SCJ1271

XDS/22 TMS320C2x Emulator 1988

XDS/22 TMS320C2x Emulator User's Guide

1988

Digital Signal Processor Products

TEXAS INSTRUMENTS
テキサス インスツルメンツ

XDS/22 TMS320C2x Emulator User's Guide

TEXAS INSTRUMENTS
テキサスインスツルメンツ

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes in the devices or the device specifications identified in this publication without notice. TI advises its customers to obtain the latest version of device specifications to verify, before placing orders, that the information being relied upon by the customer is current.

In the absence of written agreement to the contrary, TI assumes no liability for TI applications assistance, customer's product design, or infringement of patents or copyrights of third parties by or arising from use of semiconductor devices described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor devices might be or are used.

Preface

This manual consists of 10 sections and an appendix.

- Sections I through III form a tutorial manual, recommended for first-time readers.
- Sections IV through X form an operation manual. First-time users should read this material before operating the equipment. Experienced users may want to reference it from time to time.
- Appendix A is reference material for all users.

Section content is as follows:

I	Introduction to XDS with some equipment description, manual organization, and associated documentation.
II	Describes set-up for operation with an IBM PC as a terminal.
III	Walk-through demonstration of system capabilities.
IV	Emulator commands including a functional grouping for quick reference, typographical conventions, and detailed descriptions.
V	XDS/22 operation from advanced command entry through program entry to target-system connection and actual emulation.
VI	XDS/22 operation for program-debugging features - trace and hardware and software breakpointing.
VII	How to connect an XDS/22 to various terminals, computers, printers, and PROM programmers. Includes operational details where necessary.
VIII	XDS Multiprocessing Applications.
IX	XDS/22 and plug-in cards hardware description, including repair information and selected engineering drawings.
X	Description of emulator error and warning messages in numerical order for convenient reference.

Contents

Illustrations

Tables

Table *Page*

1. Introduction

This manual describes a Texas Instruments Extended Development Support (XDS) System, Model XDS/22 (Figure 1-1), configured for emulation of a TMS320C2x digital signal processor.[1] Texas Instruments also offers emulator cards for its other digital signal processors, microcomputers, and microprocessors.

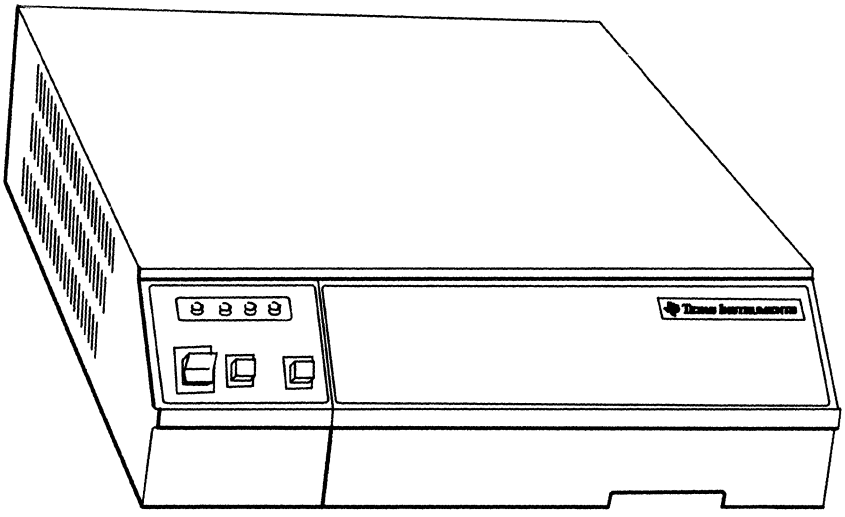


Figure 1-1. Extended Development Support (XDS) System, Model XDS/22

You can use an XDS/22 with any RS-232C terminal or terminal-emulating personal computer[2] as an excellent stand-alone development system, or use the XDS/22 and terminal to tap the power and resources of a micro, mini, or mainframe computer.

You can also connect up to nine XDS units together in either the stand-alone or computer-assisted mode to emulate multiprocessing designs.

---

1 It can also emulate a TMS32020 digital signal processor with the simple change described in Section 9.4.8. Throughout the rest of this manual, references to a TMS320C2x also apply to a TMS32020, unless otherwise stated.

2 Hereinafter referred to simply as "terminal".

1.1 XDS/22 System Block Diagram Discussion

In Figure 1-2, the terminal lets you enter emulator commands, device assembly-language statements and directives, and machine-language instructions into the XDS/22. The XDS/22 executes the commands and instructions and drives the system under development (hereinafter called the target system) just as if it contained the TMS320C2x in the XDS/22.

