The

ProDev(R) DDT8 v1.6 & DDT16 v1.7

(Development & Debugging Tool)

User Manual

ProDev, Inc.

by:

Chuck Kelly

Christ is the Answer!

CUSTOMER SATISFACTION

Should you discover any defects in either this manual or the ProDev DDT card, ProDev, at our option, will replace the documentation or provide revisions and will replace or repair the ProDev DDT at no charge to you during the 180-day period after you purchased the product.

LIMITATION ON WARRANTIES AND LIABILITY

Even though ProDev has tested this product for proper operation, neither ProDev, its dealers, representatives or agents make any warranty or representation, either expressed or implied, with respect to this documentation or the accompanying hardware and/or firmware. ProDev disclaims any and all liabilities for direct, indirect, incidental, or consequential damages as a result of using this manual or ANY ProDev hardware and/or firmware. ProDev shall have no liability for the loss of any programs or data, including the cost of recovering these programs or data or for any other losses which may occur as a result of using this product. This warranty is void in the cases of misuse or damages resulting from sources other than the operation of this product. Some states do not allow the limitation of implied warranties or liability for consequential damages, so the above limitations may not apply to you.

COPYRIGHT

CONTENTS

CHAPTER 1
INTRODUCTION

INTRODUCTION

The ProDev DDT8 is compatible with the Apple //e and enhanced //e. The DDT16 is compatible with the Apple IIGS. The 6502, 65C02 and 65802 (65816 IIGS) are supported. This manual is written with the assumption that the user is familiar with the Apple II series of computers. A working knowledge of assembly language programming for the 6502 or 65816 is also assumed, but you do not need to be an experienced programmer to begin using the ProDev DDT. Indeed, using the ProDev DDT should help you in your efforts to understand assembly language programming.

The ProDev DDT contains one of the most powerful debugging programs available for the Apple II computer. But the ProDev DDT is much more than just a tracing program. It is a combination of a very extensive debugging program along with powerful hardware features, all working together to create, what we feel, is the most powerful and unique programming tool available for the Apple II computer.

Because the ProDev DDT's program is entirely in ROM (Read Only Memory), it's always there when you need it. You don't need to worry about interfering with the program you are working with because the ProDev DDT is virtually transparent to the computer. In addition, there is a hardware STOP ON ADDRESS which actually watches the addresses as they are being sent out by the computer and stops the program when the location you have specified is accessed. Another hardware feature is the EXECUTION TIMER, which counts the number of machine cycles a section of program takes to execute. The TRACE function is executed with hardware and not software like the others, so it allows you to actually see the affect of illegal instructions on your program flow. The ProDev DDT's program is written entirely in Assembly language for the fastest operation possible.

From the very beginning the ProDev DDT was designed to be used by the demanding professional who will appreciate the many powerful features incorporated into the design. For example:

Full support of the 6502, 65C02, 65802 (65816 IIGS)!
The ability to use external displays to do program debugging while viewing your program's output on the Apple screen!
Tracing programs that actually reside in the text RAM area!
Access to the Non Maskable Interrupt for gaining control of programs that are in tight loops or lost in never, never land!
And many more features.

SYMBOLS AND CONVENTIONS OF THIS MANUAL

All commands to the ProDev DDT must end with pressing the "Return" key. Throughout the rest of this manual that fact will be assumed. A few examples use the symbol "<cr>" to indicate pressing the "Return" key.

Some inputs require entering "control characters". Control characters are represented by "<ctrl>-Letter". To enter a control character, hold down the "Control" key and press the desired key.

Command items in brackets "[A]" are optional but the operation of the command may be different if they are omitted. The letter "A" is used to indicate a memory Address. The letter "N" represents a Number other than an address.

All numbers displayed by the ProDev DDT are in Hexadecimal (base 16) format. In order to save screen space, dollar signs do not precede the digits.

The ProDev DDT assumes all inputs are in Hexadecimal format. Preceding dollar signs are optional. To specify a Decimal entry you must precede the number with an exclamation point "!".

EXAMPLE: Inputs of 10 or $10 both equal $10 (16 decimal).
An input of !10 is equal to $0A (10 decimal).

The command prompt for the ProDev DDT is ":". The prompt is changed to "!" when you enter the MINI-ASSEMBLER. When a command prompt is displayed it indicates the ProDev DDT is waiting for your input.

The term "target" refers to the program that is being debugged. For example, "the target's text display" would refer to the output generated by the program being debugged.

The letter "n" is used to represent the slot number of the ProDev DDT card. If an example has an address of "Cn00" you would substitute the slot number of the ProDev DDT card for the letter "n". If the ProDev DDT card is installed in slot number 4 you would use "C400" as the address.

The symbols {DDT8} or {8} are used in this manual to indicate features related only to the DDT8 for the Apple //e. The symbols {DDT16} or {16} refer to the DDT16 for the Apple IIGS.

INPUT ERRORS

If you input a command incorrectly, the ProDev DDT will usually respond with a "beep" and a circumflex "^" to indicate the general area in which the command was in error, followed by an error number. See Appendix E for a list of error numbers. In some cases a specific message will be displayed describing the error. Every effort was made to make the ProDev DDT as user friendly as possible without sacrificing too much valuable program space.

BANK NUMBERS

Bank numbers may be entered for most commands. VERY IMPORTANT!! If no bank number is specified, then the last bank number referenced by the ProDev DDT will be used. Bank numbers must be followed by a slash "/". (E.G. 00/2000).

{DDT8} - Bank 01 refers to the alternate 64K of RAM in 128K Apple //e's. If you have an Applied Engineering Ramworks board or equivalent, you may specify bank numbers greater than 01.

NOTE! The Ramworks manual refers to the lowest bank on the Ramworks card as bank 0. We refer to the main system memory as bank 0. Therefore, the ProDev DDT displays the lowest bank on the Ramworks card as bank 1.

***** ALERT *****
Entering bank numbers greater than the actual amount of memory in your system may result in lost data in bank 0.

INSTALLATION INSTRUCTIONS

***** **W A R N I N G** *****
Always make sure the computer power is off when installing or removing any peripheral card, including the ProDev DDT. This is very important, and failure to do so will result in damage to the ProDev DDT card and to your computer and will void your warranty.

Installation steps:

1. Turn off the power to your computer and all equipment connected to it.

***** **W A R N I N G** *****
Plugging the ProDev DDT card into the AUX CONNECTOR of an Apple //e will result in severe damage to the ProDev DDT and to your Apple.

2. Along the back edge of the Apple are several long connector slots. The ProDev DDT card will work in any of these slots with two exceptions. (DDT8) Do not use the AUX connector which is positioned closer to the keyboard and next to the power supply. Do not use slot #3 if you have an 80 column card installed in the AUX connector. (DDT16) Do not use the Memory Expansion connector which is positioned closer to the keyboard.

3. The ProDev DDT card has a row of gold plated contacts along one edge. Try not to touch these contacts. Be sure the power is still off.

4. Follow the instructions supplied with your computer for installing peripheral cards.

5. The cable may be fed through any of the available openings in the back of the computer.

6. Place the cover on your computer.

7. Turn your computer power back on and activate the ProDev DDT card by entering the following commands:

(DDT16) The control panel setting must be set for "YOUR CARD" in the DDT16 slot.

] CALL -151 or **]mtr** <cr> "This puts you in the monitor program".
*** Cn00G** <cr> "n is the slot number of the ProDev DDT".

You should be greeted with the ProDev DDT version number and copyright notice. If not, double check the installation procedures and read the section on "IT DON'T WORK".

CHAPTER 2
GETTING STARTED

FAST START

The procedure to follow when using the ProDev DDT might be something as follows:

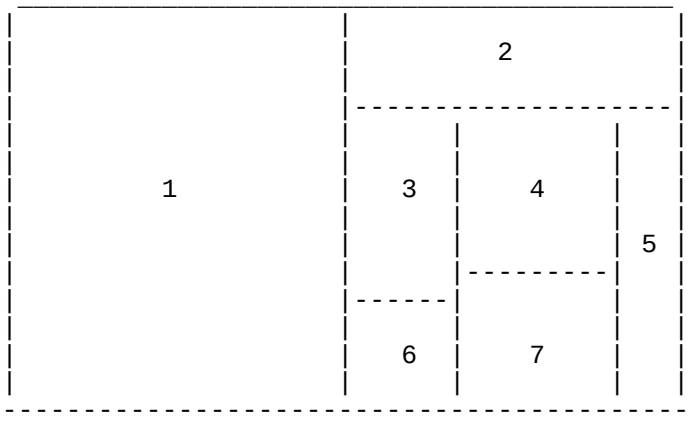
The ProDev DDT card must be initialized after powering up the computer. If you are using the DDT16 you may use the Init provided on the accompanying disk to automatically initialize the DDT during a GSOS boot. See the instruction provided in the "read.me" file for further information.

You may also initialize the DDT manually by getting into the Apple monitor, (from a BASIC prompt type "call -151" or "mtr") and entering the command "Cn00G", where 'n' is the slot number the DDT is installed in. Using the commands "PR#n", "IN#n", "n<ctrl>P", etc. will also start the DDT but they disable any DOS hooks and should be avoided.

If you enter the ProDev DDT via "Cn00G", you can enter "QU" at the DDT prompt to return to the Apple monitor.

