DECUS
PROGRAM LIBRARY

DECUS NO.	8-527
TITLE	XDDT8E
AUTHOR	Kincade N. Webb
COMPANY	Xenex Corporation Waltham, Massachusetts
DATE	March 4, 1972
SOURCE LANGUAGE	PAL 10

XDDT8E

DECUS Program Library Write-up DECUS NO. 8-527

XDDT8-E is an octal-symbolic debugging program for a PDP8-E with Extended Memory which preserves the status of the program interrupt system at breakpoints. It is the result of updating the old XDDT (Decus 8-127) debugging program to make it operate correctly on the PDP8-E.

The program occupies locations 4214 through 7577 of any memory field. In addition, XDDT8-E has a symbol table which extends from 4213 down towards 0000, where up to six-character symbols are stored using four locations per symbol. XDDT8-E distinguishes among the following four types of symbols: Memory Reference Instructions, Operate-IOT instructions, Location Names, and DEC-338 Commands. An initial symbol table including the basic PDP-8 instructions and extended memory instructions is provided, thus the highest location initially available to the user is 3723 in the field where XDDT8-e is running.

From the Teletype, the user can symbolically examine and modify the contents of any memory location in a variety of formats. Positive and negative block searches with a mask may also be performed.

XDDT8-E includes an elaborate single-breakpoint facility to help the user run sections of his program. When this facility is used, the debugger also uses locations 0006 and 0007 of every memory field.

XDDT8-E allows for RIM and BIN punching of core on either the ASR or high-speed punch.

XDDT8-E is not compatible with PDP8-L or 8-I computers.

"CTRL" Characters

In this writeup, ↑ directly preceding an alphabetic character, such as ↑E, indicates a special character which the user types by holding down the special shift key labelled "CTRL" while striking the corresponding alphabetic character key. When a user inputs one of these special characters, XDDT echos an up-arrow followed by the corresponding alphabetic character.

Loading, Starting, Restarting

XDDT8-E may be loaded into any memory field; it occupies locations 3724 through 7577. This includes the initial symbol table.

The STARTING ADDRESS of XDDT is 7000.

During the operation of XDDT8-E or during the execution of a user's program, XDDT may be restarted at 7000. When XDDT is restarted at a time when it was already in control, none of the registers or indicators associated with program status at breakpoints are affected.

When XDDT8-E is started or restarted, the setting of the Data Field is ignored.

Returning to the Monitor

For use with PDP-8 operating systems, typing ↑C causes XDDT to return to the resident monitor by transferring control to location 07600 of the PDP-8.

Constituents of Octal-Symbolic Expressions

OCTAL DIGIT is one of the characters 0, 1, ..., 7

OCTAL NUMBER is one to six octal digits, evaluated modulo $(10000)_8$

SYMBOL is one to six characters chosen from 0, 1, ..., 9, A, B, ..., Z and ".", but not an octal number.

TERM is SYMBOL or OCTAL NUMBER or * or #

EXPRESSION is one or more terms separated by either +, -, or SPACE

+ is two's complement plus

- is unary or binary two's complement minus

SPACE is inclusive OR, or separator between memory reference instruction and address

* means indirect addressing when immediately following an addressable instruction; when alone, it has the value of the "current location".

when alone, has the value of the expression last typed in or out. (Note that # is also used to set the field of attention.)

Type-in BCD - Typing a) following a symbol or octal number causes the value of that term to be taken as the trimmed BCD codes of the first two characters of the term. If the term is a single character, it is assembled in bits 0-5, and bits 6-11 are set to 00.

Null Characters - XDDT ignores characters whose code is 000. This corresponds to blank tape.

Illegal Characters - XDDT responds with a BELL, and the character is ignored.

Undefined Symbols - XDDT responds with a ? , and the expression is ignored.

RUB OUT or ? - XDDT responds with a ? , and the expression is ignored.

Symbols

XDDT distinguishes among four types of symbols:

Memory Reference Instructions

The six PDP-8 memory reference instructions, AND, TAD, ISZ, DCA, JMS, JMP, are permanently defined within XDDT. These symbols are typed out by XDDT only when examining or evaluating as an instruction.