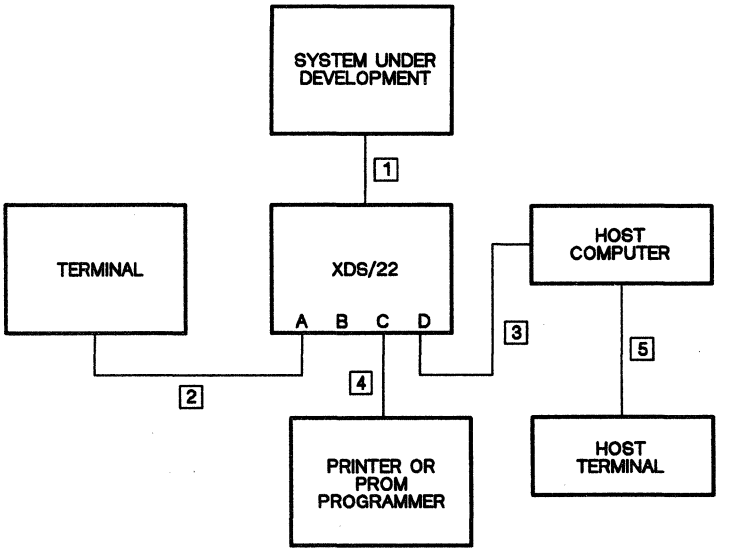


Figure 1-2. Emulation Block Diagram

If your program works exactly as you intended, congratulations! You can then connect a PROM programmer to Port C and immediately convert it to firmware.

But, if your application program doesn't work at all or doesn't work as expected, use the XDS/22 program-debugging features – trace and breakpoint – to see exactly what your program *is* doing, then fix it easily with the XDS/22 memory-change features – display, inspection, and modification.

With tracing, the XDS/22 stores selected program-execution information for later display, with the option to halt processing after storing a selected number. With breakpoint, the XDS/22 halts processing under selected conditions. You can also combine tracing and breakpoint.

If your application requires it, you can connect a host computer to XDS/22 Port D for exchange of programs in assembly or machine language.

During the emulation session, you can connect a printer to log your input and the XDS response.

Cable 1[3] called the target-system cable, connects the TMS320C2x Emulator card with the target system. The target-system connector plugs into the exact socket in your target system where the TMS320C2x device will eventually go.[4]

Cables 2, 3, and 4 are not supplied with your XDS/22; however, Section 7 contains schematics and descriptions. Cable 3 may not be the same for printer and programmer.

Cable 5 and the host terminal are shown for information only. They are *not* XDS system components.

---

[3] Supplied with your XDS/22.

[4] Section 9.4.5 details the target-system cable and connectors.

1.2 XDS/22 Components

An XDS/22 unit (Figure 1-3) consists of an XDS/22 cabinet, Memory Expansion/Communications card, TMS320C2x Emulator card, and a Breakpoint/Trace/Timing card.

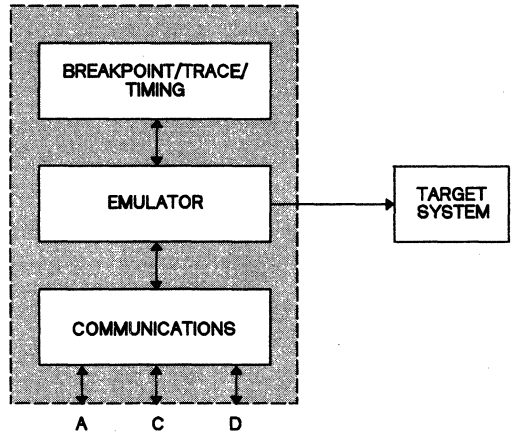


Figure 1-3. XDS/22 Block Diagram

The cabinet houses, powers, and protects the cards and mounts the RS-232C connectors for connection to the terminal and possible host computer.

The Memory Expansion/Communications card contains 64K of random-access memory (RAM) that you can use as read-only memory (ROM) or RAM, just as if it were in your target system. The card also receives all keyboard or host computer input, and generates RS-232C-compatible output signals for the host computer, printer, programmer, and screen display.

The Emulator card contains a Texas Instruments 16-bit microcomputer (SE9996) and associated devices to execute the emulator commands described in Section 4 and control the remaining operations; and a TMS320C2x to execute the appropriate assembly- or machine-language commands.

The Breakpoint/Trace/Timing card stores the machine cycles and memory accesses that you have selected for tracing and can halt the Emulator card after reaching the selected number of traces or upon satisfying the selected number of breakpoint conditions.

The Breakpoint/Trace/Timing card can also analyze your program's performance by displaying your choice of actual execution time or time spent in selected memory locations.

1.3 Manual Description

Each major section of this manual begins with a brief table of contents. Each page contains the subsection title for easy reference.

This manual specifically covers XDS/22 devices with

- Memory Expansion/Communications card, part number 2311050-0002
- Breakpoint/Trace/Timing card, part number 2243660-0002
- TMS320C2x Emulator card, part number 2234640-0001, with firmware revision level 1.2.0 or higher.