WINDOWS

After starting the ProDev DDT you will see a windowed 80 column display.



WINDOW #1

Information from Tracing, listing, mini-assembler etc., is displayed in this window. The flashing cursor that is awaiting your input, also known as the command prompt, is usually in this window.

WINDOW #2

```
 M = STK REA WRI LNG BNK PG2 80S CXR   { pseudo register
     MNE MNE MNE ROM  2  OFF OFF ON

 KEY BRK TYP SAV TXT MIX HGR 80C ALT   { state flags
 93  OUT C02 ON  ON  OFF OFF OFF OFF
```

The first 2 lines of window 2 contain the current state of the (M)achine state pseudo register. M is not really a physical register in the processor, rather it is a pseudo register that is created by the ProDev DDT. The M register is also displayed along with the processor's registers when tracing or displaying registers. The Apple IIGS monitor also contains an M pseudo register which is similar, but not identical, to the ProDev DDT's M register.

The meaning of the M state register bits from left to right are as follows:

STK = reflects the state of the ALTZP switch ($C016)
MNE = main memory AUX = auxiliary memory (*)

REA = reflects the state of the RAMRD switch ($C013)
MNE = main memory AUX = auxiliary memory (*)

WRI = reflects the state of the RAMWRT switch ($C014)
MNE = main memory AUX = auxiliary memory (*)

LNG = reflects the state of the language card switch ($C012)
ROM = Autostart ROM RAM = language card RAM (*)

BNK = reflects the state of the LC bank switch ($C011)
1 = bank #1 in 2 = bank #2 in (*)

PG2 = reflects the state of the PAGE2 switch ($C01C)
OFF = Page 2 off ON = Page 2 on (*)

80S = reflects the state of the 80STORE switch ($C018)
OFF = use RAMRD, RAMWRT ON = access display page (*)

CXR = reflects the state of the SLOTCXROM switch ($C015)
OFF = Slot ROM at $Cx00 ON = Internal ROM at $Cx00 (*)

The bottom two lines of window 2 contain the current state of various flags and switches. From left to right they are as follows:

KEY = shows the current ASCII key code (with high bit set) that will interrupt the ProDev DDT while Tracing or Executing multiple instructions. The default setting is $93 which is Control-S.

BRK = indicates if real breaks are present in the target program.
OUT = no real breaks IN = real breaks exist

TYP = indicates the type of processor the ProDev DDT is set to recognize when assembling or disassembling instructions.
02 = 6502 C02 = 65C02 802 = 65802{8} 816 = 65816{16}

SAV = reflects the state of the ProDev DDT's "Text Screen Save" mode.
ON = screen save on OFF = screen save off

TXT = reflects the state of the TEXT switch ($C01A).
OFF = text mode off (graphics) ON = text mode on (*)

MIX = reflects the state of the MIXED switch ($C01B).
OFF = full page graphics ON = mixed txt & gr (*)

HGR = reflects the state of the HIRES switch ($C01D).
OFF = lo-res graphics ON = hi-res graphics (*)

80C = reflects the state of the 80COL switch ($C01F).
OFF = 40 column display ON = 80 column display (*)

ALT = reflects the state of the ALTCHARSET switch ($C01E).
OFF = Apple][char set ON = alternate char set (*)

(*) is the state when the switch is greater than 128 ($7F).

WINDOW #3

The contents of twelve memory locations are displayed as a hexadecimal number and equivalent ASCII character. The addresses are specified by a two digit bank number followed by a four digit address.

WINDOW #4

The current location of the Hardware Breakpoint (if any) is displayed at the top of this window followed by the ten most recently accessed Breakpoints. See "HB" & "SB" for a further description of the breakpoint format.

WINDOW #5

An area of the system stack RAM is displayed around the current location of the stack pointer which is indicated by "<".

WINDOW #6

This is the effective address window. The actual address that a program instruction is referring to is not always obvious. For example LDA ($34),Y. When tracing or disassembling instructions, the ProDev DDT calculates the effective address and displays its contents in this window surrounded by the two locations above and below. The effective address is indicated by "<".

The ASCII equivalent of each locations hexadecimal contents is also displayed.

{DDT8} - The current settings of the systems various memory switches are used to calculate if the address is in main or auxiliary memory. The corresponding bank number (00/ or 01/) is displayed.

WINDOW #7

This is the protection window. You may specify three types of memory protection in any of six address ranges. See "PW" for a further explanation.

STEPPING CODE

Stepping a program is very simple. Let's assume you would like to start stepping a program beginning at memory location $800. The first thing you need to do is set the program counter to point to location $800. You do this from the ProDev DDT by using the command "PC=". To set the program counter to $800 you would enter the following:

:PC= 800 <cr> or **PC= $800** <cr> "dollar sign is optional".

If you want to see the instruction at location $800, use the "DR" command. For example with the DDT8:

```
:DR <cr>
A=00 X=00 Y=00 S=E0 M=0A L=0 P=--1----C       <--- displayed
00/0800:EA          NOP                       <---
```

The first line of the display contains the contents of the accumulator (A=00), X index (X=00), Y index (Y=00), stack pointer (S=E0), M pseudo register (M=0A), language card condition (L=0) and the status register (P=--1----C). For a detailed description of these registers, refer to the "DR" command in Chapter 5. The second line displayed shows the address of the program counter, including the bank number (00/0800:), the hexadecimal form of the instruction (EA), and the instruction mnemonic (NOP).

To begin stepping you would use the "ST [N]" command. If you wanted to step one instruction and stop you would enter the following:

```
:ST <cr>  or  ST 1 <cr> "One is assumed if no number entered".
A=00 X=00 Y=00 S=E0 M=0A L=0 P=--1----C      <--- displayed
00/0801:EA          NOP                      <---

T:    <--- The new prompt indicates "Trace/Step" mode.
```

The registers displayed reflect any changes caused by the instruction just stepped. The new program counter is displayed along with the instruction that would be executed next.

When the ProDev DDT is in the "Trace/Step" mode, pressing the "Return" key will cause the instruction pointed to by the program counter to be traced or stepped. The "Trace/Step" mode is disabled by entering any other command.

CHAPTER 3
SETUP COMMANDS

M or ? - MENU

Entering the command "M" or "?" will produce a list of the available commands. You will find this very convenient and after you have used the ProDev DDT for a while it may not be necessary to refer to the manual. The menu listing indicates the way in which the command must be entered and a brief reminder of what the command does.

MO - MODE

The MODE command allows you to change some of the DDT's settings.

Example:

MO

--- Mode of Operation ---

Text screen save
1=on 2=off
Enter <1> :

CPU
1=6502 2=65C02 3=65816
Enter <2> :

I/O slot #
1:out 2:in/out 3:normal
Enter <3> :

Text screen save

If you are not concerned about overwriting the text RAM area you can turn the screen save off by entering a "2". This will eliminate the strange screen updating that occurs when tracing code. When this mode is on, the contents of the text RAM area are preserved by exchanging the contents of the screen memory with a buffer in the ProDev DDT's own RAM memory.

If you are sending the ProDev DDT's output to a device other than the main screen the screen save should remain on.

If the target program is using the text RAM area for data or code storage, the screen save must be on to prevent the ProDev DDT from overwriting the text RAM area. With the mode on, it is possible to trace code that resides in the text RAM area.

CPU

Enter 1,2 or 3 to select the type of CPU that your code was written for. Your selection is used by the DDT so the proper mnemonics will be displayed during disassembly or accepted by the DDT's mini-assembler.

(DDT8)
** See the note in Appendix B about 65802 **
Of course, selecting the 65C02 or 65802 will not allow you to trace or run the extra instructions provided by these processors unless your computer is so equipped.

I/O slot #

Select the Input/Output slot of the DDT.

Slot 3 uses the Apple's keyboard and screen for input and output with the ProDev DDT.

Slot 2 redirects both input and output to slot 2. This option allows controlling the ProDev DDT from a remote terminal. The output from the ProDev DDT will be sent to the terminal screen and the input to the ProDev DDT will be entered on the terminal's keyboard. The input and output of the target's program is not changed. With this method of debugging, it is possible to observe the output from the target's program, be it text or graphics, and the output from the ProDev DDT at the same time. See Chp. 8 for more information.

Slot 1 directs the ProDev DDT's output to slot 1. Input is via the Apple's keyboard. This option is intended for generating a hard copy of your work or for users who do not have access to a terminal but would still like to trace programs and view their programs output at the same time. Be sure your printer is ready and on line before issuing this command. See Chp. 8 for more information.

esc - : LEVEL

The "escape" key, is used to return to the ProDev DDT's command level from various commands or windows.

SS A - SET SWITCH

This command allows you to set any of the Apple's soft switches that exist in the address range from $C000 to $C0FF. To do so, simply enter "SS" followed by the lower byte of the desired address. For example:

    :SS 50 sets switch $C050 (graphics on)
    :SS 03 sets switch $C003 (read auxiliary memory)

To ensure that all types of switches are set, the ProDev DDT does one write and two reads of the selected location.

SD T - SET DISPLAY Type

Set display is similar to Set Switch. It deals only with display formats and accepts more easily remembered switch identities. For example, to turn On the Double hi-res display, simply enter SS D. The following switch types are recognized.

