


Figure A.0 Functional block diagram of the MDT as viewed by the user

4. EXECUTION MODE

The entire computer program may be executed or the execution can be done one line of code at a time (called single-step). Single-step operation may be controlled manually or it may be automatic with a variable, but selectable, interval of time between the single-steps.

An additional feature, called TRACE, permits the optional display of the contents of all of the microprocessor registers immediately following the execution of each single-step.

Two hardware breakpoints may be implemented to stop the user's microprocessor on any desired address when one of four user selectable conditions are satisfied.

5. OUTPUT MODE

Printed outputs and optional punched paper tapes corresponding to the corrected TEXT and the object code (machine code) are available.

The printed versions of the TEXT have three possible formats: a) exactly as entered, b) "tabbed" for ease of human interpretation, and c) "squished" to reduce the length of each line which reduces printing time and improves the possibility of showing the entire line on the MDT 650 display.

The format of the punched paper tape is correct for reading into some (but not all) 6502 microprocessor based systems such as the KIM-1 microcomputer.

A.3 EXAMPLE A1—A BENCHMARK EXAMPLE

Example A1 is a benchmark example in the usually accepted sense, namely, it serves as a reference point to compare corresponding results, such as the numerical answers, and the amount of work involved in working a problem two or more different ways. Example A1 has intentionally been chosen to be so simple that results obtained with the MDT 650 may be easily compared with those obtained manually. These comparisons include the machine code obtained during assembly, symbol table generation, addition in both binary and decimal modes, and the register contents at the end of each step during program execution.

Example A1. Add two positive numbers of one byte each located in two memory locations and store the sum into a third memory location.

Comments.

1. We are considering only positive, one byte numbers.
2. In order to obtain the correct numerical results, we will have to consider the possibility of a *carry in* because the 6502 microprocessor has only an add with carry (ADC) and no simple just add (ADD) instruction.
3. We need to determine if a *carry out* occurred if we wish to verify the numerical answers in some cases.

A.4 SOURCE AND OBJECT CODE FOR EXAMPLE A1—MANUAL ASSEMBLY

The source and object code for Example A1 may be written using many different formats. If only one person is involved in writing the source code, and that same person will manually convert it to object code, no special format needs to be followed and no other special restrictions (such as the length of labels being limited to 6 characters beginning with an alphabetical character) exist. However, seldom is only one person involved, and even if that is the case, a formatted, systematic approach is desirable to prevent errors and to make the program easier to read and to understand. The ever present problem of documentation is also partially solved at this point if a systematic, standardized format is followed. We will therefore use the format dictated by the 6502 Assembler Language even though at this point we will do the assembly *manually*.

The source and object codes for Example A1 are shown in Table A.1.

A.5 MDT 650 IN THE LOAD MODE

The primary purpose of the LOAD MODE is to provide a means of transferring and storing the TEXT into the user side of the MDT 650 memory. TEXT includes the source program, comments, and assembler directives, such as .END. We note in passing that the TEXT could consist of cooking recipes or even poems. Hence an unusual application of the MDT 650 might be to modify recipes or compose poetry. We will primarily concern ourselves here with the normal, *intended applications* of the MDT 650, i.e., as a software development aid.

TEXT may be stored into the memory via the MDT 650 keyboard, but a more efficient use of the MDT's special capabilities is made if off-line peripheral devices are used to prepare the punched paper tape. The

TABLE A.1 Source and Object Code for Example A1 — *Manual Assembly*

OBJECT CODE				*SOURCE CODE*			
MEMORY ADDRESS	*BYTE 1*	*BYTE 2*	*BYTE 3*	*LABEL*	*OP CODE*	*OPERAND*	*COMMENT*
ØØØØ				NUM1			Designate one of the two numbers to be added as NUM1.
ØØØ1				NUM2			Designate the second of the two numbers to be added as NUM2.
ØØØ2				SUM			Designate the sum of the two numbers to be SUM.
						⋮	
ØØ1Ø	F8				SED		Set Decimal Mode.
ØØ11	A5	ØØ			LDA	NUM1	Load the accumulator with the numerical value of NUM1.
ØØ13	18				CLC		Clear carry.
ØØ14	65	Ø1			ADC	NUM2	Add with carry NUM1 + NUM2. (There is no ADD instruction.)
ØØ16	85	Ø2			STA	SUM	Store the sum of two numbers into memory location SUM.
ØØ18	4C	18	ØØ	WAIT	JMP	WAIT	The 6502 microprocessor has no halt or stop command. This is a jump to itself.

Comment 1: We have decided to store NUM1, NUM2, and SUM into memory locations ØØØØ, ØØØ1, and ØØØ2, respectively. Refer to Sec. 8.5 for more details.

Comment 2: We have also decided to begin storing the object code into memory locations beginning with memory location ØØ1Ø.

recommended procedure for preparing punched paper tape on a Model 33ASR Teletype is shown in Table A.2. Please note that errors have *intentionally* been made in the preparation of the paper tape so that we will be able to demonstrate the correcting and modifying capabilities in the EDIT MODE of operation.

Table A.3 presents the detailed key by keystroke sequences for loading the TEXT into the user side memory via a Teletype Model 33ASR paper tape reader. The suggested memory allocation in the user side memory is presented in a conventional format in Figure A.1. This particular memory allocation provides the maximum amount of contiguous (touching) memory for TEXT but at the same time it makes it impossible to utilize the interrupts. Other memory allocations are possible if necessary. Each block of 4K RAM may be moved to any one of 16 different locations by means of switches inside the MDT case.

Figure A.2 is another view of the TEXT as stored in the memory with the requested starting address SSSS and the ending address EEEE. When uisng the MDT in the EDIT MODE, this may be a more useful format than the memory map format shown in Figure A.1.

Table A.4 presents the detailed key by key stroke sequences for loading the TEXT via the MDT 650 keyboard. Only consecutive lines of TEXT may be entered and there is *no* editing capability except for the backspace key.

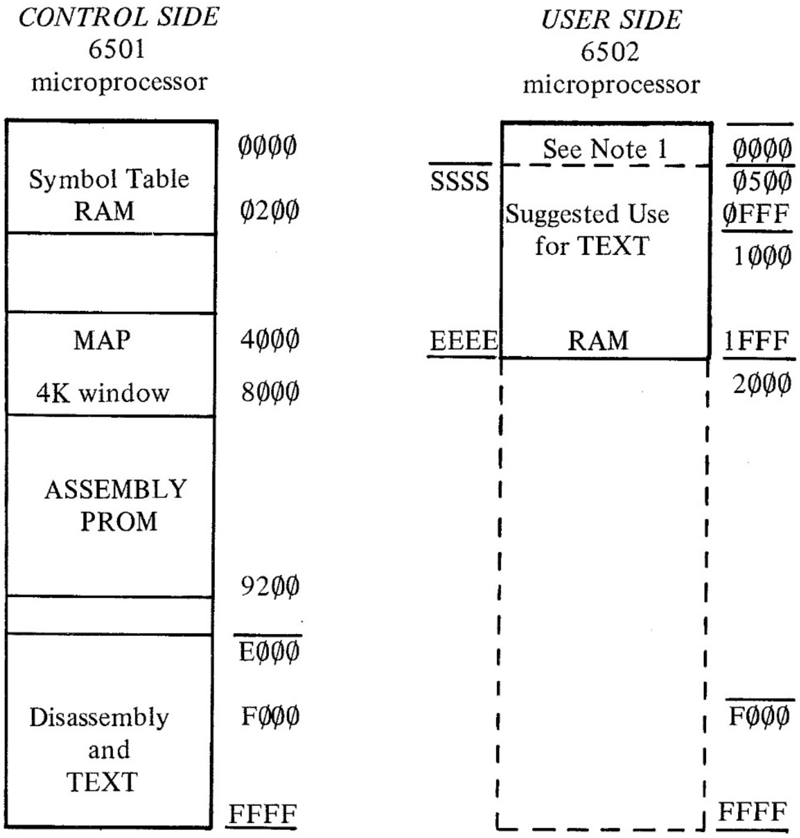


Figure A.1 Memory map of the user and control sides of the MDT 650. Only the user side is available to the user for storage. The control side is shown only for reference purposes and the memory locations are only approximate.