Operate-IOT Instructions

These symbols are normally those PDP-8 instructions whose value is between 6000 and 7777. The initial table of XDDT includes the most common IOT, OPR symbols (see the list in the Appendix). Any number of additional Operate-IOT symbols may be defined through user control. These symbols are typed out by XDDT only when examining or evaluating as an instruction. If an exact match occurs, the symbol is typed out. Otherwise, the 12-bit instruction is typed. All valid micro instruction combinations are typed out for groups 1 and 2.

Location Names

These symbols are used to refer to 12-bit addresses within the PDP-8. The initial table of XDDT contains no symbols of this type, but any number of them may be defined through user control. These symbols are typed out by XDDT when examining or evaluating as a name, or as the address of an addressable instruction when examining or evaluating as an instruction, or as the location name when XDDT is opening locations. When typing a location name XDDT always type out a name symbol if there is an exact match. If there is not an exact match, XDDT normally types a relative address (nearest symbol + an octal offset). There is an option in XDDT to type out name symbols only on an exact match. A name symbol is the only type of symbol which can be removed from the symbol table.

DEC-338 Commands

These symbols are distinguished from the other three types for the purpose of selective type-out. This classification is provided especially for the symbols used for display list commands of the DEC-338 Programmed Buffered Display. A user may wish to define his own class of symbols for special applications. The initial table of XDDT contains no symbols of this type, but any number of them may be defined through user control. These symbols are typed out by XDDT only when examining or evaluating as a DEC-338 command.

Field of Attention

At any particular time, the attention of XDDT is directed towards one memory field, and all specified locations refer to that field.

When XDDT is waiting for type-in, the Data Field lights on the PDP-8 console indicate the current field of attention.

When XDDT is first loaded, its field of attention is 0.

The user may change the field of attention by typing an expression specifying the field number followed by # . When this is done, XDDT sets its lower limit used for zeroing and searching to 0000. If the specified field is the same one as the field in which XDDT is running, the upper limit is set to the highest available location which is not being used for the XDDT sumbol table; otherwise, the upper limit is set to 7577.

Bits 6-8 location 5400 within XDDT determine the highest memory field with which XDDT is operating. The supplied binary tape of XDDT has 0010 in this location, so that an 8-K memory is assumed. This number may be changed in order to use XDDT on a PDP-8 with more than two memory fields. The contents of location 5400 is used when XDDT is running a user's program with a breakpoint assigned and also when a field specification is typed.

Examinations

One of the following characters typed after an expression (typed in or out) causes XDDT to open the location specified by the expression, and type out the contents of the location as below. If no location was already open, the current mode of examination is set to the mode associated with the character, and the value of the current location (*) is set to the value of the opened location.

Character	Mode of Examination
/	Instruction
:	Name Symbol
\ (Shift L)	DEC-338 Command
[(Shift K)	Octal number
] (Shift M)	BCD

Equivalences

One of the following characters typed after an expression (typed in or out) causes XDDT to type that expression in the mode as below.

Character	Mode of Equivalence
←	Instruction
$	Name Symbol
&	DEC-338 CommAND
=	Octal number
%	BCD

Current Location

* has the value of the location which was last opened when no location was already open. Note that the value of * is updated as locations are opened during searches.

Carriage Return

If a location is open, an expression typed in followed by a CR causes XDDT to set the contents of that location to the value of the expression.

Miscellaneous

LINEFEED	First acts like a CR and then opens location *+1 in the current mode of examination.
↑ (shift N)	First acts like CR and then opens location *-1 in the current mode of examination.
(	If a location is open, an expression typed in followed by a (causes XDDT to set the contents of that location to the value of the expression and then immediately examine in octal the location addressed by the typed expression.

Limits

XDDT has a lower limit and upper limit used for zeroing, punching, and searching. The user may find out the value of either of these quantities:

Typing ↑L alone causes XDDT to type out in octal the value of the Lower limit.

Typing ↑U alone causes XDDT to type out in octal the value of the Upper limit.

The user may set either of the limits:

Typing an expression followed by ↑L causes XDDT to set the Lower limit to the value of the expression.