1 - Select page 1
2 - Select page 2
4 - Select 40 column display
8 - Select 80 column display
A - Select Alternate character set
D - Display Double hi-res graphics
F - Display Full screen graphics
H - Display standard Hi-res graphics
L - Display Low-res graphics
M - Display mixed text and graphics
N - Select Normal character set
S - Super hi-res {DDT16}
T - Display Text screen {DDT16 turns off Super hi-res}

ON - DISPLAY ON

ON will restore the main ProDev DDT screen from almost any condition. If "text screen save" is on, the contents of the screen will be saved.

OFF - DISPLAY OFF

OFF disables all screen displays by the ProDev DDT. This speeds up tracing, which spends a great deal of time updating the screen information. The EXecution command is still much faster but there are certain situations where the trace mode may be preferred. The saved text screen will be restored if "text screen save" is on. The various display switches will be restored to there original settings. This allows you to view the output from the target program.

KEy - SET

Key is the name given to the one ASCII key code that will interrupt the ProDev DDT when it is tracing or executing multiple instructions. The default is $93 (control-S). The high bit is always on. To change Key simply enter:

:KE the ProDev DDT will respond with:

:KEY =

The next key that you type will be accepted as the new code for KEY. If you wanted control-A you would simply hold down the "Control" key and press "A".

Entering the correct Key while tracing or executing only pauses the operation. You may elect to continue by pressing any other key or exit to the ProDev DDT command level by pressing "esc".

CHAPTER 4
BREAK POINTS

SB [T A.N] - SET BREAK POINT

Break points work like stop signs in the target program. By strategically placing break points you can determine many things about the logical flow a program follows. Break points must be placed at EXACTLY the same address as the first byte of the instruction where you want the program to stop. (see the following example)

TYPES OF BREAKS

The ProDev DDT has two types of software break points, "Real" and "Implied". There is also a hardware break point, "Hard Break", that will be described later.

A Real break inserts "BRK" (break, HEX 00) commands in place of any instruction that is in RAM memory.

The Implied break does not modify your program. This allows Implied breaks to be used in ROM locations or write protected RAM. Implied breaks are ONLY affective when STepping, TRacing or EXecuting code and will not stop a program if it is running in real time.

The form of the Set Break instruction is:

SB [Type Address . Number]

Where Type is input as "R" for Real break or "I" for Implied break. Type "R" is assumed if no Type is indicated.

Address is any Hex number or decimal equivalent. Bank numbers may be specified and must be followed by a "/" slash.

Number is any Hex or decimal number in the range of $0001 to $FFFF. This is the number of times to pass this breakpoint before stopping program execution. Entering $0000 will result in passing the breakpoint 65,536 times. Therefore, the actual range is 1 to 65,536 or $0001 to $0000 but that looks weird. If no value is input for Number then the default of "1" will be assumed.

If you specify a break where one already exists, the new break information will replace the old.

Entering "SB<cr>" will provide a list of the current SOFT BREAKS, if any. The break points are listed in the order last accessed, with the most recent being at the top left of the display. This command is primarily intended for use with a serial terminal but is also useful if you have entered more breakpoints than can be displayed in the "Break Window".

A maximum of 50 breakpoints is allowed at any one time. The ten most recently accessed breakpoints are displayed in the "Break Window" of the ProDev DDT's main display screen.

EXAMPLE

SB I 800.1
Sets an Implied break at location $800 in the current bank and will stop the first time it is encountered.

SB R 1/0900.3
Sets a Real break at location $900 in bank 1 {DDT8 aux memory} and will stop the third time it is encountered.

SB R 03/2345.!200
Sets a Real break at location $2345 in bank 3 {DDT8 aux memory} and will stop after decimal 200 encounters.

SB 2003
Sets a Real break at location $2003 in the current bank and will stop the first time it is encountered.

EXAMPLE

Stop the program below when the "JMP 3000" instruction is reached.

```
00/2000: A9 A0          LDA  #A0
00/2002: 20 ED FD       JSR  FDED
00/2005: 4C 00 30       JMP  3000
```

We will use a Real Type breakpoint so the program will be interrupted from running in real time. We also want to stop the first time the breakpoint is encountered. The following will place the breakpoint:

:SB R 0/2005.1 or **:SB 2005** { assumes bank 0 is current default

The new breakpoint should appear in Window #4, the Break Window.

{Now run the program.}

:GO 2000 { start running program in real time

When the Breakpoint is reached the program is halted and the processor's registers are displayed as follows {DDT8}.

```
BREAKPOINT
A=A0 X=00 Y=FF S=FB M=0A L=0 P=N-1-----  { registers may vary
00/2005:4C 00 30    JMP 3000
```

Note! The word "BREAKPOINT" was displayed, indicating the ProDev DDT stopped the program execution with a Breakpoint.

***** HOW IT WORKS *****

When you enter a Breakpoint, the ProDev DDT saves the current instruction to its own internal memory and replaces the instruction byte with a BRK command (HEX 00). Then it waits for a BRK command to be executed. When it sees one, it checks to see if it is a break that you have entered. If it is, the ProDev DDT stops the program and gives you control. However, when

it displays the instruction, it displays the original instruction byte and not the BRK command which really occupies that location.

When the "GO" command is used to run a program and a Real Break is encountered, the program is stopped, the DISPLAY REGISTERS command is automatically executed, and you are returned to the ProDev DDT's command level.

***** MOVING CODE & BREAKPOINTS *****

Caution!!! If a section of code containing Real Breaks is relocated the ProDev DDT will not recognize the Breaks and undesirable results will occur.

If part of a program containing Real Breaks is overwritten by new code, the ProDev DDT will not know about it. If you then proceed to remove the Real Breaks the existing code will be overwritten by the contents of the original breakpoint location.

CONDITIONAL BREAKPOINTS {DDT16}

Conditional breakpoints extend the function of normal breakpoints by allowing you to use the contents of the processor's registers as break conditions. The command takes the following form:

```
SB C $01/2345 , X = $6789
  ^  ^^^^^^^   ^ ^  ^^^^
  |_____|_______|_|___|____ Indicates a conditional breakpoint
        |_______|_|___|____ Address of breakpoint
                |_|___|____ Register for comparison
                  |___|____ Logical operator
                      |____ Value to compare to register
```

This example would cause a break at location $01/2345 if the contents of the "X" register was equal to $6789.

The registers that may be used for comparison are:

A,X,Y,S,D,B,M,Q and P.

The allowable logical operators are:

<	less than
>	greater than
=	equal
#	not equal
[	less than OR equal
]	greater than OR equal

All comparisons are done in full 16 bit native mode. 8 bit registers are compared to the low byte of the comparison word.

Conditional breakpoints modify the target program the same way Real breakpoints do and therefore they may not be used in ROM code locations.

The purpose for adding this feature was to allow breaking on particular tool calls. Tool calls are made by placing a two-byte value in the X register that identifies the tool set and the command value to use and then doing a JSL to the common tool entry point at $E1/0000. To break on a particular tool call we simply determine the value of the X register that corresponds to the desired tool to break on and place a conditional break point at location $E1/0000. The following statement would break on a TLShutDown tool call:

```
SB C $E1/0000,X=0301
```

RB A - REMOVE BREAK

The REMOVE BREAK command allows you to remove any of the Breakpoints which you have entered. The address is mandatory, as indicated by the fact that the letter "A" is not enclosed in brackets. When you remove a Breakpoint the original instruction is fully restored and the address is removed from the list of Breakpoints.

EXAMPLE

Remove the Breakpoint from address $2005

:RB 0/2005 or **:RB 2005**

HB [A] - HARD BREAK

The HARD BREAK command appears very similar to the Set Break command. The command actually activates the one hardware Stop-On-Address the ProDev DDT is equipped with. Omitting the optional address will result in a display of the current HARD BREAK address. When activated, the ProDev DDT watches the addresses that are being accessed by the processor. It stops the target program when it sees the specified address being accessed.

The Hardware Break may be used as a real time breakpoint for programs that are running from read only memory (ROM). This command may also be used to monitor any memory location and stop your program when that location is accessed. For example, if a certain memory location is being modified and you wish to find out which instructions are causing the modification, you simply use the HARD BREAK command to stop the program when the memory location is accessed.

The bank address specified in the Hard Break address is not actually used in the DDT's hardware comparator. This is because the DDT's comparator is only able to compare 16 bits at once. The comparison of the bank address is done in software by looking at the target programs 'program' and 'bank' registers. If no match is found the Hard Break is ignored. You can tell the DDT to ignore bank addresses when doing a Hard Break by using the HBX command:

```
HBX address
```

This will place a Hard Break at the specified address and will stop in all banks, not just the current program or data bank. The command is useful if your program is being stepped on and you want to find the offending instruction. The normal Hard Break would usually work unless the offending instruction is a 'long indirect' addressing type such as STA [aa]. In this

case the destination bank address is not the current program or data bank and the normal Hard Break would not stop the program.

If you enter the command correctly the Breakpoint display will be HX address instead of HB address.

EXAMPLE Used as a real time ROM breakpoint.

Let's assume the following program resides in ROM and you wish to place a real time breakpoint at location $FDB5.

```
00/FDB3  A5 3C    LDA 3C
00/FDB5  09 07    ORA #07
00/FDB7  85 3E    STA 3E
```

Place the Hard Break with the following command:

:HB 0/FDB5

The Break Window should display the new Hard Break location in the top of the window as "H00/FDB5".