Note 1. These RAM memory locations are suggested for symbols and machine code (object code)

TABLE A.2 Off-Line Preparation of Paper Tape on the TTY Model 33ASR for Subsequent Entering into the MDT 650 Memory. The TTY is operated in the LOCAL mode with no connection to the MDT 650. Example A1 is used to illustrate the procedure.

Step 1: Apply 115 VAC, 60 Hz. to the TTY and turn switch to LOCAL.

Step 2: On the paper tape READER/PUNCH unit, press the ON button.

Step 3: Punch a series of RUBOUTS (all 8 holes punched with even parity). You may press the REPEAT and RUBOUT keys simultaneously to do this.

WARNING: If you punch a leader consisting of characters other than RUBOUTS, you may encounter difficulty later when reading the paper tape into the MDT 650. Even no print characters such as CR (carriage return) and LF (line feed) cause problems. For example, a CR as the first character read will cause an automatic transfer from the LOAD MODE to the EDIT MODE before any data are read in.

To be more specific: The first line encountered in TEXT which contains a CR as its first character will immediately cause an automatic transfer from the LOAD MODE to the EDIT MODE. This is a useful feature most of the time, but is can cause problems.

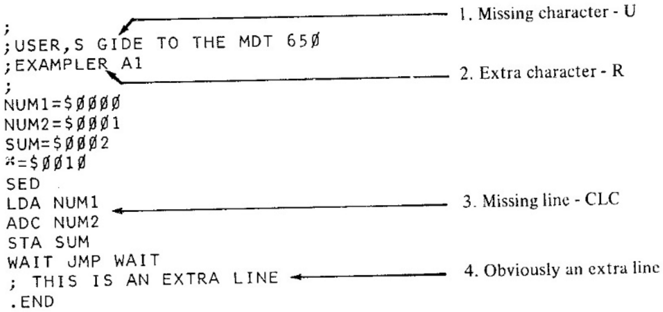
Step 4: Type your program on the TTY keyboard.

Comments: The input is relatively "free format" and every line of TEXT *may* begin in column 1. About the only format constraint is that a blank must appear between the LABEL (if any), OP CODE, and OPERAND fields. LABELS are limited to a maximum of 6 characters, the first of which must be alphabetical. None of the OP CODES may be chosen to be LABELS.

Step 5: Although not mandatory, it is suggested that that last line of your TEXT have as its first and only character a CR in order to automatically transfer from the LOAD MODE to the EDIT MODE.

Step 6: Punch a series of RUBOUTS to protect the end of your paper tape.

TABLE A.2 (continued) Actual TTY punched paper tape and TTY printed output for the source code of Example A1. Many errors have been deliberately made so that subsequently we will be able to demonstrate the correcting and modifying features of the EDIT MODE of operation.



```
;
;USER,S GIDE TO THE MDT 65Ø
;EXAMPLER A1
;
NUM1=$ØØØØ
NUM2=$ØØØ1
SUM=$ØØØ2
*=$ØØ1Ø
SED
LDA NUM1
ADC NUM2
STA SUM
WAIT JMP WAIT
; THIS IS AN EXTRA LINE
.END
```

1. Missing character - U

2. Extra character - R

3. Missing line - CLC

4. Obviously an extra line

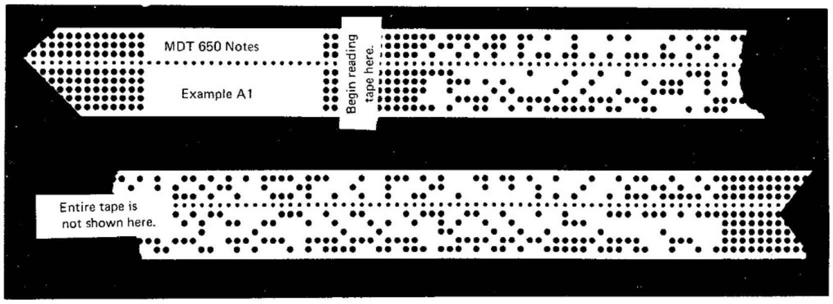


TABLE A.3 Procedure for loading off-line prepared punched paper tape into MDT 650 memory via a Model 33ASR Teletype paper tape reader. The specific data applies to Example A1 as shown in Table A.1.

Press Keys on MDT Keyboard	*See Displayed*	*Comments*
COLD START Plug MDT into 115 VAC. Plug fans into 115 VAC. There are no ON/OFF switches.	Blinking ? and S	The MDT is not being used. There is no power ON/OFF switch. Wait about 15 sec. for the display to blink.
TEXT	ENTER SSSS EEEE LL	MDT is asking for Starting Address, Ending Address, and the Line Length of the TEXT.
	⋮	
BATCH PROCESSING		There is a waiting line of users. Previous user has left and the MDT is in an unknown state. 115 VAC, 60 Hz. power is still applied.
RSET TEXT	ENTER SSSS EEEE LL	MDT is asking for Starting Address, Ending Address, and the Line Length of the TEXT.
Ø 5 Ø Ø 1 F F F 2 Ø		You have just entered the suggested addresses and line length as given in Figure A.1.
Check that TTY is one and switch in LINE position. LOAD TTY Move lever switch on TTY paper tape reader to START momentarily and release.	Lines of text are displayed, line by line, as they are read in. Miscellaneous characters may appear in the MDT display, but are usually not an indication of a problem.	Don't forget to load the paper tape in such a way that only rubouts are read before the TEXT characters are encountered. *No* characters, not even nonprint characters such as CR or LF, are permitted before the TEXT.
Any of the EDIT keys. (True only if the automatic transfer to EDIT MODE feature was used. If not, proceed to Table A.5.)		The first time user may wish to check the TEXT completely. This may be done by pressing the TOP key and then proceeding through the TEXT, line by line using the L+1 key. If the TEXT is correct, go to Table A.5, the ASSEMBLER MODE. If not, go to Table A.4, the EDIT MODE.

TABLE A.4 Keystroke sequences for entering the *Source Code* for Example A.1 (as shown in Table A.2) via the MDT keyboard in the *LOAD* mode. Only consecutive lines of text may be entered with *no* editing capability except for the backspace key.