It is also generally applicable to units with other configurations and firmware revision levels; however, it does not cover units with a Breakpoint/Trace card installed.

The Preface to this manual describes the content of each Section.

1.4 Associated Documentation

> **Note:** Literature part numbers are *for reference only*. Always contact your Texas Instruments Sales Office to order the correct revision level.

1) TMS32020 Users Guide, Literature Part Number SPRU004B

2) TMS320C2x Users Guide, Literature Part Number SPRU012

2. Getting Started

This Section assumes that you have unpacked the XDS/22 and its Manual and moved it to an operating location that meets the environmental specifications detailed in Table 2-1 on Page 2-2.

This Section also assumes that you are using an IBM PC[5] with the CROSSTALK software package[6] as a terminal. Refer to Section 7 for other applications.

Warning:

Texas Instruments cannot be responsible for damage caused by operating your XDS/22 before reading the instructions in this Section.

Caution:

To avoid XDS damage from electrostatic discharge, be sure to protect target-system connector at all times that it is not plugged into target system.

Topics in this Section include:

5 IBM PC is a trademark of International Business Machines.

6 CROSSTALK is a trademark of Microstuf, Inc.

2.1 XDS/22 Setup

1) Place XDS/22 in its permanent operating station. Be sure to lift the XDS/22 only by grasping it from the sides and bottom. Do not use the front panel, decorative trim, or back-panel connectors as handles.

2) The operating station must have the environmental characteristics listed in Table 2-1.

Table 2-1. Environmental Specifications

Operating Temperature	60°F - 90°F (15°C - 35°C)
Operating Humidity	30% - 80%
Maximum Temperature Rise	12°F/ (6.6°C)/Hour
Storage Temperature	-40°F - +185°F (-30°C - 85°C)
Storage Humidity	5% - 90%

3) Remove card-cage cover by loosening two captive thumbscrews (Figure 2-1), pressing upward and inward on cover, then lifting it out. Be careful not to snag cables.