Now if you run the program the Hard Break will interrupt execution when $FDB5 is reached and will display the following.

```
HARD BREAK
A=07 X=00 Y=00 S=F4 M=0A L=0 P=--1B----  { contents may vary
00/FDB7:85 3E       STA 3E
```

Note! The address displayed is $FDB7 and not $FDB5 where you placed the Hard Break. This assumes you started the program running prior to location $FDB3. This is normal operation. One or two addresses following the selected Break location is displayed because of the way the CPU handles interrupts.

***** ADVANCED EXPLANATION *****

When the ProDev DDT sees the address $FDB5 it sends the processor an interrupt. However, the processor does not stop program execution until it has finished the instruction that it is currently working on. Which in this case is "ORA #$07". The processor completes the "ORA #$07" instruction and then gives control over to the ProDev DDT which is why the address displayed by a HARD BREAK is the instruction or two following the actual address of the BREAK.

EXAMPLE memory location being changed

Memory location $21 is being changed when you run your program and you wish to know which instructions are responsible. Simply use the HARD BREAK feature as follows:

:HBX 0/21 or :HBX 21

When you run your program and location $21 is accessed your program will be stopped at the instructions following the one which accessed location $21. If

you wish to check for any other instructions which also access location $21, simply enter the "GO" command to continue running your program. If any other instructions access location $21 they will also cause the HARD BREAK to stop the program.

RH - REMOVE HARD BREAK

The REMOVE HARD BREAK command allows you to remove the current HARD BREAK. An address is not required, since there is only one HARD BREAK.

RA - REMOVE ALL BREAKS

The REMOVE ALL BREAKS command allows you to remove all existing SOFT BREAKS and the one HARD BREAK with one command.

EXAMPLE

To remove all existing BREAKS simply enter the following:

:RA

The BUTTON INTERRUPT

The BUTTON INTERRUPT gives you access to the NMI (non-maskable interrupt) line through the ProDev DDT card. This allows you to regain control of programs that are running in real time or of programs that are running out of control. The BUTTON INTERRUPT will not work until the ProDev DDT card has been initialized following a system power up.

The hardware on the DDT card generates an NMI to interrupt the currently running program.

{DDT16}
The DDT uses the NMI vector at $03FB to redirect program control to the DDT. This vector is set when you initialize the DDT. If this vector gets changed after the DDT has initialized it, the button interrupt will not get vectored to the DDT and will cause bad things to happen. Booting ProDOS 8 is one way to change the $03FB vector. Always try to initialize the DDT just prior to using it to avoid this problem.

{DDT8}
The DDT uses hardware to disable the Apple //e's ROMs and take control of the NMI handler. It is possible to interrupt any program at any time.

CHAPTER 5
DISPLAYING AND MODIFYING

DR - DISPLAY REGISTERS

Display registers produces the following displays:

```
{DDT8}
A=00 X=00 Y=00 S=E0 M=0B L=0 P=--1B----    { contents may vary }
00/FF69:20 E0 FE    JSR FEE0

{DDT16}
A=0000 X=0000 Y=0000 S=01E0 D=0000 B=00    { contents may vary }
M=09 Q=FF P=30=--MX---- L=0 E=0 I=0
00/0800:EA          NOP
```

The various registers are represented by single letters with the exception of the program counter which is represented in number form.

The letters and their corresponding registers are as follows:

```
A = Accumulator
X = X index
Y = Y index
S = Stack pointer
M = Machine state (pseudo register)
L = Language card state (pseudo register)
P = Processor status

{DDT16}
D = Direct page
B = Data Bank
Q = Quagmire state
E = Emulation (1 = 6502, 0 = Native)
I = Interrupt (1 = Interrupts disabled)
```

The status bits are also represented by single letters as follows:

```
N = Negative flag
V = oVerflow flag
1 = expansion flag (always one) {8}
M = Memory size {16}
B = Break command flag {8}
X = indeX reg. size {16}
D = Decimal mode flag
I = Interrupt disabled flag
Z = Zero result flag
C = Carry flag
- = flag not set (flag clear)
```

"M=0B" shows the condition of the M pseudo register.
"L=0" gives the condition of the language card where:
0 = ROM selected, 1 = RAM bank #1, 2 = RAM bank #2

{DDT8}
The bank number (00 or 01) is calculated for the current program counter location according to the settings of the various memory configuration switches. The bank number is then displayed before the program counter and followed by "/". In this example "00/FF69", the bank is 00 and the program counter is $FF69.

R=N - REGISTER = N

The REGISTER = N command allows you to enter a HEX value into any of the processor's registers. "REG=" indicates the letter(s) representing any of the valid registers followed by an "=" equal. The program counter is signified by "PC", all of the other registers are signified by the same letters described above in the DISPLAY REGISTERS command with the exceptions; M, Q and L registers are pseudo registers and may not be directly modified.

EXAMPLE

:A= $34 or **:A=34** {Change the value of the accumulator to $34}

Note! The dollar sign is optional and also note the spaces before the number are ignored. In both cases the value $34 is entered into the accumulator. To verify this simply use the DR command to display the registers.

The processor status flags may also be individually set or cleared by entering "P=" followed by a "Return" and then entering a "1" or a "0" for individual flags.

EXAMPLE

:P=

FLAGS NV1BDIZC
STATUS 00110000
ENTER 1 {Set the Z flag}

{DDT16}
:I= 1 - I Flag

On a][GS interrupts are used extensively. If you are tracing code with the DDT16 and an interrupt occurs the normal flow of the computer program takes you to the interrupt handler routine. This can be very frustrating if you are not interested in tracing the interrupt routine. To get around this problem a pseudo register has been added, the "I" register. The "I" register is used to prevent outside interrupts from being serviced while tracing code.

Enter the command "I=1" to prevent servicing interrupts while TRacing, STepping or EXecuting code. Enter "I=0" to clear the "I" register. Be sure to set the "I" flag in the Status register to the proper state to enable interrupt servicing. Please note that while the "I" register equals 1, the "I" flag in the status register will always get set to 1.

Also, please note that while tracing with the "I" register set to 1, any commands that affect the "I" flag will essentially be ignored. It is therefore

up to the user to determine the proper setting of the "I" flag in the status register after using the "I=1" command.

The "I" register will not function properly if an accelerator is enabled.

Li [A] - LIST

The LIST command causes 20 lines of code to be displayed to the current output device, starting at the address specified by "[A]". Omitting the optional address will cause the listing to start at the last address accessed by the ProDev DDT.

EXAMPLE

```
:LI 0/E000  or  L 0/E000          {Note! If the "I" is omitted
                                    then at least one space must
00/E000:RC 28 F1    JMP F128        follow the "L".
00/E003:4C 3C D4    JMP D43C
"   "   "            "
"   "   "            "
"   "   "            "
00/E02A:A5 14       LDA 14        {"L <cr>" will continue list
                                    from next instruction.
```

MD [A] - MEMORY DUMP

The MEMORY DUMP command displays 256 bytes of HEX data with 16 bytes per row. Each HEX byte's corresponding ASCII character is displayed at the end of each row. Omitting the optional address will cause the display to start at the last address accessed by the ProDev DDT. Control characters are displayed as a period.

EXAMPLE

```
:MD 0/D0D0

00/D0D0: 45 4E C4 45 4F D2 4E 45 58 D4 44 41 54 C1 49 4E  ENDFORNEXTDATAIN
00/D0E0: 50 55 D4 44 45 CC 44 49 CD 52 45 41 C4 47 D2 54  PUTDELDIMREADGRT
"   "    "  "  "  "  "  "  "  "  "  "  "  "  "  "  "  "          "
"   "    "  "  "  "  "  "  "  "  "  "  "  "  "  "  "  "          "
 etc.
```

MM [A] - MEMORY MODIFY

The MEMORY MODIFY command allows you to enter HEX data directly into memory. Data is input one byte (2 HEX digits) at a time. Pressing the "down arrow", or "Return" on a blank line, will display the next address and data byte. Press the "up arrow" to display the previous address and data. If more than two digits are entered, the ProDev DDT will take the last two digits entered. To exit the MEMORY MODIFY command, press the "esc" key.

EXAMPLE

:MM 0/2000

00/2000 00 11
00/2001 00 22
00/2002 00 33
00/2003 00 "up arrow" {Backup}
00/2002 33 56744
00/2003 00 "up arrow"
00/2002 44 {The last two digits entered were 44}
{Press "esc" to exit}

MA [A] - Mini-Assembler

The Mini-Assembler allows you to enter assembly language code directly from the ProDev DDT. This eliminates reloading your assembler, editing in your modifications, assembling and loading your program each time you wish to make a simple modification or add small sections of code to try new ideas. With the ProDev DDT's mini-assembler you can modify your program directly and as many times as needed to get the desired result. Once the program is working properly it is only necessary to edit and assemble your source program one time. This is a great time saver and should greatly increase your programming productivity.

The ProDev DDT's mini-assembler is patterned after the mini-assembler that was included in the old Apple II Monitor ROM, with some exceptions.
All mnemonics are as specified by "The Western Design Center, Inc." in their W65C816 data sheet. The addressing modes for the 6502 instructions are the same as those described for the Apple mini-assembler. The new addressing modes for the 65C02 and 65802/816 are described in the W65C816 data sheet.