Typing an expression followed by ↑U causes XDDT to set the Upper limit to the value of the expression.

The limits are set by XDDT when a user specifies a field of attention (by #). The lower limit is set to 0000. If the specified field is the same one as the field in which XDDT is running, the upper limit is set to the highest available location which is not being used for the XDDT symbol table; otherwise, the upper limit is set to 7577. (For special applications, the user may wish to alter this default upper limit, which is stored within XDDT at location 5125.)

XDDT may also change the upper limit when a user defines a symbol.

Typing ↑H causes XDDT to type out in octal the Highest available location which is not being used for the XDDT symbol table.

Typing Mode for Locations

Locations typed out by XDDT prior to its typing one of the examination characters are normally typed as name symbols. The user is provided with an option to change this mode of type out:

Typing ↑O causes XDDT to type locations in Octal.

Typing ↑S causes XDDT to type locations as name Symbols.

Zeroing

Typing an expression followed by ↑Z causes XDDT to set the contents of the locations between the lower and upper limits inclusive to the value of the expression. If no expression is typed preceding the ↑Z, a value of 0 is assumed.

Positive (Negative) Searching

This command to XDDT is given by typing an expression followed by > (<). When this format is used an assumed MASK of 7777 is used for the search. The user may, however, specify a MASK as part of the command by first typing an expression followed by a ; as a separator.

Typing an expression followed by >(<) causes XDDT to search locations between the lower and upper limits inclusive. All locations whose contents do (not) match the value of the expression in the bits specified by the MASK are typed out. During the course of a search, the user may stop the search by striking any key.

Name Symbol Option

This option controls the typeout of name symbols by XDDT when examining or evaluating as a name, or as the address of an addressable instruction when examining or evaluating as an instruction, or as the location name when XDDT is opening a location. XDDT normally types out a name symbol plus an octal number when there is not an exact match.

Typing ↑E causes XDDT to type name symbols only when an Exact match occurs. Octal numbers are typed when there is not an exact match.

Typing ↑R causes XDDT to type name symbols as Relative addresses when necessary, i.e., as a name symbol plus an octal number when there is not an exact match.

Symbol Definition

The table of six PDP-8 memory reference instructions is fixed, but the user may define symbols of the other three types. The definition of a symbol always causes the length of the XDDT symbol table to increase by four PDP-8 locations, thereby decreasing the available space for a user program in the field where XDDT is running. If the field in which XDDT is running is the same as the field of attention, XDDT will adjust the upper limit to not exceed the highest available location which is not being used for the XDDT symbol table. The upper limit will then be lowered as each symbol is defined.

Operate-IOT Instructions

Typing an expression followed by ; followed by a symbol and terminated by ↑P causes XDDT to define the symbol to have an associated value equal to the value of the expression. The symbol is defined as an OPerate-IOT Instruction.

Example: SZA CLA; SZACLA↑P

Name Symbol

Typing an expression followed by ; followed by a symbol and terminated by ↑N causes XDDT to define the symbol to have an associated value equal to the value of the expression. The symbol is defined as a Name symbol.

Example: 200;START↑N

An alternative method is available for defining a name symbol: typing a symbol followed by , causes XDDT to define that symbol as a name symbol with an associated value equal to the value of * .

DEC-338 Command

Typing an expression followed by ; followed by a symbol and terminated by ↑A causes XDDT to define the symbol to have an associated value equal to the value of the expression. The symbol is defined as a DEC-338 CommAnd.

Example: 3000;POP↑A

Kill Name Symbol

Typing a name symbol followed by ↑K causes XDDT to Kill the definition of that symbol. The symbol table remains the same length since the table entry is merely cleared.

Typing ↑K alone causes XDDT to kill the definition of all current name symbols. The highest available location available for the user in the field where XDDT is running is then appropriately adjusted. However, the upper limit is not changed.

Reading a Symbol Table into XDDT

Two special characters are recognized by XDDT for the purpose of reading symbol table tapes. A symbol table tape begins with blank leader followed by the special character ctrl shift N, which causes XDDT to cease typing out while a symbol table tape is being read. The body of the tape is signalled by the special character ctrl shift O, followed by blank trailer.