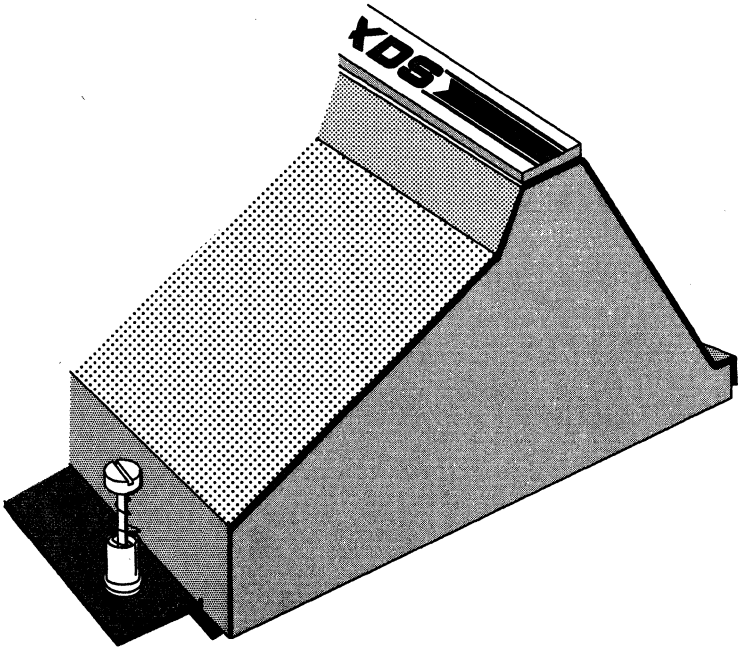


Figure 2-1. XDS/22 Card-Cage Cover, Right Side

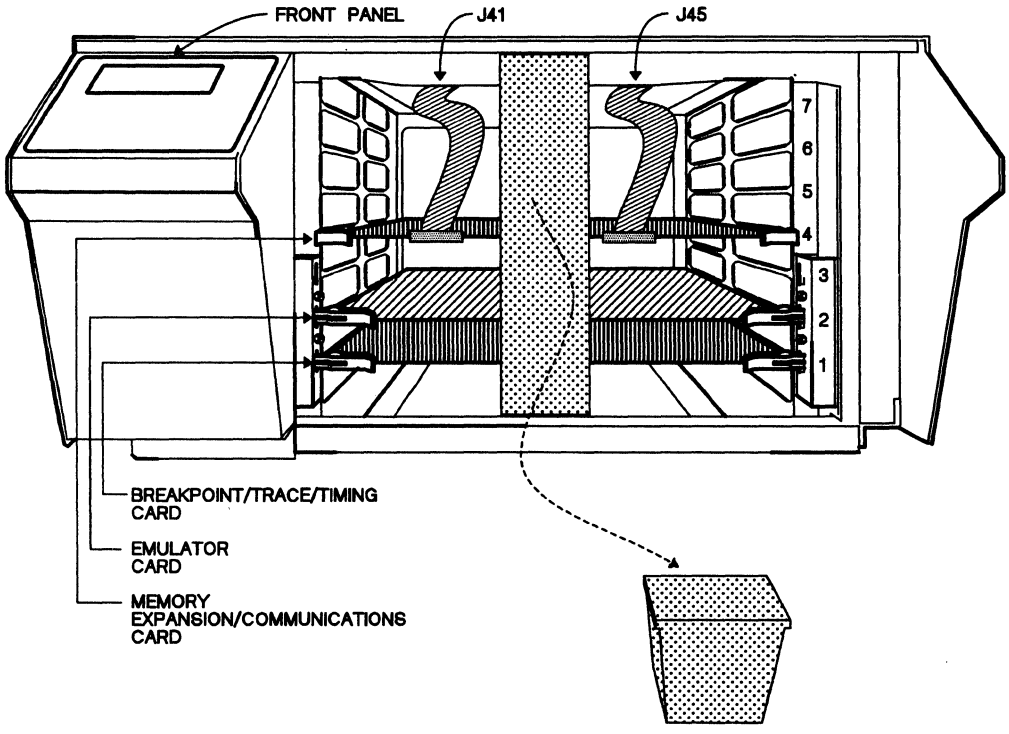


Figure 2-2. XDS/22 Interior

4) Discard shipping retainer and CAUTION tag (Figure 2-2).

5) The TMS320C2x Emulator card comes with eight 4K x 4 RAM installed at card locations U31 through U38 (Section 9.4.1.1 on Page 9-29) which form addresses >0 through >FFFF (0 - 4095) in the target-system program-memory space and addresses >400 through >13FF in the data-memory space. If this is not what you want, refer to that Section for the procedure to change it.

6) Visually check the communications switch settings on the Memory Expansion/Communications card (Table 2-2). S4 is the furthest left, S3 is just visible behind J41.

Table 2-2. Communications Switch Settings for XDS/22 Operation with an IBM PC as a Terminal

SWITCH POSITION	DIP SWITCH	
	S4	S3
1	ON	ON
2	ON	OFF
3	OFF	ON
4	OFF	ON
5	ON	OFF
6	ON	ON
7	OFF	OFF

7) Replace card-cage cover and tighten thumbscrews. Be careful not to snag cables.

Caution: Since the XDS/22 card-cage cover directs cooling air over the cards, keeps outside electromagnetic interference (EMI) away from the cards, and keeps any XDS-generated EMI inside the cabinet, never operate the machine with that cover removed.

2.2 Cabling

Connect the Communications Adapter card[7] in your IBM PC to XDS/22 Port A with the cable shown in Figure 2-3.

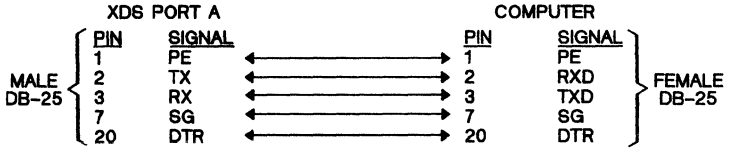


Figure 2-3. Cable, XDS/22 Port A to IBM PC

7 An IBM Printer Adapter will NOT work.

2.3 Power

Table 2-3. XDS/22 AC Requirements

RATED VOLTAGE	RATED FREQUENCY, Hz	RATED CURRENT, AMPERES	NEMA WALL RECEPTACLE
100 V_{ac}	50/60	3	5-15R
115 V_{ac}	60	3	5-15R
220 V_{ac}	50	2	Schuko 1050
240 V_{ac}	50	2	BSI 1363

1) Make sure the local power source (Table 2-3) matches the XDS/22 power requirement shown on its back-panel model plate, then press lower half of POWER switch (OFF - 0) on the XDS/22 Operator Panel (Figure 2-4), and plug XDS/22 into power source.

2) Connect PC to its power source and turn it on. It is desirable that both equipments use the same power outlet.

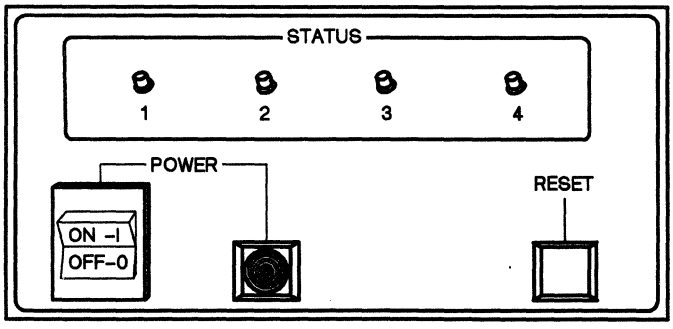


Figure 2-4. XDS/22 Operator Panel

Caution:

Always turn XDS/22 power off before connecting or disconnecting connectors or installing or removing circuit cards.