Press Keys on MDT Keyboard	*See Displayed*	*Comments*
Plug into 115 VAC 60 Hz. (There is no ON/OFF switch.)	Blinking ? and S	Display is dark for about 15 sec. and then blinks.
TEXT	ENTER SSSS EEEE LL	MDT is asking for the Starting Address, Ending Address, and Line Length of the TEXT.
Ø5ØØ1FFF2Ø	Blinking + Characters are displayed as entered.	You have just entered the suggested starting and ending addresses given in Figure A.1 for the text and a Line Length of 2Ø hex.
LOAD K	Blinking +	MDT is now in the LOAD from keyboard mode.
NUM1=$ØØØØ CR	Characters are displayed as entered. When the CR key is pressed, a blinking + is displayed; if a line of text is rejected, a blinking ? is displayed.	Memory location ØØØØ is assigned to NUM1 CR key terminates each line of text from here on.
NUM2=$ØØØ1 CR		Memory location ØØØ1 is assigned to NUM2.
SUM=$ØØØ2 CR		Memory location ØØØ2 is assigned to SUM.
*=$ØØ1Ø CR		From here on, memory locations will be assigned consecutively beginning with ØØ1Ø hex.
SED CR		Set the decimal mode.
CLC CR		Clear carry.
LDA NUM1 CR		LOAD into the accumulator the value of NUM1.
ADC NUM2 CR		Add with carry NUM1 + NUM2.
STA SUM CR		Store the sum into the SUM memory location.
WAIT JUMP WAIT CR		Jump to itself.
.END CR		.END must be the last line of text.
CR		A line of text with the first nonblank character being a CR puts us into the EDIT mode.

A.6 MDT 650 IN THE EDIT MODE

The purpose of the EDIT MODE is to provide a convenient way for the user to correct and/or to change TEXT which is currently stored in the user side of memory of the MDT 650. In general, the ability to "edit" implies the capability to insert, delete, and change characters within a line without retyping the entire line; to insert, delete, and change entire lines of TEXT; and to locate lines containing key words. The MDT provides all of the above editing capability with the following keys:

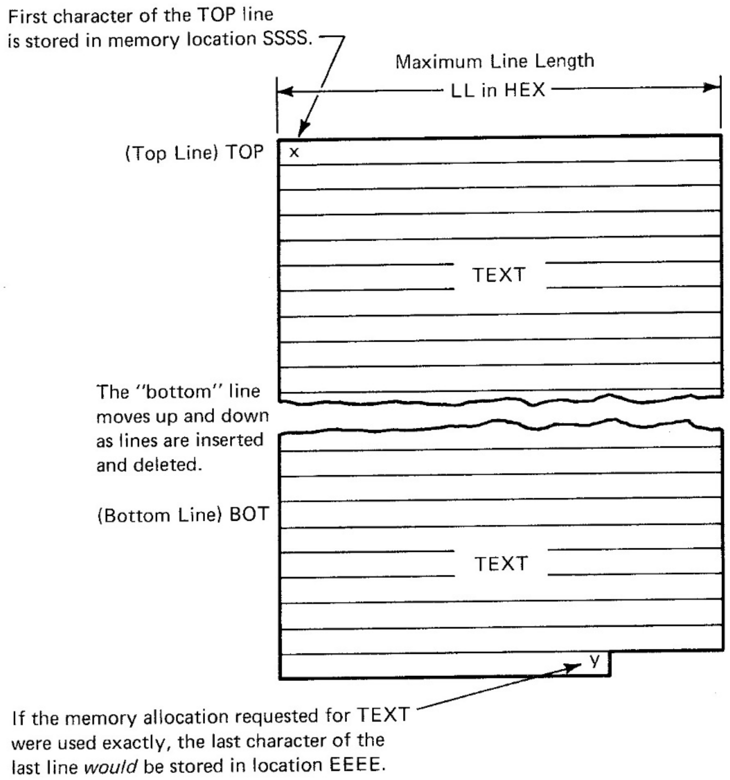

First character of the TOP line
is stored in memory location SSSS.
Maximum Line Length
LL in HEX
(Top Line) TOP
x
TEXT
The "bottom" line
moves up and down
as lines are inserted
and deleted.
(Bottom Line) BOT
TEXT
y
If the memory allocation requested for TEXT
were used exactly, the last character of the
last line *would* be stored in location EEEE.

Figure A.2 A user's view of the TEXT as stored in memory with the requested starting address SSSS and the ending address EEEE. When using the MDT in the EDIT mode, this may be a more useful viewpoint than the memory map shown in Figure A.1.

TOP Displays the top line of TEXT.

BOT Displays the bottom line of TEXT.

FND Locates and displays a line of TEXT selected by a key word or character string.

L+1 Displays the next line of TEXT.

L–1 Displays the previous line of TEXT.

C+1 Moves the blinking cursor one character position to the right, until it reaches the rightmost position, where it remains.

C–1 Moves the blinking cursor one character position to the left, unless it is already in its leftmost position.

DLL Deletes the line being displayed and moves all lines below this line, one line up, respectively.

INL The new line will be inserted *after* the line being displayed. After the INL key is depressed, the display will go blank, but it may take several seconds in long TEXTS. Do *not* attempt to insert characters until the display goes blank and a single, blinking + appears on the display. Terminate with CR.

BS Moves cursor one character position to the left each time it is depressed. The blinking character may be *replaced* by simply pressing the key corresponding to the desired character.

STE Restart Text Editor. Destroys all TEXT currently stored.

NEXT Moves cursor forward to blank or end of line.

LAST Moves cursor backward to blank or start of line.

DLC Delete character located in the cursor position.

Table A.5 presents a detailed key-by-keystroke illustration of the use of the EDIT keys as they apply to Example A1. At this time, the reader may wish to refer back to part 2 of Table A.2 to review the intentional errors which were made at that time.

A.7 MDT 650 IN THE ASSEMBLE MODE

The purpose of the ASSEMBLE MODE is to translate mnemonic or symbolic computer programs into actual machine code. Assemblers, although usually slightly different in detail, are essentially the same in substance. The three parts of the source program (TEXT) are handled as follows:

1. Mnemonics for op codes are mapped, one to one, to the corresponding op codes in machine code, usually hex.
2. Assembler directives are used to reserve storage, to initialize memory locations, and, in general, to direct information to the assembler.
3. User comments are stored, but otherwise ignored by the assembler.

TABLE A.5 Keystroke sequences for editing TEXT as loaded in user side memory of the MDT 650. Refer to Table A.4 for the TEXT (with errors) currently stored in memory.

Press Keys on MDT Keyboard	*See Displayed*	*Comments*
RSET RTE	Blinking ; and +	Pressing RSET and RTE is optional if automatic transfer to EDIT MODE was used.
L+1	;USER'S GIDE TO THE MDT 650	Second line of TEXT is displayed. ; blinks and alternates with + to indicate position of the cursor.
C+1 press (9 times)	;USER'S GIDE TO THE MDT 650	I blinks to indicate cursor position.
U	;USERS GUIDE TO THE MDT 650	Character U has been inserted as shown.
L+1	;EXAMPLER A1	; blinks and alternates with +
C+1 (8 times)	;EXAMPLER A1	R blinks and alternates with +
DLC	;EXAMPLE A1	Space between E and A blinks with + R has been deleted.
L+1 (7 times)	;LDA NUM1	; blinks and alternates with +
INL	Display remains unchanged for a few seconds; then a blinking + appears.	Do *not* attempt to insert characters until display goes blank and a single, blinking + appears. With long TEXTS, this may take several seconds.
C L C	CLC+ (+ blinks)	Blinking + indicates cursor position.
CR	CLC	First C blinks and alternates with +
L+1 (4 times)	; THIS IS AN EXTRA LINE	Line to be deleted is displayed
DLL	.END	. blinks and alternates with a +

Suggestion: As a recheck on the correctness of the TEXT now stored in memory, press the TOP key and then press L+1 key repeatedly to display all of the TEXT, line by line, consecutively.