When the end of a symbol table tape is recognized, XDDT stops reading tape. It then waits for the user to turn off the paper tape reader. In order to continue, the user must then strike any key. XDDT responds by typing out in octal the highest available location which is not being used for the XDDT symbol table.

Punching a Symbol Table

After a user has been using XDDT for a while, he may wish to punch a binary tape of the section of memory containing the user-defined symbols in order to save these definitions. Such a binary tape could then be reloaded for use with XDDT at a later date. Before punching, the user should use the ↑H command to determine the highest available location which is not being used for the XDDT symbol table. Let x be the number which XDDT types out. The locations to be punched are:

x+1 through 3723, and

6400 through 6402.

Locations 6400 through 6402 contain pointers to three of the subtables of the XDDT symbol table.

Adjusting the XDDT Symbol Table

The user may restore the XDDT symbol table to its initial state by setting the contents of locations 6400, 6401, and 6402 to 3724. The upper limit is not affected by these changes.

The user may remove all symbols from the XDDT table except for the eight basic PDP-8 instructions in order to have a maximum amount of available space for his program. This is accomplished by setting the contents of locations 6400, 6401, and 6402 to 4214. DO NOT then use mneumonic accessing.

Reading a Binary Tape

In order to read a binary tape on the Teletype reader, place the tape in reader in the leader area and START the reader. XDDT immediately transfers control to PDP-8 location 17777, where there is presumably some type of binary loader which then reads in the tape. After reading is finished, XDDT may be restarted at its starting address.

Program Execution and Breakpoints

XDDT aids the user in program debugging by giving him an elaborate single breakpoint facility. The user may command XDDT to assign a breakpoint at any location of any field. This assignment has no immediate effect, but is important when the user commands XDDT to start running his program. When he does this, XDDT saves the contents of the location where the breakpoint was assigned and replaces it by a special breakpoint instruction (5006). XDDT also plants special instructions into locations 0006 and 0007 of every memory field (without saving their contents). It then appropriately sets the Accumulator, Link, Data Field, MQ, Teletype output flag and program interrupt status and transfers control to the user's program.

The user's program will then execute normally unless control passes to the breakpoint location. In this event, a "breakpoint trap" occurs, which consists of a jump back to XDDT. When XDDT is re-entered the location of the trap, Accumulator, Link, MQ, Data Field, Teletype output flag, and program interrupt status are all saved. At this time XDDT types out the location of the breakpoint trap (including the field) followed by a right parenthesis followed by the values of the saved Link and Accumulator.

The user may then use the facilities of XDDT for examination and modification of any memory location. He may also remove the breakpoint assignment or reassign it to some other location. There are special facilities for examination and modification of those flags and registers saved at a breakpoint.

When XDDT is restarted at a time when it was already in control, none of the registers or indicators associated with program status at breakpoints are affected.

If the user wishes to resume his program where it left off, he may command XDDT to "proceed". The proceed command causes XDDT to restore the saved status, perform the instruction which is at the location of the last breakpoint trap, and to continue to run the user's program.

If a user's program doesn't return to XDDT through a breakpoint trap, the user may restart XDDT at its starting address. When this is done, the contents of the breakpoint location are restored and the saved Accumulator, Link, and Teletype flag are all cleared. XDDT will not allow the user to perform a proceed command until after the next breakpoint trap.

The breakpoint instruction must not be inserted into a user's program as an instruction by location modification.

Running a Program

Typing an expression followed by ' causes XDDT to:

1) if a breakpoint is assigned, set it up and also set up locations 0006 and 0007 of every memory field.

2) set the Data Field to the field of attention.

3) have program interrupt off.

4) restore the saved Accumulator, MQ, Link, and Teletype output flag.

5) Jump to the location (in the field of attention) specified by the value of the expression preceding the ' .

Breakpoint Assignment

Typing an expression followed by " causes XDDT to assign a breakpoint at the location (in the current field of attention) specified by the value of the expression. If no expression precedes the " , any breakpoint assignment is removed. Only one breakpoint may be assigned at a time, but it may be changed, even before proceeding back to the user's program.