Remove and install circuit cards and connectors only in a static-free workstation following normal MOS-device static-handling procedures.

2.4 IBM PC Setup

Computer setup largely involves installing a communications or terminal-emulation program that meets the XDS requirements of seven data bits, two stop bits, even parity, and any standard baud rate from 300 to 19,200, 9600 preferred. The XDS monitor program will set any of these requirements that can be set by software.

The following example uses the CROSSTALK program.[8] If you use another program, please refer instead to Section 2.5 on Page 2-11.

2.4.1 First-Time Usage

1) Call the program.

2) When the display of Figure 2-5 appears, follow this procedure.

   a) An entry consists of a command name, a space, and a command entry.

   b) You can enter the entire command name, or just its abbreviation (the two upper-case letters) in upper or lower case. For example, for the first entry, any of these will work:

      - NAME
      - name
      - NA
      - na

   c) The same also applies to a command entry, except that, in many cases, you can further abbreviate a command name to just its first letter, either upper or lower case. The following example uses lower case.

[8] CROSSTALK is a trademark of Microstuf, Inc.

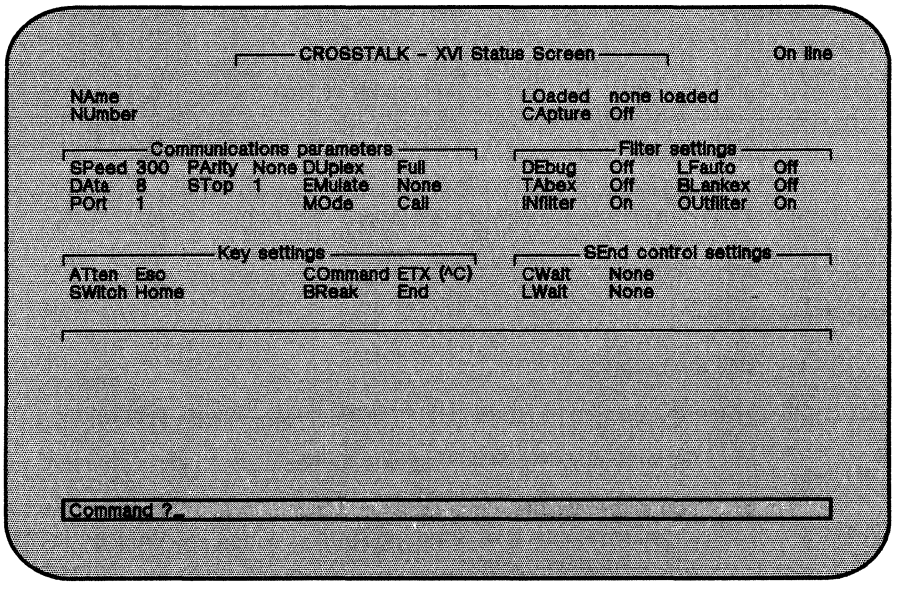


Figure 2-5. First CROSSTALK Screen

DISPLAY	ENTER
Command? [].	
NAme XDS	NAme XDS<CR>
NUmber XDS	NUmber XDS<CR>
SPeed 9600	SPeed 9<CR>
DAta 7	DAta 7<CR>
POrt A	POrt a<CR>
PArity Even	PArity e<CR>
STop 2	STop 2<CR>
EMulate ADDS Vpnt	EMulate a<CR>
NAK (^U)	ATtention ^u<CR>
BReak (DC4 ^T)	BReak ^t<CR>
	ACcept e<CR>

3) At this point, the display changes to Figure 2-6.