TABLE A.6 TEXT after corrections were made in the EDIT MODE. Note the relatively free format which is acceptable.

```
;
;USER'S GUIDE TO THE MDT 65Ø
;EXAMPLE A1
;
NUM1=$ØØØØ
NUM2=ØØØ1
SUM=ØØØ2
*=$ØØ1Ø
SED
LDA NUM1
CLC
ADC NUM2
STA SUM
WAIT JMP WAIT
.END
```

Additional characteristics of the MDT 650 assembler are discussed very briefly below. For further information, as well as numerous examples, please refer to the CROSS ASSEMBLER MANUAL, MOS Technology, Inc.

Instruction Format. The full set of op codes for the 6502 microprocessor is available on the MDT 650 resident assembler. There is a limitation, however, on the type of expressions allowed. At the present time, only addition (+) and subtraction (–) are evaluated in the expressions. Multiplication (*) and division (/) are not implemented.

Assembler Directives. There are 8 assembler directives:

.BYTE Reserves and loads one or more bytes of memory.

.WORD Reserves and loads two bytes of data at a time.

.DBYTE Is exactly like .WORD except that bytes are stored in high byte, low byte, order. Warning: Files generated by .DBYTE *cannot* be used as indirect addresses.

.PAGE Causes an immediate jump to the top of a page and may also be used to generate or reset the title printed at the top of the page. In the MDT 650, this is not true. Instead, .PAGE causes 4 line feeds plus a new page heading rather than top of form.

.SKIP Generates blank lines in a listing.

.OPT Controls generation of output files, listing and expansion of ASCII strings in .BYTE directives. In the MDT 650 the only useful parameters for the .OPT directive is GEN for the generation of ASCII strings in a .BYTE.

= = is the EQUATE directive and is used to reserve memory locations, reset the program counter, or assign a value to a symbol.

.END Signals the physical end of the program. When using the MDT 650 assembler, .END *must* be used as the last line of the TEXT. It is optional in the MOS Technology assembler.

User Comments. Begin with a semicolon ; .

Before assembly can take place on the MDT 650, the user must perform several "housekeeping" tasks using the following keys:

NUL Inserts null characters after each carriage return to allow time for the 'carriage' to return to the left side of the paper. TTY Model 33ASR requires no nulls; Model 38 requires at least one null. For other devices, the number of nulls may be determined experimentally by observing if, and when, extraneous characters appear on the "retrace." Enter number of nulls, and press CR .

TAB Allows the user to specify the format of the displayed and printed (black key) outputs. The options are:

ØØ Prints the source code exactly as entered by the user.
1Ø Prints and displays output in discrete columns.
Ø1 Permits the user to manually step through the assembly by depressing the SPACE bar.

SYM Initializes the symbol table. The user must enter *four (4)* hex characters corresponding to at least the maximum number of symbols in the source program. The MDT responds by displaying the number of symbols requested and an approximation of the number of lines of source code which this table should be able to accommodate.

SRC The MDT can be instructed to assemble from several sources and devices such as paper tape, cassette, etc. It is a dual pass assembler, so if external devices are used, instead of internal user side memory, the program must be "put through" twice.
If the program is to be assembled directly from user side memory, two options are available:

SRC M or SRC M TTY (if printed Teletype
output is desired)

ASB Assembles the source program and creates a new symbol table. The assembler assumes that the program is complete and that no reference is made to symbols which are not defined someplace.

ASO Assembles the source program, but does not create an entirely new symbol table. During the assembly process, each *new* label encountered will be added to the symbol table. However, all symbols previously defined will remain and can be referenced in the new program.

Table A.7 presents a detailed key by keystroke illustration of the above ASSEMBLE MODE keys as they apply to Example A1.

TABLE A.7 Suggested procedure to be used to ASSEMBLE the correct TEXT located in the user side memory of the MDT 650. The specific data applies to Example A1 as shown in Table A.6.

Press Keys on MDT Keyboard	*See Displayed*	*Comments*
	Almost anything	The TEXT is located in user side memory; it seems "correct"; you are ready to ASSEMBLE.
RSET RSET	MDT V1	With only one RSET additional miscellaneous characters *may* appear. Newer versions of the MDT assembler will require only one RSET.
NUL	NULLS = Ø1	Enter desired number of nulls and press CR. We are satisfied with 1 null, so just press CR.
CR	S (blinking)	
TAB (black key) 1 Ø CR	TAB,SQUISH = XX	Enter 1Ø for TAB yes, SQUISH no.
SYM (Do not use CR)	SYMBOLS?	MDT is asking the user for the number of SYMBOLS. User *must enter four (4)* hex characters.
Ø Ø Ø F	S ØØØF Ø22A LINES (S blinks)	Displays the number of symbols requested and an approximation of the number of lines of source code which this symbol table should be able to accommodate. ØØØF is a conservative estimate; we really have only 4 symbols.
Check that TTY switch is in LINE and punch is OFF.		
SRC M TTY	ON	Teletype prints ON; if Teletype prints OFF, then press TTY key again. If no Teletype printed output is desired, do not press TTY key even once.
ASB	MDT displays each line of output, line by line, as printed on the TTY.	Teletype prints PASS1 PASS2 The remainder of the Teletype printed output is shown in Tables A.8, A.9, and A.10.

Examples of the assembler output as printed on a Teletype are shown in Tables A.8, A.9, and A.10 for the three TAB,SQUISH options. The fourth option, TAB,SQUISH = 11 (TAB yes, SQUISH yes) is not possible, of course.

The reader will also note that an ERRORS and WARNINGS count is printed. There are 23 possible errors. Two of these (Symbol Table Overflow and Length Table Overflow) are assembly errors which cause the assembly to stop. The remaining errors are caused by invalid assembly coding and do not cause the assembly to stop. When errors occur, they are printed in the assembly listing and identified by the following numbers:

1. Undefined Symbol
2. Label Previously Defined
3. Illegal or Missing Op Code
4. Address Not Valid
5. Accumulator Mode Not Allowed
6. N/A (Not Available)
7. Ran Off End of Card
8. Label Doesn't Begin with Alphabetic Character
9. Label Greater Than Six Characters
10. Label or Op Code Contains Nonalphanumeric
11. Forward Reference in Equate
12. Invalid Index - Must be X or Y
13. Invalid Expression
14. Undefined Assembler Directive
15. Invalid Operand for Page Zero Mode
16. Invalid Operand for Absolute Mode
17. Relative Branch Out of Range
18. Illegal Operand Type for This Instruction
19. Out of Bounds on Indirect Addressing
20. A, X, Y, S, and P are Reserved Labels
21. Program Counter Negative - Reset to Ø
22. Symbol Table Overflow (Assembler stops!)
23. Length Table Overflow (Assembler stops!)

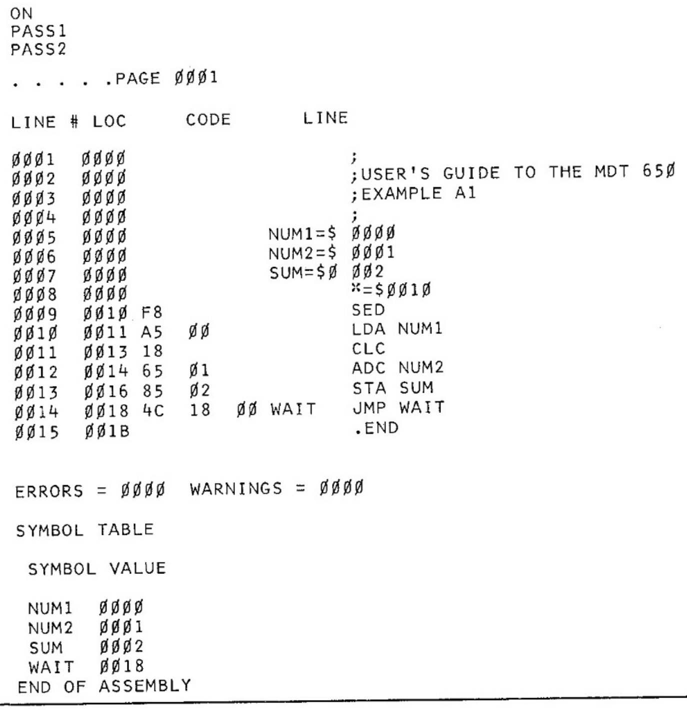
The assembler output also provides a SYMBOL TABLE which contains a listing of the symbols and corresponding values, but it is not sorted alphabetically, as in the case of the MOS Technology assembler. There is no option for cross-references, error files, or op code count.