A breakpoint should not be assigned at a location which is either modified or whose contents are used as data. Otherwise, there are no restrictions on breakpoint placement, as far as the breakpoint trap occurring. There are, however, three restrictions on the breakpoint assignment if the user wishes to proceed back to his program:

1) A breakpoint must not be assigned at any CIF instruction, nor at any instruction which follows a CIF instruction until after the next JMP or JMS instruction. More precisely, the restriction exists at locations where the contents of the Instruction Field register differ from the contents of the Instruction Buffer register.

2) A breakpoint must not be assigned at a location where the interrupt system could be on and where the programs depends upon the preservation of the contents of the Save Field register.

3) A breakpoint must not be assigned at any of the following EAE instructions: MUY, DVI, SHL, ASR, LSR.

Proceeding

Typing a ! causes XDDT to:

1) if a breakpoint is assigned, set it up and also set up locations 0006 and 0007 of every memory field.

2) restore the saved Accumulator, MQ, Link, Data Field, Teletype output flag, and program interrupt status.

3) perform the instruction which is at the location of the last breakpoint trap, and proceed from there.

If an expression is typed preceding the ! XDDT generates automatic multiple proceeds until it has been re-entered for the number of times equal to the value of the expression. Then, a normal breakpoint trap occurs, allowing the user to use the facilities of XDDT.

Program Interrupt

When a breakpoint trap occurs, XDDT determines the status of the program interrupt system and then the interrupt system is kept off during XDDT operations.

When a breakpoint trap occurs, further program interrupts are disabled within a couple of memory cycles by the execution of a CIF instruction. Thus if a user wishes to use XDDT on a 4-K PDP-8 to debug a program using the interrupt system, he must have an interrupt service routine which works.

If XDDT is being used to debug a program occupying only memory field zero and using the program interrupt system, and if XDDT is in another memory field, it is advisable to include an RMF instruction in the interrupt service routine. The chances of an interrupt occurring during one of the critical two machine cycles are rather small. However, if the user performs multiple proceeds, then there are many cycles during which the interrupt system is not disabled while control is within XDDT. This necessitates a working interrupt service routine which includes an RMF instruction and preserves the Accumulator and Link.

Saved Program Status

Whenever XDDT is in control, the user may examine and modify the flags and registers which are saved at breakpoints.

Link and Accumulator

At the time of a breakpoint trap, XDDT types out the values of the saved Link and Accumulator. Typing @ alone also causes XDDT to type out the saved Link and Accumulator.

Typing an expression followed by @ causes XDDT to set the value of the saved Accumulator to the value of the expression.

Typing an expression whose value is (non-) zero followed by ; @ causes XDDT to (set) clear the saved Link.

Typing an expression whose value is (non-) zero followed by ; followed by a second expression followed by @ causes XDDT to both (set) clear the saved Link and also set the value of the saved Accumulator to the value of the second expression.

MQ

The MQ is saved and restored at breakpoint and go time. Typing ↑Q above causes XDDT to type out the octal value of the saved MQ. Typing an expression followed by ↑Q causes XDDT to set the saved MQ to the value of the expression.

Data Field

Typing ↑D alone causes XDDT to type out in octal the value of the saved Data Field.

Typing an expression followed by ↑D causes XDDT to set the value of the saved Data Field to the value of the expression, which cannot be greater than 7.

Teletype Output Flag

Typing ↑T alone causes XDDT to type out the status of the saved Teletype output flag. "0" indicates a clear flag, and "1" (or non-zero) indicates a set flag.

Typing an expression whose value is (non-) zero followed by ↑T causes XDDT to (set) clear the saved Teletype output flag.

Interrupt Status

Typing ↑I alone causes XDDT to type out the saved interrupt status. "0" indicates Interrupt is off, and "1" (or non-zero) indicates interrupt is on.

Typing an expression whose value is (non-) zero followed by ↑I causes XDDT to turn (on) off the saved Interrupt status.