Two assembler directives have been included to facilitate the entry of Data and ProDOS calls. They are:

HEX - Allows direct entry of 2 digit hexadecimal numbers.
ADR - Allows direct entry of 4 digit hexadecimal addresses. The addresses are stored in low order / hi order byte format.

Following are the addressing modes as recognized by the ProDev DDT's Mini-Assembler:

STANDARD 6502 ADDRESS FORMATS

TYPE:	FORMAT:
accumulator	Mnemonic only
implied	Mnemonic only
immediate	#n
direct	d
direct indexed	d,X
	d,Y
program counter relative	r
direct indexed indirect	(d,X)
direct indirect indexed	(d),Y
absolute	a
absolute indexed	a,X
	a,Y
absolute indirect	(a)

65C02 ADDRESSING FORMATS

TYPE:	FORMAT:
direct indirect	(d)
absolute indexed indirect	(a,X)

65802/816 ADDRESSING FORMATS

TYPE:	FORMAT:
program counter relative long	rl
direct indirect long	[d]
direct indirect long indexed	[d],Y
absolute long	al
absolute long indexed	al,X
stack relative	d,S
stack relative indirect indexed	(d,S),Y
block move	bb

Abbreviations:

n	1 byte number
d	1 byte direct address
r	2 byte relative offset
rl	3 byte relative long offset
a	2 byte absolute address
al	3 byte absolute long address
b	1 byte bank number

All input values are assumed to be HEX digits and as such a preceding dollar sign, "$nn", is optional. If fewer than the required number of digits are input for an address, the ProDev DDT will automatically add

leading zeros to fill the address; if more than the required number of digits are entered, the last to be entered will be used.

The ProDev DDT makes no distinction between direct and absolute addressing modes. If you enter an instruction which could use either direct or absolute addressing, direct will always be used.

Instructions with accumulator and implied addressing modes do not use operands.

The destination address for branch instructions is entered directly and the relative offset is automatically calculated. If the destination address is out of branch range, the ProDev DDT will respond with a "beep", the message "ERR BRANCH OUT OF RANGE" and will reject the input. If you input an instruction which the ProDev DDT does not recognize or if you attempt to modify an instruction which currently has a Real break at that address, the ProDev DDT will "beep" and place a circumflex, "^", under the general area of the problem.

Mnemonics and addressing modes of the 65C02 or 65802/816 will not be accepted unless the proper mode has been selected. (See the "MO" command for an explanation of how to change modes.)

When you enter the mini-assembler the prompt will be changed from a colon, ":", to an exclamation point, "!". The current instruction is disassembled and displayed at each address.

Note! The program counter is not affected by the "MD", "MM" or "MA" commands. This means you can be tracing a program, stop, go modify data in memory and return to tracing without losing your last position or the contents of any of your registers.

EXAMPLE

Input a short program. (This program will also be used in later examples)

```
:MA 0/2000

00/2000:00         BRK       { the current instruction is BRK
                  !LDA #00   { enter the new instruction
        A9 00                { code generated by mini-assembler
00/2002:00         BRK       { the next instruction is displayed
                  !STA 6     { etc.
        85 06
00/2004:00         BRK
                  !LDA#30    { notice that spaces are optional
        A9 30
00/2006:00         BRK
                  !STA 7
        85 07
00/2008:00         BRK
                  !LDY #1
        A0 01
00/200A:00         BRK
                  !LDA (06),Y
        B1 06
```

```
00/200C:00        BRK
                 !TAX
        AA
00/200D:00        BRK
                 !STX 3002
        8E 02 30
00/2010:00        BRK
                 !INX
        E8
00/2011:00        BRK
                 !BNE 200D
        D0 FA
00/2013:00        BRK
                 !       { "esc" returns to command level

:
```

MW - MEMORY WINDOW

Command MW allows you to select the various memory locations that you are interested in monitoring in the memory window. When you enter "MW" the cursor will be placed in the memory window on the first address. To change an address, use the Return key or Down arrow to move the cursor over the desired address. Change the desired digits by using the right and left arrows to position the cursor and simply enter the new digits. Press the "esc" key to exit the memory window. The contents of each memory location are displayed in Hex and ASCII.

In Serial mode the MW command will not display anything on your terminal but will expect you to enter the addresses anyway. You would NOT want to do that since you don't have the memory window displayed on your terminal. Use the Memory Modify command to examine memory locations instead.

PW - PROTECTION WINDOW

The protection window consists of six address ranges. Each address range consists of a type letter followed by a bank number followed by the first address of the range, a period, and the last address in the range.

There are three types of ranges. They are:

T - Trace range. A trace range allows specified sections of code to be executed in real-time. During tracing or executing, if the program counter is placed inside a T range by a JSR{8} or JSL{16} instruction, the program will start real-time execution until the instruction following the JSR/JSL in memory is reached. This is very useful for time critical sections of code such as disk I/O.

{DDT16} Note! The stack is treated as native. Failure will occur if target code is emulation mode (8 bit) JSR & stack is at $101 or $100.

P - Program only. If one or more program only ranges are specified, the program counter must remain inside one of the P ranges at all times during tracing or executing. If the program counter ventures outside these

ranges, execution will stop and the message "OUTSIDE RANGE LIMITS" will be displayed. Code that is running in real-time or is contained in a Trace range is not affected.

N - No access. This range is active ONLY when Tracing. If the effective address of any instruction, or the current program counter, enters a No access address range, program execution will be halted and the message "NO ACCESS HALT" will be displayed. This is one case when using the "OFF" command to speed up tracing may be desired.

"Space" - cancel current range. Entering a space over the range type will cancel the effect of the current range. The address of the range will not be affected and the range type may be restored.

If you enter the command "PW" the cursor will be placed in the protection window. Use the "down" arrow or "Return" key to position the cursor over the desired range. Enter the range type letter followed by the address information. Leave the window by pressing the "esc" key.

In serial I/O mode, the ranges will be displayed one at a time and you will be given an opportunity to enter new information or go to the next range.

EXAMPLES

T00/8000.8200 Begin real-time execution if the program counter enters the range of $8000 to $8200 in bank 00.

P01/0800.4000 Stop execution if the program counter gets outside the range from $800 to $4000 in bank 01.

N00/0100.01FF Stop tracing if the program counter or the effective address of any instruction enters the range from $100 to $1FF in bank 00. Works only when tracing.

CHAPTER 6
TRACING AND RUNNING

GO [A] - RUN

The GO command is used to start real time execution of the target program. If no address is specified in the command then execution will begin at the current program counter location. If the optional address is present, program execution will begin at the indicated address.

EXAMPLE

Let's assume that you have a program in memory, starting at address $2000. Now, run the program from the ProDev DDT.

:GO 2000 {Begin full speed execution at location $2000

ST [N] - STEP

The STEP command allows you to execute instructions one at a time. After each instruction is executed your program is interrupted and the DISPLAY REGISTERS command is executed. The instruction which is displayed is the next that will be executed.

The number of instructions to step is indicated by "[N]" and is optional. If no number is input then "1" is assumed. If zero is input, 65,536 instructions will be stepped unless paused by the "KEY" value (default is <ctrl>-S). Pressing "esc" will cancel the step, any other key will continue.

Once the STEP command is invoked, the prompt will change from ":" to "T:" to indicate the 'Trace/Step' mode is active. When the 'Trace/Step' mode is active it is only necessary to press the "Return" key to trace or step the current instruction. The 'Trace/Step' mode remains active until canceled by entering a different command.

If you are stepping more than one instruction (E.G. ST 5) and a SOFT or HARD BREAK is encountered, the stepping will stop. To continue, simply enter the STEP command again. If a HARD BREAK is encountered it will show up twice. Once at the instruction where it was placed and once at the following instruction. This is normal and is caused by the way the CPU pre-fetches instructions.

EXAMPLE {DDT8}

Step the program that was entered in the "MA" command description.

```
:PC=2000     {set the program counter}

:DR
A=00 X=00 Y=00 S=E0 M=0A L=0 P=--1-----
00/2000:A9 00       LDA #00     {this instruction will be traced

:ST  {invoke the step command}
A=00 X=00 Y=00 S=E0 M=0A L=0 P=--1---Z-   {A = 00 and Z flag set
00/2002:85 06       STA 06        {next instruction to be stepped
T:  {Press "Return" to trace the STA 06 instruction}
A=00 X=00 Y=00 S=E0 M=0A L=0 P=--1---Z-
00/2004:A9 30       LDA #30
T:
A=30 X=00 Y=00 S=E0 M=0A L=0 P=--1-----   {A = 30 and Z cleared
00/2006:85 07       STA 07
T:
A=30 X=00 Y=00 S=E0 M=0A L=0 P=--1-----
00/2008:A0 01       LDY #01
T:
A=30 X=00 Y=01 S=E0 M=0A L=0 P=--1-----  {Y = 01
00/200A:B1 06       LDA (06),Y {003001  {effective address $3001
T:
A=00 X=00 Y=01 S=E0 M=0A L=0 P=--1---Z-  { $3001 was 00
00/200C:AA          TAX
```

etc.