It would be instructive for the reader at this point to compare the results of manual assembly for Example A1 as shown in Table A.1, with the results of MDT 650 assembly as shown in Table A.8, A.9, and A.10.

A.8 MDT 650 IN THE EXECUTE MODE

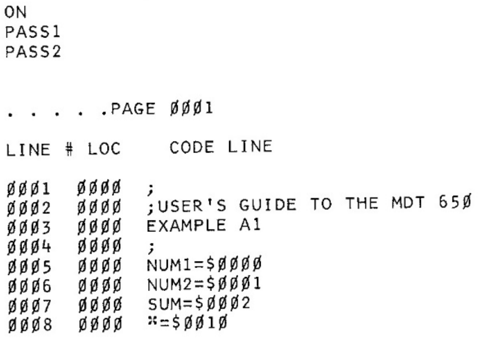
The purpose of the EXECUTE MODE is to provide the user with the means to execute the computer program instructions that modify data, registers, or memory. The entire program may be executed either at clock speed or the execution may be done one line of code at a time

TABLE A.8 Teletype printed output corresponding to Table A.6. Note that TAB,SQUISH = 1Ø (TAB yes, SQUISH no).



```
ON
PASS1
PASS2
. . . . .PAGE ØØØ1

LINE # LOC     CODE      LINE

ØØØ1  ØØØØ                              ;
ØØØ2  ØØØØ                              ;USER'S GUIDE TO THE MDT 65Ø
ØØØ3  ØØØØ                              ;EXAMPLE A1
ØØØ4  ØØØØ                              ;
ØØØ5  ØØØØ                     NUM1=$ ØØØØ
ØØØ6  ØØØØ                     NUM2=$ ØØØ1
ØØØ7  ØØØØ                     SUM=$Ø ØØ2
ØØØ8  ØØØØ                              *=$ØØ1Ø
ØØØ9  ØØ1Ø F8                           SED
ØØ1Ø  ØØ11 A5  ØØ                       LDA NUM1
ØØ11  ØØ13 18                           CLC
ØØ12  ØØ14 65  Ø1                       ADC NUM2
ØØ13  ØØ16 85  Ø2                       STA SUM
ØØ14  ØØ18 4C  18  ØØ WAIT              JMP WAIT
ØØ15  ØØ1B                              .END

ERRORS = ØØØØ  WARNINGS = ØØØØ

SYMBOL TABLE

 SYMBOL VALUE

 NUM1   ØØØØ
 NUM2   ØØØ1
 SUM    ØØØ2
 WAIT   ØØ18
END OF ASSEMBLY
```

TABLE A.9 Teletype printed output corresponding to Table A.6 *except* that TAB,SQUISH = Ø1 (TAB no, SQUISH yes)



```
ON
PASS1
PASS2

. . . . .PAGE ØØØ1

LINE # LOC    CODE LINE

ØØØ1  ØØØØ  ;
ØØØ2  ØØØØ  ;USER'S GUIDE TO THE MDT 65Ø
ØØØ3  ØØØØ  EXAMPLE A1
ØØØ4  ØØØØ  ;
ØØØ5  ØØØØ  NUM1=$ØØØØ
ØØØ6  ØØØØ  NUM2=$ØØØ1
ØØØ7  ØØØØ  SUM=$ØØØ2
ØØØ8  ØØØØ  *=$ØØ1Ø
```

```
ØØØ9  ØØ1Ø  F8  SED
ØØ1Ø  ØØ11  A5  ØØ  LDA  NUM1
ØØ11  ØØ13  18  CLC
ØØ12  ØØ14  65  Ø1  ADC  NUM2
ØØ13  ØØ16  85  Ø2  STA  SUM
ØØ14  ØØ18  4C  18  ØØ  WAIT JMP WAIT
ØØ15  ØØ1B  .END

ERRORS = ØØØØ WARNINGS = ØØØØ


SYMBOL TABLE

 SYMBOL VALUE

 NUM1  ØØØØ
 NUM2  ØØØ1
 SUM   ØØØ2
 WAIT  ØØ18
END OF ASSEMBLY
```

TABLE A.10 Teletype printed output corresponding to Table A.6 *except* that TAB,SQUISH = ØØ (Source code is printed as entered)

```
ON
PASS1
PASS2


. . . . .PAGE ØØØ1

LINE # LOC     CODE        LINE

ØØØ1  ØØØØ                ;
ØØØ2  ØØØØ                ;USER'S GUIDE TO THE MDT 65Ø
ØØØ3  ØØØØ                ;EXAMPLE A1
ØØØ4  ØØØØ                ;
ØØØ5  ØØØØ                NUM1=$ØØØØ
ØØØ6  ØØØØ                NUM2=$ØØØ1
ØØØ7  ØØØØ                SUM=$ØØØ2
ØØØ8  ØØØØ                *=$ØØ1Ø
ØØØ9  ØØ1Ø  F8            SED
ØØ1Ø  ØØ11  A5  ØØ        LDA NUM1
ØØ11  ØØ13  18            CLC
ØØ12  ØØ14  65  Ø1        ADC NUM2
ØØ13  ØØ16  85  Ø2        STA SUM
ØØ14  ØØ18  4C  18  ØØ    WAIT JMP WAIT
ØØ15  ØØ1B                .END

ERRORS = ØØØØ  WARNINGS = ØØØØ


SYMBOL TABLE

 SYMBOL VALUE

 NUM1  ØØØØ
 NUM2  ØØØ1
 SUM   ØØØ2
 WAIT  ØØ18
END OF ASSEMBLY
```

(called single-step). Single-step operation may be controlled manually, or it may be automatic with a variable, but user selectable, interval of time between the single-steps.

An additional feature, called TRACE, permits the optional display of the contents of all of the microprocessor registers immediately following the exeuction of each single-step.

Two hardware breakpoints may be implemented to stop the user microprocessor on any desired address when one of four user selectable conditions are satisfied.

The user controls the MDT 650 in the EXECUTE MODE by depressing the following keys:

$\boxed{\text{BRK}}$ Causes the user processor to stop on a desired address when one of four conditions are satisfied. Two break points are available. Immediately after the BRK key is depressed, the MDT displays

BREAKPOINTS=ØØØØ RN ØØØØ RN

The first two fields ØØØØ RN control Breakpoint 1 as follows:

field ØØØØ corresponds to the breakpoint address
field R corresponds to the break condition
field N allows the user to specify if the processor should stop when a breakpoint is encountered.

How to initialize Breakpoint 1. (Breakpoint 2 is similar.)

Press $\boxed{\text{BRK}}$ key

Enter 4 hex characters; if the control processor cannot identify them as a valid address, a ? will be displayed and the cursor will return to the high order position. Try again. When successful, cursor will move to the first R.