Using XDDT with the SEETXT Subroutine

The SEETXT subroutine may be used with XDDT on a DEC-338 in order to display output on the screen rather than (or in addition to) printing on the Teletype. If SEETXT is occupying memory field n then the following patch to XDDT can be made through the switches. Do not attempt to use XDDT to modify itself in this area.

Location	Contents	
5144	62n1	
5145	JMS*	.+1
5146	0200	

Punching

There are 6 control letters associated with punching. These are:

↑B - do a BIN punch from ↑L to ↑U from current data field.*

↑F - select field punching on BIN tapes:

0↑F means do NOT punch field bits
1↑F means DO punch field bits

↑G - punch leader/trailer and clear checksum

↑V - do a RIM punch from ↑L to ↑U from current data field.*

↑W - select low/high speed punch:

0↑W means low speed (ASR) punch
1↑W means high speed punch.

When the low speed punch (ASR) is selected, XDDT will halt to allow the punch to be turned off and on. For the high speed punch, no halts are done. See below for exactly when halting is done.

↑X - punch checksum, trailer, and clear checksum. (implicit ↑G)

* AC displays the current punching address.

Select Commands

Two commands set states for the punching. Neither actually cause punching.

Field

↑F allows the user to decide if he wants or does not want a field punch (Channel 8-7 and field) on his BIN tape. (N.A. for RIM punching). "Zero ↑F" means no field punch will be put on the tape. "Non-Zero ↑F" requests a field punch each time BIN punching is requested.

↑F alone displays the state in octal. It is set to 0 (no field punch) upon loading.

Device Select

↑W allows the user to decide between the ASR punch (low speed) or the high-speed punch. "Zero ↑W" selects the ASR. "Non-Zero ↑W" selects the high-speed punch. ↑W alone displays the state in octal. It is set to 0 (ASR) upon loading.

Three other regular XDDT commands are used to select the field and the lower and upper addresses of punching. "Expr #" selects the field from which the data will be obtained. If ↑F≠0, then the BIN tape will have the selected field punched on the tape.

↑L and ↑U set the lower and upper limits of the punching.

Punching Commands

Leader/Trailer

↑G punches leader and/or trailer and clears the checksum count. Therefore, always begin a BIN punch sequence with ↑G and then do NOT use it until the BIN punch is done. If ↑W=0=ASR, XDDT halts right after echoing " ↑G". Turn punch on and hit CONT. At the end, XDDT halts. Turn punch off and hit CONT. XDDT is now ready for any valid input.

Checksum

↑X punches the checksum and trailer and clears the checksum cell. Perform ↑X after all the BIN punching is done. If ↑W=0=ASR, XDDT halts right after echoing " ↑X". Turn punch on and hit CONT. At the end, XDDT halts. Turn punch off and hit CONT. XDDT is now ready for an valid input.

Bin Punching

↑B punches a BIN tape from the low address (↑ L) to the upper address (↑U) from the data field selected (#). It also keeps a running checksum value of what it has punched. If W=0=ASR, XDDT halts right after echoing " ↑B". Turn punch on and hit CONT. At the end, XDDT halts. Turn punch off and hit CONT., XDDT is now ready for any valid input.

Example: Punch locations: 00000-00177
10200-10377

with field punches. Here is the sequence of typing necessary.

```
      0#
      1 ↑F

      n ↑W        (the following will be annotated for Halts)
                        n selects device
      0000 ↑L
      0177 ↑U
       ↑G         (n=0=turn punch on, hit CONT)
(done)            (n=0=turn punch off, hit CONT)
       ↑B         (n=0=turn punch on, hit CONT)
(done)            (n=0=turn punch off, hit CONT)
      1#
      200 ↑L
      377 ↑U
       ↑B         (n=0=turn punch on, hit CONT)
(done)            (n=0=turn punch off, hit CONT)
       ↑X         (n=0=turn punch on, hit CONT)
(done)            (n=0=turn punch off, hit CONT)
```

Tape is done.

Rim Punching

↑V punches a RIM tape from the low address (↑ L) to the upper address (↑U) from the data field selected (#). Since RIM format has no provision for field information, no field punches can be done. The user may RIM punch from any field. Be careful not to mix up fields.