Note! Remember all numbers input to the ProDev DDT are assumed to be in hexadecimal format. If you wanted to step 10 decimal instructions, you would enter the command "ST !10". The exclamation point must precede all decimal numbers. If you entered the command "ST 10" you would step the next 16 decimal instructions (E.G. 10 = $10 = 16 decimal).

TR [N] - TRACE

The TRACE command is identical to the step command in operation except for the information displayed. The TRACE command does not display the register contents. This is done to allow more program steps to fit on the screen while tracing.

TS A - TRACE WITH SUB

The TRACE WITH SUBROUTINE command is the most flexible of all the ProDev DDT's commands. This command allows you to write your own subroutine that is executed after each instruction of the target program. The target program is stopped when your testing subroutine returns with the Carry flag set. Entering "esc" will cancel the command.

Your subroutine may be placed anywhere in system RAM that is not used by the target program. The location of your subroutine is indicated by "A"

in the command description and is mandatory. Your subroutine lets the ProDev DDT know when you wish to stop the target program by setting the CARRY flag (C=1) of the processor status register. When the ProDev DDT sees that your test subroutine has set the CARRY flag, it stops the target program and gives you control. All of the processor registers are available for use in your subroutine, without fear of affecting the operation of the target program. You should not use memory locations as storage unless you are absolutely sure the target program does not use them. Several examples of test subroutines are given in Chapter 9.

IMPORTANT! Make sure the program counter is properly set before entering the TRACE WITH SUB command. The target program will begin execution at the current location of the program counter as soon as the command is entered.

{DDT16} Bank numbers may be specified for the test subroutine. Your test subroutine is entered via "JSL" and must end with "RTL".

EXAMPLE

Use the TRACE WITH SUB command to determine when memory location $3002 reaches a value of $80, when executing the program described with the Mini-Assembler command.

Use the Mini-Assembler to enter the following test subroutine at memory location $2800.

```
2800-    LDA $3002  ;load the accumulator with location $3002
2803-    CMP #$80   ;has location $3002 reached $80 yet?
2805-    BEQ $280B  ;it is $80, set CARRY and leave
2807-    CLC        ;not $80 yet, clear the CARRY flag
2808-    BCC $280B  ; and return
280A-    SEC        ;yes it is $80 so set the CARRY flag
280B-    RTS        ;return to the ProDev DDT
```

:PC= $2000 {Set the program counter to a good location}

:TS 2800 {Now enter the TRACE WITH SUB command.}
A=00 X=80 Y=01 S=E0 M=0A L=0 P=N-1-----
00/2010:E8 INX

The target program was interrupted when location $3002 reached $80. To verify this, use the MEMORY MODIFY command to examine location $3002.

:MM 3002

00/3002: 80 {memory location $3002 does contain $80}

As you can see the possible uses for this command are practically limitless.

Note! The execution time of the target program will be increased when using this command. This is caused by the added steps required to save all the registers after each instruction of the target program and by the time required to run your subroutine. So don't be too alarmed if it takes a few seconds before the program is stopped.

EX [N] - EXECUTE

The EXECUTE command is very similar to the TRACE command with one obvious exception. The processor registers are displayed only after the last instruction has been executed. For example, if you entered the command "EX 1" it would be the same as entering the TRACE command "TR or TR 1". However, if you enter the command "EX !10", then the next 9 instructions of the program would be executed with nothing being displayed and only after the tenth instruction was executed would the processor registers be displayed. This greatly increases the speed with which the program is executed and makes it more convenient to run through large sections of code.

If you invoke the EXECUTE command with N = 0 (E.G. EX 0) the target program will continue to execute until you stop it. Pressing the pause "KEY" (default is <ctrl>-S) will pause the execution of the program until another key is pressed. While paused, pressing "esc" will stop the program from executing and cancel the EXECUTE command.

This command can be very useful for debugging graphic animation routines. When invoked, the target program will execute at a much slower speed, thereby allowing you to see the animation steps as they proceed and to stop the target program at any point. This can also be useful if you wish to examine the methods used by other programs to perform their animation.

***** SPECIAL NOTE *****

The ProDev DDT was never intended to be used as a tool for copying protected software or to break copy protection codes. Any use of the ProDev DDT for these purposes may be illegal and is not promoted or supported by ProDev in any way.

EXAMPLE

:PC= $2000 {position the program counter}

:EX !10 {execute the next 10 decimal instructions}
A=00 X=01 Y=01 S=E0 M=0A L=0 P=--1-----
00/200D:8E 02 30 STX 3002

T: {Ten instructions were executed and the program was stopped

ER - EXECUTE TO RTS

The EXECUTE TO RTS command will cause the target program to begin executing at the current program counter location and continue until an RTS instruction is found. This can be very useful when you are tracing a program in which you know the subroutines work properly.

IMPORTANT! Make sure the program counter is properly set before entering the EXECUTE TO RTS command. Your program will begin execution at the current location of the program counter as soon as the command is entered.

EXAMPLE

Use the subroutine described in the TRACE WITH SUB command.

:PC= 2800 {set the program counter}

```
:ER
A=00 X=01 Y=01 X=E0 M=0A L=0 P=N-1-----
00/2809:60          RTS
```

T: {the program runs until the RTS is found}

Note! Subroutines may be called from within subroutines, in which case, the ProDev DDT will stop at the 1st RTS/RTL it sees. Simply enter "ER" to continue to the next RTS/RTL.

ET A.A - EXECUTION TIME

The EXECUTION TIME command displays the number of clock cycles required to execute the code between two inclusive addresses. The command activates a hardware device on the ProDev DDT card which actually runs the specified code and counts the number of clock cycles required. The counter has a limit of $FFFF or 65,535. If this limit is exceeded the counter will start over at $0000.

The addresses may not contain a bank number.

{DDT8} The various bank switches will be used to determine the bank of the instructions. In other words, if you enter the command "ET 800.802" and you have set the memory switches to read from AUX memory, the execution time will be performed on the instructions in auxiliary memory from $800 to $802 inclusive.

{DDT16} The current program bank will be used.

This command can be very useful for writing delay loops and for writing time critical code. The output is in the form of the number of clock cycles required to execute the code. To make a fairly accurate conversion to seconds, multiply the number of clock cycles by 0.9775 E -06 (0.0000009775).

{DDT16} This command should only be used with the system clock set to NORMAL (1Mhz). If the system clock is set to FAST then "(x ~2.6) will be displayed along with the number of NORMAL (1Mhz) clock cycles that were required. (x ~2.6) indicates that the number of FAST clock cycles required is approximately 2.6 times the number of NORMAL clock cycles displayed.

* ADVANCED NOTE

The 1Mhz system clock of the Apple computer does not have a constant period of 977.5 nS. The clock cycles are periodically extended by approximately 140nS. These extended clock cycles must be taken into account if you are trying to develop an extremely accurate software time base. The ProDev DDT does not treat these extended clock cycles any differently than the standard clock cycle. There are quite a few good books on Apple hardware which describe the system timing and we will not attempt to do so here.

EXAMPLE

Referring to the program which we previously described in the TRACE WITH SUB command at address $2800.

How many clock cycles are required to execute the LDA $3002 instruction at address $2800?

:ET 2800.2800
CLOCK CYCLES = $0004 {four clock cycles required}

How many clock cycles are required to execute the program from address $2800 inclusive to address $2809 inclusive?

:ET 2800.2809
**CLOCK CYCLES = $0011** {$11 or 17 clock cycles required}

RT - REAL TIME

If the current instruction is a "JSR" then execute the program in real-time until the instruction following the "JSR" is reached. This command is simply a one time version of the T range as described previously in the Protection Window description. The instruction following the "JSR" must reside in RAM so a break can be automatically placed there to halt the program after the real-time execution of the subroutine. This instruction also works with the special case of a ProDOS8 & ProDOS16 MLI call by placing the break beyond the MLI parameters. If you are using a similar technique for parameter passing then the "RT" or "T range" commands should not be used.

{DDT16} Also works with "JSL" "RTL" when in native mode. DO NOT use "RT" or "T" range commands with a JSR when in emulation mode.

EXAMPLE

If the current instruction is:

00/2000:20 DD FB JSR FBDD

Entering the command "RT" will result in the subroutine at $FBDD being executed in real-time. In this case $FBDD is the Apple ROM's ring bell routine. The bell will ring and control will be returned to the ProDev DDT at the instruction following the JSR.

00/2003:EA NOP or whatever.

down - SKIP THE CURRENT INSTRUCTION

Pressing the "down arrow" when in the "Trace" mode will cause the current instruction to be skipped. The word "SKIP" is displayed and the program counter is automatically positioned at the next instruction.

TRACING PROGRAMS IN OTHER PERIPHERAL CARDS

The ProDev DDT will trace programs in other peripheral cards as long as the card does not use a bank switching scheme to fit more than 2K bytes in the 2K ROM space. The best way to find out if it works is to try it and see. The slot number of the other peripheral card should be in memory location $7F8. Enter the ProDev DDT by pressing the button.

TRACING PROGRAMS IN the $C800 - $CFFF MEMORY SPACE.