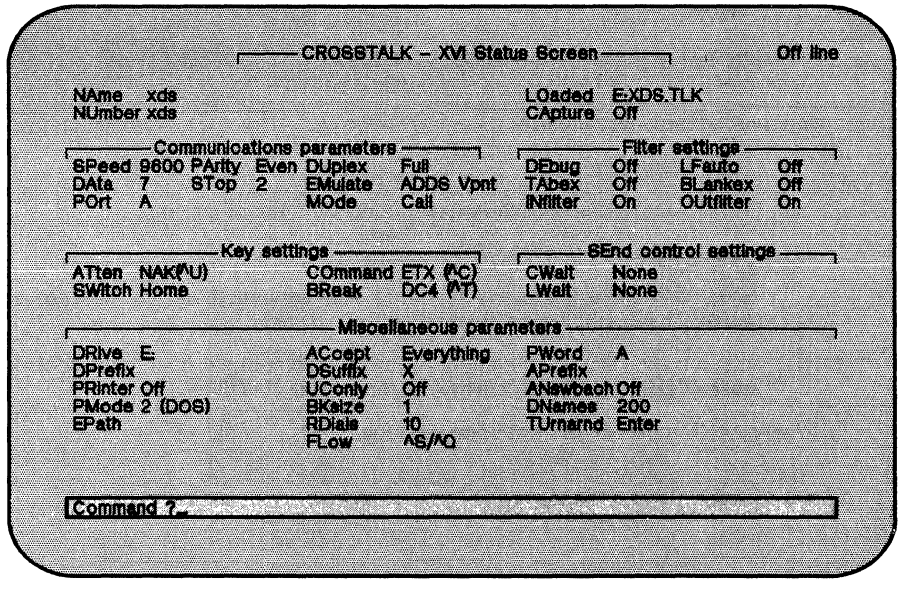


Figure 2-6. Second CROSSTALK Screen

4) Enter:

DISPLAY	ENTER
`ACcept Everything`	PWord a<CR>
`PWord A`	ANswerback o<CR>
`ANswerback Off`	SAve *filename.filetype*<CR>

The SAve command stores all the CROSSTALK parameters, both defaults and the ones you have changed, in a file (named XDS.TLK in this example). You can reload these parameters at any later time as described in Section 2.4.2.

5) When the screen displays `Command`, enter:

`Command? []`	go local XDS<CR>

6) When the reverse-video line displays a local-data message, the computer is ready to communicate with the XDS.

You can also enter the following just before the "go local" entry (or at any time with the Command? prompt displayed):

`Command? []`	fk 1 icc!^Z!!!

then press RETURN. This assigns execution of the XDS ICC (Initialize Cursor) to function-key #1, with ^Z as the cursor-up key (the only combination recognized by the terminal that CROSSTALK emulates). Just press function-key #1 at any time during the session to execute the command.

2.4.2 At any Later Time

You can reload the parameters you changed (and the default value of a number of other parameters that you did not change) at any time.

The procedure is as follows:

1) Call the program.

2) Press Enter to display the reverse-video Command? prompt.

3) Type lo *filename.filetype*<CR>

   where *filename.filetype* is the name you entered after "sa" in the sequence above. The examples in this manual use *xds.tlk*.

4) If the screen displays a dialing message accompanied by a countdown display, press CNTL and U together twice, then enter N<CR>

When the Command? prompt appears, type

go local<CR>.

2.5 Initial Power-Up

1) Turn XDS/22 on by pressing upper half of POWER switch. If the POWER indicator (Figure 2-4) doesn't light, disconnect the XDS/22 from its power source immediately and refer to Section 9.6.

2) Wait at least 3 seconds, then press RETURN on the computer keyboard twice to enable the emulator's Autobaud feature. The screen should display the menu shown in following text. The final ? is the XDS prompt, indicating that the XDS is ready to accept any of the commands displayed on that menu.

```
                    TEXAS INSTRUMENTS
                TMS320C2x XDS VERSION x.x.x

COMMANDS:
   INIT       IPM      IR       RUN       BTT      IT       HOST     IMP
   IPORT      IDM      DR       CRUN      IBTT     FT       IHC      IMD
   IPRM       DPM      MR       SS        DBTT     DT       UL       ID
   ICC        DDM      DIO      RTR                         DL       BGND
   RCC        MPM      MIO
   RESTART    MDM

   LOAD       FILL     DPS      ARM       SSB      DTIME    LOG      GRUN
   SAVE       FIND     DES      DISARM    DSB      XTIME    SNAP     TRUN
   HEX        ITR      DHS      STOP      CSB               HELP     GHALT
   DEC        MAP      DTS                CASB     XA       DV       THALT
   MAG                 DMAP               XRA

VARIABLES:
   PC         AR       ACC      ARP       INTM     ARB      XF       C
   ST         S        T        OV        DP       CNF      FO       HM
              GREG     P        OVM                TC       TXM      FSM
                                                    SXM      PM

   ?
```

> **Note:**
>
> You can redisplay the Help menu (less the ID line) at any time that the screen displays the ? prompt by typing HELP, then pressing RETURN. If the screen is not displaying the ? prompt, press ESC on the terminal keyboard or press RESET on the XDS/22 Operator Panel.

3) If the menu doesn't appear, refer to Section 9.6.2 on Page 9-44.

4) Refer to Section 2.6 for system checkout.

2.6 Checkout

2.6.1 Initialization

1) Type INIT, then press RETURN. The screen displays:

```
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]      = INTERNAL []
```

2) Accept INTERNAL (5 MHz, 200 ns) by pressing RETURN. TMS320C2x emulation also allows 10-MHZ, 100-ns operation by typing 2 or OSC or just O (oh), then pressing RETURN.[9] If you change the default value, the screen displays OSC regardless of what you type.