Enter one of the following to specify the condition which will initiate the breakpoint:

A specified address is encountered for any reason
I specified address is accessed for an op· code fetch (sync = high)
W user processor *writes* into the specified address
R user processor *reads* from the specified address

Cursor now moves to the first N position.

Enter one of the following:

Y user processor will stop when breakpoint is encountered (Yes)

N user processor will not stop but a pulse will be provided on the rear BNC connector each time the breakpoint is encountered. This pulse could, for example, be used to trigger an oscilloscope. (No)

DSP Format: DSP h h h h where h is a hex character.

Displays the contents of memory location hhhh, the three previous bytes, and the next four bytes. Brackets [ ] frame the current byte and the cursor is positioned at the first character of the current byte in preparation for changing its contents. To alter, type in two hex characters and the display will then be automatically updated one memory position in each position and the cursor will be returned to its original position with the brackets [ ].

GO (Must always be followed by a four (4) hex character address of the first instruction of the program to be executed.)

The control processor immediately places the 4 hex character address into FFFC and FFFD (reset vector location) of the user processor memory and then activates reset on the user processor along with all input/output devices. The user processor then executes the instruction loaded at the "GO" address and *stops*. This instruction is then disassembled and displayed. The remainder of the program can be executed by depressing the RUN key.

REG Displays the contents of all of the user microprocessor registers with this format:

*xxxx Axx Yxx Sxx NV BDIZC

The registers are in the order: Program Counter, Accumulator, X-register, Y-register, Stack Pointer, and Processor Status.

RUN Directs the user processor to execute the program starting at the address currently in the Program Counter register.

SI (Single-step Instruction) Executes user program one instruction at a time each time the SI key is depressed. This instruction is then disassembled by the control processor and displayed.

The SI key may be pressed whether the control processor is stopped or running. If it is stopped, a single instruction is executed. If running, the control processor "snap shots" the instruction being executed at the time, i.e., the current instruction will be disassembled and displayed. The user processor will continue running.

$\boxed{\text{SLO}}$ Causes the user processor to execute the user program one instruction at a time with an automatic advance to the next instruction. Immediately after the $\boxed{\text{SLO}}$ key is depressed, the MDT displays

DELAY COUNT=ØØ I & Y =

with the cursor located in the high order column of the DELAY COUNT field ØØ. Enter a delay count in the range of 3Ø to FØ. The lower number will cause a rapid execution (about 1 sec/step) and the higher number will cause a *very* slow execution (about 60 sec/step).

After the delay count is entered, a Y will appear to the right of the I & R field. If the user enters a Y (Yes), the MDT will print the contents of the user processor registers after each instruction execution. Entering a N (No) will direct the MDT to print only the instructions (disassembled) but not the register contents.

Warning: The slow run mode does not automatically provide for setting the program counter to the desired starting point.

$\boxed{\text{TTY}}$ (It is optional to also have the Teletype print all of the output while executing the program in either the single-step $\boxed{\text{SI}}$ or the slow $\boxed{\text{SLO}}$ modes.)

The $\boxed{\text{TTY}}$ key *must* be depressed before depressing the $\boxed{\text{SLO}}$ key. Execution speed will be determined by the time required by the Teletype to print the results, so it is recommended that a delay count of approximately 1Ø be entered.

The $\boxed{\text{TTY}}$ key is a "toggle switch" type, that is, press once for ON, press again for OFF, press again for ON, etc.

$\boxed{\text{STP}}$ "Stops" the user processor. This is one of two ways to stop the execution of programs in the user processor. The other way is to use breakpoints.

Detailed key by keystroke illustrations of the above keys in the EXECUTE MODE are provided in the following Tables:

TABLE A.11 Using the $\boxed{\text{RUN}}$ key. Also $\boxed{\text{DSP}}$, $\boxed{\text{M+1}}$, $\boxed{\text{M+8}}$, $\boxed{\text{REG}}$, and $\boxed{\text{STP}}$ keys.

TABLE A.11a Using the $\boxed{\text{GO}}$ key.

TABLE A.12 Using the $\boxed{\text{SI}}$ key.

TABLE A.13 Using the $\boxed{\text{SLO}}$ key. (With registers displayed)

TABLE A.14 Using the breakpoint $\boxed{\text{BRK}}$ key.

TABLE A.11 Suggested procedure for executing the program associated with Example A1 in the EXECUTE MODE using the $\boxed{\text{RUN}}$ key. We are adding 9 + 6 in the decimal mode.

Press Keys on MDT Keyboard	*See Displayed*	*Comments*
$\boxed{\text{RSET}}$	MDT V1	Source program for Example A1 has been assembled.
$\boxed{\text{DSP}}\boxed{Ø}\boxed{Ø}\boxed{1}\boxed{Ø}$	ØØ1Ø xx xx xx [F8] A5 ØØ 18 65	Contents of memory location ØØ1Ø are shown in [ ] See Sec. A.8 for more detail on the display format. At this point the reader should compare the contents
$\boxed{\text{M+1}}$	ØØ11 xx xx F8 [A5] ØØ 18 65 Ø1	of memory location ØØ1Ø through ØØ1A with Table
$\boxed{\text{M+8}}$	ØØ19 85 Ø2 4C [18] ØØ xx xx xx	A.8 and verify that the correct op codes are stored.
$\boxed{\text{CR}}$	S (blinking)	Terminates this display mode.
$\boxed{\text{DSP}}\boxed{Ø}\boxed{Ø}\boxed{Ø}\boxed{Ø}$	ØØØØ xx xx xx [xx] xx xx xx xx	Contents of memory location ØØØØ are shown in [ ].
$\boxed{Ø}\boxed{9}$	ØØØ1 xx xx Ø9 [xx] xx xx xx xx	Loaded Ø9 for NUM1 into memory location ØØØØ and display automatically incremented 1 memory location.
$\boxed{Ø}\boxed{6}$	ØØØ2 xx Ø9 Ø6 [xx] xx xx xx xx	Loaded Ø6 for NUM2 into memory location ØØØ1.
$\boxed{Ø}\boxed{Ø}$	ØØØ3 Ø9 Ø6 ØØ [xx] xx xx xx xx	Cleared the SUM memory location to zero.
$\boxed{\text{CR}}$	S (blinking)	Terminates this display mode.
$\boxed{\text{REG}}$	*xxxx Axx Xxx Yxx Sxx xxxxxx	Displays contents of Program Counter, Accumulator, X-register, Y-register, Stack Pointer, and Status
$\boxed{Ø}\boxed{Ø}\boxed{1}\boxed{Ø}$	*ØØ1Ø Axx Xxx Yxx Sxx xxxxxx	Register, in this order. We have just loaded the starting address ØØ1Ø of our program into the PC.
$\boxed{\text{CR}}$	S (blinking)	Terminates this register display mode.
$\boxed{\text{RUN}}$	R (blinking)	Example A1 program is executing.
$\boxed{\text{STP}}$	ØØ18 WAIT JMP WAIT	Displays last instruction executed.
$\boxed{\text{CR}}$	(S and ? blinking alternately)	Terminates this display mode.
$\boxed{\text{REG}}$	*ØØ18 A15 Xxx Yxx Sxx xxxDxx	As expected: Program Counter = ØØ18 and the contents of the Accumulator are 15 (SUM = 9+6 = 15). D indicates we are in the decimal mode.
$\boxed{\text{CR}}$		
$\boxed{\text{DSP}}\boxed{Ø}\boxed{Ø}\boxed{Ø}\boxed{Ø}$ $\boxed{\text{CR}}$	ØØØØ xx xx xx [Ø9] Ø6 15 xx xx	Contents of NUM1, NUM2, and SUM are displayed corresponding to memory locations ØØØØ, ØØØ1, and ØØØ2.