The $C800 - $CFFF memory space is shared by all the peripheral cards and is reserved for their use (The Apple also uses this space). If you wish to trace a program on a peripheral card which resides in this space it is necessary to take the following steps:

a) Set the memory location $7F8 (MSLOT) to the number $Cn of the card being traced.

b) Use the "GO" command to run a "do nothing" program. (E.G. $2000 JMP $2000).

c) Now press the BUTTON INTERRUPT. This saves the $Cn number of $7F8 (the active peripheral card) to the ProDev DDT's RAM. This is the card the ProDev DDT will access when told to access $C800 - $CFFF memory space.

CHAPTER 7
EXIT

QUit

"QUit" exits the ProDev DDT. The Quit command assumes the ProDev DDT was entered with a "JSR" and exits via an "RTS" instruction. If you entered the ProDev DDT from the system monitor with "Cn00G", entering a Quit command will return you to the system monitor. If any Real breaks remain in the program a warning message, "REAL BRKS IN", will be displayed.

Do NOT use the "QU" command if you have entered the DDT via the "Button". When you enter the DDT via the "Button" no return address is available on the stack for the "QU" command to use.

When you exit via the Quit command, the ProDev DDT remains active and will interpret any breaks that occur.

GO TO MONITOR

--{DDT8}--
**** - GO TO MONITOR**

The command "**" places you in the Apple's monitor where you may use any of the monitor commands. Use <ctrl>Y to reenter the DDT.

--{DDT16}--
**** [I,C] - GO TO MONITOR**

"**" places you in the Apple's monitor where you may use any of the monitor commands. Use <ctrl>Y to reenter the DDT. From the monitor you may enter the control panel to make changes of use CDAs.

"**I" disables interrupts before entering the monitor.

"**C" disables interrupts and enables "Cloaking" before entering the monitor. The ONLY way to reenter the DDT after enabling cloaking is with the "Button". Cloaking does a hardware write protect of the DDT. In its write protected mode the DDT is not vulnerable to program crashes that might otherwise disable it by inadvertently writing to the cards memory space.

The DDT16 uses the][GS vector at $00/03FB. If this 3 byte jump is trashed then pressing the DDT "Button" will do unpredictable things.

"**" command summary.
"**" - Go to monitor, interrupts on, ^Y enabled. {DDT8 or DDT16}
"** I" - Go to monitor, interrupts off, ^Y enabled. (DDT16}
"** C" - Go to monitor, interrupts off, ^Y disabled, Cloaking on. {DDT16}

If any Real breaks remain in the program a warning message, "REAL BRKS IN", will be displayed. This is to remind you not run the program. The breaks would not be interpreted by the ProDev DDT. To clear the Real breaks, use <ctrl>Y or press the "Button" to return to the ProDev DDT and use the "RA" or "RB [A]" command. You may now use "**" to reenter the system monitor.

CHAPTER 8
TALKING TO THE WORLD

PRINTER OUTPUT

With the ProDev DDT it is possible to use your printer as an output device. This can be done to obtain a hard copy of the screen information or to allow you to view graphics on the screen while still being able to debug your program.

The printer interface must be in slot #1 and must support the Pascal v1.1 protocol. Most printer interface and serial cards should work. See the "MO" mode command for selecting printer I/O.

{DDT16 - When using the IIGS printer port, set the baud rate via the control panel settings}.

REMOTE TERMINAL

Using a remote terminal with the ProDev DDT is also possible. The serial interface must be in slot #2 and must support the Pascal v1.1 protocol. Select the desired baud rate by following the instructions supplied with your interface card. If you are using a Super Serial Card or equivalent, use the switches to set the baud rate & data size. We recommend the following switch settings for 9600 baud operation on the SSC:

sw1 D D D U U U U sw2 U U U U D x D

{DDT16 - When using the IIGS serial port, set the baud rate via the control panel settings}.

The ProDev DDT does not support error checking. Operation may be unreliable at very high baud rates. If you experience any communications problems, try using the next lowest baud rate. We have found operation at 9600 baud to be very reliable.

When you are operating the ProDev DDT from a terminal you will not see the windowed display on the terminal screen. The information displayed to the terminal is the same information that would normally appear in the main display window. The Serial I/O mode is activated from the Mode menu (See MO command).

Other computers make great terminals. All you need is a serial interface and some type of terminal software. Most modem software will work. The][GS even has built in terminal emulation that will work. We prefer to use modem software that has a scroll back buffer. This allows you to scan back through instructions that you have already traced in case you missed something.

Use the following commands if you want to use the][GS' built in terminal emulation. From a basic prompt type the following:

```
] IN #2 <cr>      Note! No return on the following lines
<ctrl>AT          this starts terminal mode
<ctrl>A14B  9600 baud
<ctrl>AME         mask line feeds
<ctrl>ABE         buffer enable
<ctrl>AED         echo disable
```

CHAPTER 9
CUSTOMIZING

CUSTOMIZING THE {DDT8}

MEMORY LOCATIONS FOR DDT8

n = Slot number of ProDev DDT card

$Cn00 = Initialize and enter the ProDev DDT.
$Cn03 = Initialize the ProDev DDT and do an RTS.
$Cn24 = Warmstart entry into ProDev DDT.

LOCATIONS INSIDE ProDev DDT RAM.

Where your program's registers are saved.

$C811 = Accumulator
$C813 = X index
$C815 = Y index
$C817 = Stack Pointer
$C81A = Processor Status
$C81B = Program counter (Low byte)
$C81C = Program counter (Hi byte)
$C81D = Bank number

ProDev DDT8 FLAGS

```
$C81E DS 1 ;"Text screen save": 0= off, bit 7=1= on
$C81F DS 1 ;CPU type: 0= 6502, bit 7=1= 65C02, bit 6=1= 65802
$C820 DS 1 ;I/O mode: 0= normal, bit 7=1= serial, bit 6=1= printer
$C821 DS 1 ;bit 7=1= don't send DDT output to screen
```

You can use the following "start up" program to initialize the above FLAGS to your preference. Simply modify the program to provide the setup you desire.

```
*************************************************************
*  ProDev DDT8 setup program                                *
*                                                           *
*  replace "n" with the slot number of the ProDev DDT8 card.*
*************************************************************

DDT8IN2   EQU  $Cn03     ;Enter here to initialize the DDT8
CARDOFF   EQU  $CFFF     ;Disable the peripheral cards 2K ROMs.
TEXTSAVE  EQU  $C81E     ;Text screen save
CPUTYPE   EQU  $C81F     ;CPU, $00=6502, $80=65C02, $40=65802
IOMODE    EQU  $C820     ;I/O, $00=Screen, $80=Serial, $40=Printer
DISPLAY   EQU  $C821     ;DDT8 display On/Off switch, $80 = Off

INITDDT8  BIT  CARDOFF   ;Disable all peripheral cards 2K ROMs
          JSR  DDT8IN2   ;Initialize DDT8 and return.
          BIT  $Cn00     ;Enable DDT8 card.

* Set the Text screen save mode
```

```
          LDA  #0        ;Turn screen save off
          STA  TEXTSAVE  ;Set the flag

* Select the CPU
          LDA  #$80      ;CPU type 6502
          STA  CPUTYPE   ;Set the flag

* Select serial I/O
          STA  IOMODE    ;Set for serial I/O

* Turn the display off. This will prevent the ProDev DDT from
* changing the screen display of the target program, which is nice
* during serial I/O operation.
          STA  DISPLAY   ;Turn display off

* The following two lines have the same effect as leaving the
* ProDev DDT via the "*" command.
          LDA  #$DC      ;Ignore interrupts & write protect DDT8
          STA  $C80C     ;Talk directly to the VIA chip

          RTS            ;Setup routine finished
```

SUBROUTINES FOR "TRACE WITH SUBROUTINE" COMMAND

"Bcc" - refers to any valid branch instruction.

TEST A BYTE IN MEMORY

```
TESTMEM LDA $ADDRESS  ;get contents of location to test
        CMP #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTS           ;return to ProDev DDT
```

TEST THE ACCUMULATOR

```
TESTACC CMP #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTS           ;return to ProDev DDT
```

TEST THE X INDEX

```
TESTX   CPX #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTS           ;return to ProDev DDT
```

TEST THE Y INDEX

```
TESTY   CPY #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTS           ;return to ProDev DDT
```

TEST THE STACK POINTER

```
TESTSP  TSX           ;transfer the stack pointer to X index
        INX           ;
        INX           ;correct for the JSR to your subroutine
        CPX #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTS           ;return to ProDev DDT
```

TEST THE LOW BYTE OF THE PROGRAM COUNTER

```
TESTPC  CLC           ;clear CARRY flag
        LDA $C816     ;get low byte of pc from DDT8 RAM
        CMP #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTS           ;return to ProDev DDT
```

TEST THE PROCESSOR STATUS REGISTER

```
TESTPS  CLC           ;clear CARRY flag
        LDA $C815     ;get the status register contents from DDT8
        CMP #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTS           ;return to ProDev DDT
```

CUSTOMIZING THE {DDT16}

MEMORY LOCATIONS FOR DDT16

```
     n = Slot number of ProDev DDT card

$Cn00 = Initialize and enter the ProDev DDT.
$Cn04 = Initialize the ProDev DDT and do an RTS.
$Cn08 = Warmstart entry into ProDev DDT.
```