If ↑W=0=ASR, XDDT halts right after echoing " ↑V". Turn punch on and hit CONT. At the end, XDDT halts. Turn punch off and hit CONT. XDDT is now ready for any valid input.

Example: Punch a RIM tape of XDDT with extra user symbols. (assume XDDT is in field 1):

```
      1#
      ↑ H 2777
      3000↑ L     (2777+1 for bottom of XDDT)
      7577↑ U
      n ↑W        (select low/high speed punch)
       ↑G         (n=0=ASR.  Turn punch ON, hit CONT)
(done)           (n=0=ASR.  Turn punch OFF, hit CONT)
       ↑V         (n=0=ASR.  Turn punch ON, hit CONT)
(done)           (n=0=ASR.  Turn punch OFF, hit CONT)
       ↑G         (n=0=ASR.  Turn punch ON, hit CONT)
(done)           (n=0=ASR.  Turn punch OFF, hit CONT)
```

Tape is done. Note: do not use ↑X.

At any time, either mode of punching may be killed by striking any key. When this is done, NO HALT occurs.

APPENDIX

The initial XDDT symbol table includes the eight basic PDP-8 instructions: AND, TAD, ISZ, DCA, JMS, JMP, IOT, OPR plus the following Operate-IOT symbols:

Symbol	Value (octal)	Symbol	Value (octal)
HLT	7402	RTF	6005
RIB	6234	SGT	6006
RMF	6244	CAF	6007
RIF	6224	IOF	6002
RDF	6214	ION	6001
CIF	6202	OSR	7404
CDF	6201	SZL	7430
SKP	7410	SNL	7420
GLK	7204	SNA	7450
STL	7120	SPA	7510
LAS	7604	SZA	7440
CIA	7041	SMA	7500
STA	7240	IAC	7001
BSW	7002	RTL	7006
MQL	7421	RTR	7012
SWP	7521	RAL	7004
ACL	7701	RAR	7010
CSWP (CLA,SWP)	7721	CML	7020
CAM	7621	CMA	7040
SKON	6000	CLL	7100
SRQ	6003	CLA	7200
GTF	6004	NOP	7000
MQA	7501	CLA2*	7600

* used when typing out group 2 micro instructions

XDDT QUICK REFERENCE

SPACE	Inclusive OR, instruction separator
+	Two's complement PLUS
-	Two's complement MINUS
.	Symbol constituent
)	Take as BCD
*	Current location and Indirect Addressing
#	Last expression and Field Setting
;	Separator for searches, symbol definitions, and setting L and AC
,	Define name symbol as *
<	Negative search between limits
>	Positive search between limits
NULL	Ignored
Leader (200)	Read binary tape
RUBOUT,?	Ignore current expression
Illegal Character	Bell rings, ignore character

Mode	Equivalence	Examination
Instruction	←	/
Name Symbol	$	:
DEC-338 Command	&	\ (Shift L)
Octal	=	[(Shift K)
BCD	%	] (Shift M)

CR	Modify open location (if one)
↑	Like CR, then open *-1
LF	Like CR, then open *+1
(	Modify open location, then [
↑A	Define DEC-338 Command
↑B	Bin Punch
↑C	Return to Monitor
↑D	Examine or modify saved Data Field
↑E	Type name symbols on exact match
↑F	examine or modify Field Flag
↑G	Punch leader/trailer
↑H	Type highest available location
↑I	Examine or modify saved Interrupt
↑K	Kill name symbol or all name symbols
↑L	Examine or modify lower limit
↑N	Define name symbol
↑O	Type all locations in octal
↑P	Define Operate-IOT instruction
↑Q	Examine or modify MQ
↑R	Type name symbols as relative addresses
↑S	Type all locations as name symbols
↑T	Examine or modify saved Teletype output flag
↑U	Examine or modify upper limit
↑V	RIM Punch
↑W	Examine or modify punch device
↑X	Punch Checksum & Trailer
↑Z	Zero or set memory between limits
'	Go to user's program
"	Set or remove breakpoint
!	Proceed back to user's program
@	Examine or modify Link and Accumulator