Selecting 1 with a target system not connected (as in this case) or selecting 2 with a TMS32020 device on the card or with no oscillator installed causes the emulator card to revert to selection 0=INTERNAL.

3) After accepting or selecting a clock, the screen displays:

```
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]      = INTERNAL
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES]   = NO        []
```

The new prompt controls whether the monitor program keeps software breakpoints set after executing them or whether it automatically deletes each breakpoint after executing it. For this checkout, accept the default of NO by pressing RETURN.

4) The screen now displays:

```
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]      = INTERNAL
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES]   = NO
      HEX NOTATION [0=NONE, 1=TI, 2=X, 3=HS, 4=HP]    = NONE      []
```

The new prompt lets you enter decimal numbers or hexadecimal numbers (Section 5.1.7) as parameter values.[10] For this checkout, accept hexadecimal by pressing RETURN.

5) The screen now displays:

```
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]      = INTERNAL
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES]   = NO
      HEX NOTATION [0=NONE, 1=TI, 2=X, 3= HS, 4= HP] = NONE
  BP: NUMBER OF EXTENDED-ADDRESS BITS (0 - 8)         = 0         []
```

6) Press RETURN to reject extended-addressing (Section 6.9).

---

9 This shorthand entry method works for any XDS command parameter presented as a list enclosed in square brackets. The first item in such a list is always number 0, the second number 1, etc. In some long prompts, numbers don't appear, but, with a few exceptions, the monitor program accepts list numbers for any prompt in square brackets.

10 If you select decimal entry, you can also select decimal display as described in Section 5.1.7

7) The screen now displays:

```
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]      = INTERNAL
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES]   = NO
      HEX NOTATION [0=NONE, 1=TI, 2=X, 3= HS, 4= HP] = NONE
  BP: NUMBER OF EXTENDED-ADDRESS BITS (0 - 8)         = 0
 EXP: MAP OVERLAP MODE [0=HIGH/LOW,1=PROGRAM/DATA]    = HIGH      []
```

The new prompt controls mapping of expansion memory (Section 5.4). For this checkout, accept the default of HIGH(/LOW) by pressing RETURN.

8) The screen now displays:

```
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]      = INTERNAL
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES]   = NO
      HEX NOTATION [0=NONE, 1=TI, 2=X, 3= HS, 4= HP] = NONE
  BP: NUMBER OF EXTENDED-ADDRESS BITS (0 - 8)         = 0
 EXP: MAP OVERLAP MODE [0=HIGH/LOW,1=PROGRAM/DATA]    = HIGH
      WAIT FOR TARGET READY [0=NO,1=YES]              = NO        []
```

The new prompt controls whether the emulator responds immediately to commands or waits for the target-system READY signal (Section 5.4.1). For this checkout, accept the default of NO by pressing RETURN.

9) The screen now displays:

```
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]      = INTERNAL
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES]   = NO
      HEX NOTATION [0=NONE, 1=TI, 2=O, 3= HS, 4= HP] = NONE
  BP: NUMBER OF EXTENDED-ADDRESS BITS (0 - 8)         = 0
 EXP: MAP OVERLAP MODE [0=HIGH/LOW,1=PROGRAM/DATA]    = HIGH
      WAIT FOR TARGET READY [0=NO,1=YES]              = NO
      ENABLE BUS-REQUEST MODE [0=NO,1=YES]            = NO        []
```

The new prompt controls memory access (Page 4-75). For this checkout, accept the default of NO by pressing RETURN.

10) The screen now displays:

```
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]      = INTERNAL
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES]   = NO
      HEX NOTATION [0=NONE, 1=TI, 2=O, 3= HS, 4= HP] = NONE
  BP: NUMBER OF EXTENDED-ADDRESS BITS (0 - 8)         = 0
 EXP: MAP OVERLAP MODE [0=HIGH/LOW,1=PROGRAM/DATA]    = HIGH
      WAIT FOR TARGET READY [0=NO,1=YES]              = NO
      ENABLE BUS-REQUEST MODE [0=NO,1=YES]            = NO
?
```

Redisplay of the ? prompt shows that you have completed entry of the INIT command.

2.4.2 Verification

To verify initialization, enter the DES (Display Emulator Status) command:

```
DISPLAY                                                    ENTER
?                                                          DES<CR>
DES

                    CYCLE TIME = 200 NSEC
  AUTOMATIC FIRST WAIT STATE = OFF
                          ARM = OFF


      PORT A      RTS:1      CTS:1      DSR:1    ONLINE
      PORT C      RTS:1      CTS:0      DSR:1    OFFLINE
      PORT D      RTS:1      CTS:1      DSR:1    ONLINE

?
```

> **Note:**
>
> The actual values that appear on your screen may be different. In particular, the cycle time would be 100 ns if you have a TMS320C25 device installed and selected OSC in response to the first INIT-command prompt.