LOCATIONS INSIDE ProDev DDT16 RAM.

```
     --- Where your program's registers are saved ---

$C811 DS 2 ;A
$C813 DS 2 ;X
$C815 DS 2 ;Y
$C817 DS 2 ;S
$C819 DS 2 ;D
$C81B DS 1 ;B
$C81C DS 1 ;M
$C81D DS 1 ;Q
$C81E DS 1 ;P
$C81F DS 1 ;PC low byte
$C820 DS 1 ;PC hi byte
$C821 DS 1 ;PBR Program Bank Register

     --- DDT16 flags ---

$C822 DS 1 ;"Text screen save": 0= off, bit 7=1= on
$C823 DS 1 ;CPU type: 0= 6502, bit 7=1= 65C02, bit 6=1= 65816
$C824 DS 1 ;I/O mode: 0= normal, bit 7=1= serial, bit 6=1= printer
$C825 DS 1 ;bit 7=1= don't send DDT output to screen
$C826 DS 1 ;BIT 7=1= normal entry, bit 7=0 do RTS after DDT init
```

You can use the following "start up" program to initialize the above FLAGS to your preference. Simply modify the program to provide the setup you desire.

```
*****************************************************************
*  ProDev DDT16 setup program                                   *
*                                                               *
*  replace "n" with the slot number of the ProDev DDT card.     *
*  Call in emulation mode.                                      *
*****************************************************************

DDT16IN   EQU  $Cn04     ;Enter here to initialize the DDT16
CARDOFF   EQU  $CFFF     ;Disable the peripheral cards 2K ROMs.
TEXTSAVE  EQU  $C822     ;Text screen save
CPUTYPE   EQU  $C823     ;CPU, $00=6502, $80=65C02, $40=65816
IOMODE    EQU  $C824     ;I/O, $00=Screen, $80=Serial, $40=Printer
DISPLAY   EQU  $C825     ;DDT display On/Off switch, $80 = Off

INITDDT   BIT  CARDOFF   ;Disable all peripheral cards 2K ROMs
          JSR  DDT16IN   ;Initialize DDT and return.
          BIT  $Cn00     ;Enable DDT card.
```

```
* Set the "Text screen save" mode
          STZ  TEXTSAVE  ;Turn screen save off

* Select the CPU
          LDA  #$80      ;CPU type 65C02
          STA  CPUTYPE   ;Set the flag

* Select serial I/O
          STA  IOMODE    ;Set for serial I/O

* Turn the display off. This will prevent the ProDev DDT from
* changing the screen display of the target program.
          STA  DISPLAY   ;Turn display off

* The following two lines have the same effect as leaving the
* ProDev DDT via the "**" command.
          LDA  #$DC      ;Ignore interrupts & write protect DDT8
          STA  $C80C     ;Talk directly to the VIA chip
          RTS            ;Setup routine finished
```

SUBROUTINES FOR "TRACE WITH SUBROUTINE" COMMAND

"Bcc" - refers to any valid branch instruction.

The DDT16 calls "TS" routines in 16 bit native mode with the following register contents:

```
      A = targets Accumulator
      X = target's X
      Y = target's Y
      P = target's status with I=1 to prevent interrupts.
      D = $0000
      B = $E0
```

The direct page register "D" & data bank register "B" must be preserved by your subroutine.

TEST A BYTE IN MEMORY

```
TESTMEM LDA $ADDRESS  ;get contents of location to test
        CMP #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTL           ;return to ProDev DDT
```

TEST THE ACCUMULATOR

```
TESTACC CMP #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTL           ;return to ProDev DDT
```

TEST THE X INDEX

```
TESTX   CPX #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTL           ;return to ProDev DDT
```

TEST THE Y INDEX

```
TESTY   CPY #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTL           ;return to ProDev DDT
```

TEST THE STACK POINTER

```
TESTSP  TSX           ;transfer the stack pointer to X index
        INX           ;
        INX           ;correct for the JSR to your subroutine
        CPX #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTL           ;return to ProDev DDT
```

TEST THE LOW BYTE OF THE PROGRAM COUNTER

```
TESTPC  CLC           ;clear CARRY flag
        LDA $C81F     ;get low byte of pc from DDT RAM
        CMP #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTL           ;return to ProDev DDT
```

TEST THE PROCESSOR STATUS REGISTER

```
TESTPS  CLC           ;clear CARRY flag
        LDA $C81E     ;get the status register contents from DDT
        CMP #$value   ;compare to some value
        Bcc STOP      ;the condition was met so stop the program
        CLC           ;condition not met, clear the CARRY flag
        BCC LEAVE     ; and leave.
STOP    SEC           ;set the CARRY flag to stop the program
LEAVE   RTL           ;return to ProDev DDT
```

APPENDIX A
IT DON'T WORK!

NEW INSTALLATION

Go through the installation instructions again and repeat each step.

Check for physical damage to the ProDev DDT board or for IC's which may have become loose during shipment.

Try a different slot in the computer.

Try removing any other peripheral cards from the computer. Some peripheral cards may interfere with the ProDev DDT's operation. Accelerator cards may not be in use at the same time as the ProDev DDT is active.

{DDT8} The Titan Technologies Accelerator //e suffers from an apparent design problem and may not be present in the computer at the same time as the ProDev DDT.

If it still doesn't work, contact ProDev for assistance. (See the last page of this manual).

!!!!!! PLEASE, DO NOT RETURN THE PRODUCT TO ProDev UNLESS WE SPECIFICALLY DIRECT YOU TO DO SO. !!!!!!

IT DID WORK

Sometimes the connectors will get dirty and lose contact. Try removing the card and reinstalling it again. You might also use a pencil eraser to gently clean the gold plated connectors. (Make sure the computer power is off and you follow the procedure for installing as outlined in this manual)

JUST ADDED A NEW PERIPHERAL

Have you just installed a new peripheral card? If so, it may be interfering with the operation of the ProDev DDT. Try removing the newly installed peripheral card.

RESET WHEN DDT IS ACTIVE?

Reseting your computer when the DDT is waiting for input or when you are tracing code will disable the DDT. Reseting while tracing code may result in a "locked up" condition.

If a "Locked up" condition occurs while using the DDT, take the following steps if you wish to avoid powering down your system.

Reset your system & try to start the DDT via $Cn00G.

If the DDT fails to start, repeat the above procedure and this time hold down the "DDT Button" while doing $Cn00G. It may be necessary to repeat the above procedure several times.

If you are unable to restart the DDT then it will be necessary to turn off your computer for at least 15 seconds.

IF SERVICE IS NEEDED

We support what we sell! If service is required we will provide it for a reasonable fee. The Technical Support staff will notify you of our current repair rates and provide you with instructions for shipping your ProDev DDT board. See the last page of this manual for the address and telephone number.

APPENDIX B
GENERAL COMMENTS

{DDT8 65802}

The 65802 mode is not fully supported at this time. The effective addresses of the 65802's unique addressing modes are not all properly calculated for display in the effective address window.

{DDT8 MOUSE}

The ProDev DDT will allow tracing of programs that make use of the AppleMouse // if the mouse is used in passive mode only. We used the sample program that appears in Appendix B of the "AppleMouse // User's Manual" as a general compatibility test. If you encounter any problems please let us know.

STACK

The ProDev DDT generates interrupts and uses the system's host processor. Since the 6502 has only one stack pointer, it is difficult to leave the stack RAM undisturbed. We have gone to great lengths to preserve the integrity of the system stack. In most applications you should not encounter any stack problems. If your program switches between the Main memory stack and the Auxiliary memory stack and/or does manipulation of the stack pointer with the "TXS" instruction you may encounter undesired changes in the stack RAM. The best way to preserve the system stack is to do tracing or executing and avoid real-time running of your program. If you experience problems of this type, please report them as you would any bug. The more information we collect the better our chances of reducing the problem.

BUGS

No, not the cartoon rabbit. Please don't assume that bugs you encounter are so obvious that we must already know about them. You would be amazed at how blind we can be to the obviUOs.

APPENDIX E
ERROR NUMBERS

The following is a table of error numbers:

ERROR #	DEFINITION
01	**Invalid command** - Check the menu for proper command syntax.
02	**Address expected** - This command must be followed by an address (some commands require an address range).
03	**Number expected** - This command expects a number parameter.
04	**Bad command parameter** - This may include missing or incorrect addresses, numbers or symbols.
05	**Illegal bank selected** - An invalid address bank was entered.
10	**No room for additional Breakpoints** - Use "RB A" to remove an old breakpoint.
20	**Command will not work in ROM** - This command tried to write to a location that did not allow a write.
30	**"RT" or "T range" JSL while in emulation mode** - A JSL may not be used as part of an "RT" or "T range" operation while in emulation mode. {DDT16}

TO THE USER

We welcome your comments or suggestions about this product.

Please help stop illegal software pirating. Remember, every dollar of revenue lost to software pirates is made up in higher prices to the rest of us.

TECHNICAL SUPPORT

Please contact us by telephone, U.S. mail or Email if you have any technical questions concerning the operation of this product.

Attn: DDT Tech Support
ProDev, Inc.
P.O. Box 162
LaSalle, MI 48145-0162

(313) 848-4012 voice/fax

24 hour answering machine operation. Leave a very brief description of the problem, your name, phone number, and the best time for us to return your call.

Try calling between 10:00 am & 8:00 pm M-F Eastern Time if you hate talking to machines.

Internet - <info@prodev.biz>

Special thanks to all the people who made this product possible, including our beta testers & loyal customers.