TABLE A.11a Procedure for executing the program associated with Example A.1 in the EXECUTE MODE using the GO key.

Press Keys on MDT Keyboard	*See Displayed on MDT*	*Comments*
	With the memory allocation shown in Figure A.1, the GO key can not be used because the control processor immediately places the 4 hex characters into user side memory locations FFFC and FFFD. (Figure A.1 shows FFFC and FFFD in user memory space where no actual hardware RAM memory was allocated.) The user processor then executes the instruction loaded in the GO address and "stops." The instruction is then disassembled and displayed. The remainder of the program can be executed by depressing the RUN key as in Table A.11. To demonstrate this would require a relocation of memory shown in Figure A.1 and will not be done here because of the similarity with just using the REG and RUN keys.	

TABLE A.12 Suggested procedure for executing the program associated with Example A1 in the EXECUTE MODE using the SI key. Again, we are adding 9 + 6 in the decimal mode. To avoid duplication, we will enter Table A.11 at the CR line just above the RUN key line and use the SI key instead.

Press Keys on MDT Keyboard	*See Displayed on MDT (Teletype printed output, optional)*			*Comments*
CR	S (blinking)	TOP PORTION OF TABLE A.11		Terminates current display mode.
~~RUN~~	~~R (blinking)~~			~~Example A1 program is executing.~~
				⋮
SI	ØØ1Ø.	SED		First line of code was executed, disassembled, and displayed. Refer to Table A.8.
SI	ØØ11.	LDA NUM1	Ø9	Same for the second line of code.
SI	ØØ13.	CLC		Same for the third line of code.
SI	ØØ14.	ADC NUM2	Ø6	Same for the fourth line of code.
SI	ØØ16.	STA SUM	15	Same for the fifth line of code.
SI	ØØ18. WAIT	JMP WAIT		Same for the sixth line of code.
SI	ØØ18. WAIT	JMP WAIT		We are now in the jump to itself loop and will continue to do this each time SI is pressed.
SI	ØØ18. WAIT	JMP WAIT		

SI	ØØ18. WAIT JMP WAIT	
CR	(S and ? blinking alternately)	
REG	*ØØ18 A15 Xxx .Yxx Sxx xxxDxx	As expected, identical to the corresponding line in Table A.11.
CR		
DSP	ØØØØ xx xx xx [Ø9] Ø6 15 xx xx	Again, identical to the corresponding line in Table A.11.
CR		Terminates current display mode.

TABLE A.13 Suggested procedure for executing the program associated with Example A1 in the EXECUTE MODE using the SLO key. Again, we are adding 9 + 6 in the decimal mode. To avoid duplication, we will enter Table A.11 at the CR line just above the RUN key and use the SLO key instead of the RUN key.

Press Keys on MDT Keyboard	*See Displayed on MDT (Teletype printed output, optional)*	*Comments*
	TOP PORTION OF TABLE A.11	
CR	S (blinking)	Terminates current display mode.
~~RUN~~	~~R (blinking)~~	~~Example A1 program is executing.~~
		⋮
SLO	DELAY COUNT=ØØ I&R=	Enter DELAY COUNT in the range of 3Ø·to FØ. 3Ø provides about 1 sec./step; FØ provides about 30 sec./step.
3 Ø	DELAY COUNT=3Ø I&R=Y (Y blinks)	We wish to display registers; so type Y for YES.
Y	ØØ1Ø. SED	Be patient; wait for 1 to 30 seconds.
	*ØØ11 AØØ Xxx Yxx Sxx xx xDxxx	Accumulator contents are ØØ; D implies decimal mode. Now, new lines will appear in the display at the rate of approximately 1 sec./line.
	ØØ11. LDA NUM1 Ø9	
	*ØØ13 AØ9 Xxx Yxx Sxx xx xDxxx	Accumulator loaded with Ø9.
	ØØ13. CLC	
	*ØØ14 AØ9 Xxx Yxx Sxx xx xDxxx	Carry flag was not set previously so effect of CLC carry is not apparent.
	ØØ14. ADC NUM2 Ø6	
	*ØØ16 A15 Xxx Yxx Sxx xx xDxxx	Accumulator contents are 15, the sum of 9 + 6.
	ØØ16. STA SUM 15	
	*ØØ18 A15 Xxx Yxx Sxx xx xDxxx	Accumulator contents remain 15.
	ØØ18.WAIT JMP WAIT	
	*ØØ18 A15 Xxx Yxx Sxx xx xDxxx	
	ØØ18.WAIT JMP WAIT	We are now quite obviously in the jump to itself loop.
	*ØØ18 A15 Xxx Yxx Sxx xx xDxxx	
STP	ØØ18.WAIT JMP WAIT	Display remains constant.

TABLE A.14 Suggested procedure for adding a breakpoint to Table A.13 so that when a memory location ØØ18 is encountered for any reason, the µser processor will "stop."

Press Keys on MDT Keyboard	*See Displayed on MDT (Teletype printed output optional)*	*Comments*
	⋮	
$\boxed{\text{BRK}}$	BREAKPOINTS=ØØØØ RN ØØØØ RN	Breakpoints may be set anytime prior to execution with the $\boxed{\text{SLO}}$, $\boxed{\text{SI}}$ or $\boxed{\text{RUN}}$ keys.
$\boxed{\text{Ø}}\boxed{\text{Ø}}\boxed{1}\boxed{8}$	BREAKPOINTS=ØØ18 AN ØØØØ RN	R blinks asking for condition A, I, W, or R? Enter A for the "any reason" condition.
$\boxed{\text{A}}$	BREAKPOINTS=ØØ18 AN ØØØØ RN	N blinks asking for Yes or No. Enter $\boxed{\text{Y}}$ for Yes.
$\boxed{\text{CR}}$		Terminates breakpoint entry mode. We did not wish to activate the second breakpoint.
$\boxed{\text{TTY}}$	ON	Activates the Teletype.
	⋮	
$\boxed{\text{SLO}}$	DELAY COUNT=ØØ I&R=	Enter DELAY COUNT in the range of 3Ø to FØ. 3Ø provides about 1 sec./step; FØ provides about 60 sec./step.
$\boxed{3}\boxed{\text{Ø}}$	DELAY COUNT=3Ø I&R=Y (Y blinks)	We wish to display registers; so type $\boxed{\text{Y}}$ for Yes.
$\boxed{\text{Y}}$	ØØ1Ø. SED	Be patient; wait for 1 to 30 seconds.
	*ØØ11 AØØ Xxx Yxx Sxx xx xDxxx	Accumulator contents are ØØ; D implies decimal; Now, new lines will appear in the display at the rate of approximately 1 sec./line.
	ØØ11. LDA NUM1 Ø9	
	*ØØ13 AØ9 Xxx Yxx Sxx xx xDxxx	Accumulator loaded with Ø9.
	ØØ13. CLC	
	*ØØ14 AØ9 Xxx Yxx Sxx xx xDxxx	Carry flag was not set previously so effect of CLC carry is not apparent.
	ØØ14. ADC NUM2 Ø6	
	*ØØ16. A15 Xxx Yxx Sxx xx xDxxx	Accumulator contents are 15, the sum of 9 + 6.
	ØØ16. STA SUM 15	
	*ØØ18 A15 Xxx Yxx Sxx xx xDxxx	Accumulator contents remain 15.
	ØØ18.WAIT JMP WAIT	User processor "stopped" by the breakpoint.

A.9 MDT 650 IN THE OUTPUT MODE

The MDT 650 is said to be in the OUTPUT MODE when peripheral devices, such as a Teletype or a Cathode Ray Tube (CRT) Terminal, are connected and activated. The primary concern here will be a Teletype with a paper tape punch/reader unit. The punched paper tape copy of the TEXT may be reentered into the MDT at some later time for reassembly and execution. The punched paper tape copy of the object code may be used to program a PROM or be read into a microcomputer memory, such as the KIM-1 microcomputer. It is also possible to obtain a printed copy of the output during single-step and slow modes of execution, but because of the slow printing speed of the Teletype, this is not done very often.

The punched paper tape copy of the object code cannot be reentered into the MDT, but this is not a disadvantage because the TEXT may be assembled in just a few seconds (provided the Teletype is not energized).

The user controls the MDT 650 by depressing the following keys:

DMP Directs the MDT to output the object code with a format to include the number of bytes per record, the number of records, and the checksums. Immediately after the DMP key is depressed, the MDT displays

DUMP SSSS EEEE

The SSSS refers to the starting address of the object code dump. The EEEE refers to the ending address of the object code dump. The Teletype is automatically energized.

LST Directs the MDT to output the TEXT (including comments). The Teletype is automatically energized.

Warning: Depress the TOP key in the EDIT MODE if a copy of the entire TEXT is desired.

Detailed key by keystroke illustrations of the use of the above keys in the OUTPUT MODE are provided in the following Tables:

TABLES A.15 and A.15a	Using the LST key to output a printed and a punched paper tape copy of the TEXT.
TABLE A.16	Using the DMP key to output a printed and a punched paper tape copy of the object code.

TABLE A.15 Suggested procedure to punch source code (TEXT including comments) on a paper tape. This paper tape may be read back into the MDT 650 via a Teletype which has a paper tape reader unit.

Press Keys on MDT Keyboard	*See Displayed*	*Comments*
With the TTY switch in LOCAL prepare a paper tape leader.[1]	Almost anything	The TEXT resides in the MDT 650 memory need not have taken place.
Rotate the TTY switch to LINE; depress the ON punch button.		
RSET RTE		Reset and return to TEXT editor mode.
TOP		Output is to begin at the top of TEXT.

LST TTY[2]	MDT displays each line of output, line by line, as printed on the TTY. The last line of output remains on the MDT display.	The TTY is automatically energized when the TTY key is depressed.
Rotate the TTY switch to LOCAL. Punch about 6 rubouts to protect the end of the paper tape.		Both the actual TTY punched paper tape and the TTY printed outputs are shown in Table A.15a.

Footnotes: [1] It is suggested that a suitable paper tape leader be prepared with the usual warnings about extraneous characters (including nonprint characters, such as line feed or carriage returns) appearing at the beginning of the tape. Even a blank leader (only sprocket holes) is not acceptable and is usually indicated by a series of @@@@@s displayed on the MDT display. For the inexperienced user, the best procedure is to punch a number of rubouts.

[2] LST K is another option.

TABLE A.15a Teletype printed output and punch paper tape generated by TABLE A.15 for the TEXT of Example A1.

```
;
;USER'S GUIDE TO THE MDT 65Ø
;EXAMPLE A1
;
NUM1=$ØØØØ
NUM2=$ØØØ1
SUM=$ØØØ2
*=$ØØ1Ø
SED
LDA NUM1
CLC
ADC NUM2
STA SUM
WAIT JMP WAIT
.END
```

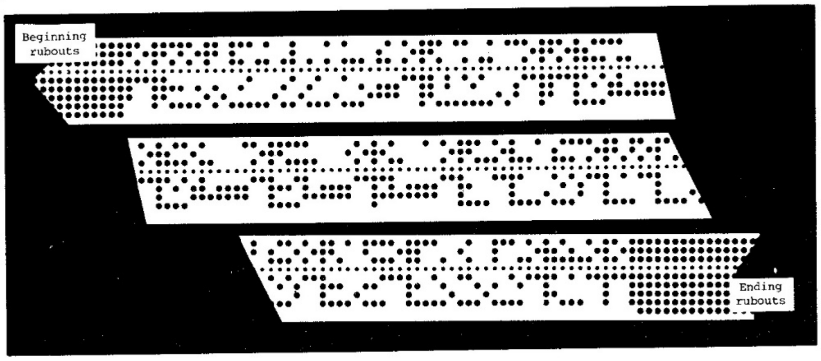
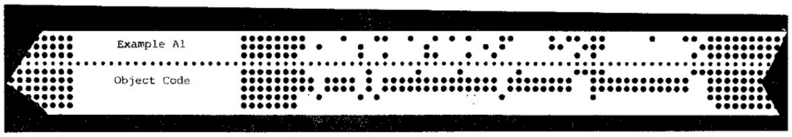


TABLE A.16 Procedure to punch object (machine) code on a paper tape. This paper tape could be used to program a PROM or it could be entered into a microcomputer memory, such as the KIM-1 microcomputer, using a Teletype paper tape reader.

Press Keys on MDT Keyboard	*See Displayed*	*Comments*
With the TTY switch in LOCAL prepare a paper tape leader. Rotate the TTY switch to LINE; depress the ON punch button.	Almost anything	The TEXT has been assembled. Execution need not have taken place.
RSET	MDT V1	
DMP	DUMP SSSS EEEE	The MDT is asking for the starting and ending addresses of the object code which the user wishes to punch on paper tape.
Ø Ø 1 Ø Ø Ø 1 B	MDT displays each line of output, line by line, as printed on the Teletype. The last line of output remains on the MDT display.	The starting and ending addresses may be obtained from any of the assembled outputs, such as TABLE A.8. The Teletype is automatically energized when the last character of SSSS EEEE is entered.

Teletype Printed Output

```
 ;ØBØØ1ØF8A5ØØ1865Ø185Ø24C18ØØØ321
;ØØØØØ1ØØØ1
```



A.10 IN CASE OF DIFFICULTY!

TABLE A.17 A partial list of difficulties which have been noted and may be of some assistance to the user of the MDT 650

SYMPTOMS	*WHAT TO DO*
TEXT appears to be read in normally from the Teletype but in the EDIT MODE can not be found and appears not to have been stored.	Locate and *&$#@! the last user who relocated the two 4K RAMs from the locations shown on the memory map shown in Figure A.1 and did not tag the MDT to notify other users.

	Each 4K RAM board has a 4 section DIP switch. One board should have all 4 sections *not* OPEN; the other board should have A - OPEN and B, C, D *not* OPEN.
The RSET key on the keyboard is depressed but the display remains blank.	Lift the cover of the MDT and press the super reset *red* button. The black button next to it is essentially the same as the RSET key on the keyboard. Press both, if you wish.
During the initial portion of the LOAD MODE from the Teletype a series of @@@@s are displayed.	An illegal character is punched on the paper tape leader. The most typical error is a nonprinting character such as a carriage return or line feed. Verify the tape contents with the Teletype in LOCAL and print the contents of the paper tape on the Teletype. Do not read sprocket holes.
During the LOAD MODE from Teletype, characters on the MDT display do not correspond with those on the tape, and many are even nonASCII.	Remove the cover of the MDT and note the red *toggle* switch on the top of the rightmost board. For a Teletype, it should be in the forward position (110 baud) and not toward the rear (the 300 baud position).