2.4.3 Memory Definition

For this checkout, define memory with the ITR (Initialize Target RAM) command.

With the ? prompt displayed, type ITR:

```
?                                                    ITR<CR>
ITR
 PROGRAM MEMORY RAM [0=OFF,1=ON] = OFF               []
```

Turn program memory on by typing 1, then pressing RETURN. The screen now displays:

```
ITR
 PROGRAM MEMORY RAM [0=OFF,1=ON] = OFF ON
 DATA MEMORY    RAM [0=OFF,1=ON] = OFF []
```

Leave data memory off by just pressing RETURN. The screen now displays:

```
ITR
 PROGRAM MEMORY RAM [0=OFF,1=ON] = OFF ON
 DATA MEMORY    RAM [0=OFF,1=ON] = OFF
?
```

2.7 Emulator Card Checkout Procedure

The short checkout consists of filling memory with a known pattern, then finding and displaying the addresses that contain that pattern.

1) With ? displayed, type FILL, then press RETURN.

2) The screen displays:

```
FILL
 START ADDRESS                    = 0000
```

3) Press RETURN to accept 0000 as the starting address.

4) The screen further displays:

```
FILL
 START ADDRESS = 0000
 END   ADDRESS                    = 0000
```

5) Type 3FF, then press RETURN to set >3FF as the ending address.

6) The screen displays:

```
FILL
 START ADDRESS                    = 0000
 END   ADDRESS                    = 0000 3FF
 DATA                             = 0000
```

7) Type any value, for example AAAA, then press RETURN.

8) The screen displays:

```
FILL
 START ADDRESS                     = 0000
 END   ADDRESS                     = 0000 3FF
 DATA                              = 0000 AAAA
 DESTINATION [0=PROGRAM, 1=DATA] = PROGRAM
```

9) Press RETURN to accept program memory.

10) When the ? prompt returns, type

```
?                                                    DPM<CR>
```

11) The screen displays:

```
DPM
 START ADDRESS = 0000
```

12) Press RETURN to start data-memory display at >0000.

13) The screen displays:

```
DPM
 START ADDRESS = 0000
 END ADDRESS   = 0000 3FF<CR>
```

14) Type 3FF, then press RETURN to end memory display at >03FF.

15) Display begins in the following format:

```
   DPM
    START ADDRESS = 0000
    END ADDRESS   = 0000 3FF

0000=AAAA AAAA AAAA AAAA AAAA AAAA AAAA AAAA .. .. .. .. .. .. .. ..
0008=AAAA AAAA AAAA AAAA AAAA AAAA AAAA AAAA .. .. .. .. .. .. .. ..
      .
      .
      .
03F8=AAAA AAAA AAAA AAAA AAAA AAAA AAAA AAAA .. .. .. .. .. .. .. ..
```

16) The display continues until it displays all the requested memory, scrolling as required, or until you press ESC which both ends the display and ends the command.

This completes the short emulator checkout. If you want more checkout procedures, the next Section contains numerous operation examples; otherwise, you may prefer to skip to Section 4 for either a quick look at the XDS commands (Section 4.1) or for a more detailed description (Section 4.3) or to Section 5 for detailed operating instructions.

3. Example Sessions

This Section contains sample sessions that explain and demonstrate commonly-used TMS320C2x Emulator applications. This section assumes that you have set up and initialized the emulator as described in Section 2.

Texas Instruments suggests that you work through as many of these examples as time permits; however, you will also benefit by working an example before attempting a similar emulation.

For communications applications, refer to Section 7.6. For multiprocessing applications, refer to Section 8.

Topics in the Section include:

3.1 Memory Applications

The XDS monitor program has powerful commands for manipulating emulator memory. You can fill a block of memory with a particular value, display a block of memory, modify memory, find a particular value in memory, and enter processor instructions using the assembler.

3.1.1 Setting/Changing Memory (FILL, IxM, MxM Commands)

The FILL command lets you enter a specific value into a specific data- or program-memory location or range of locations. The IDM and IPM (Inspect Data Memory and Inspect Program Memory) commands let you both see and change, if desired, one memory location per command execution. The MDM and MPM commands do the same, but you can change any number of consecutive memory locations per command execution.

This section gives further examples of the FILL command first described in Section 2.7 and also demonstrates changing individual locations with the MDM (Modify Data Memory) and MPM (Modify Program Memory) commands.

3.1.1.1 FILL Command

```
DISPLAY                                                    ENTER
?                                                          FILL<CR>
FILL
 START ADDRESS                          = 0000             300<CR>
 END ADDRESS                            = 03FF             325<CR>
 DATA                                   = AAAA             1A1A<CR>
 DESTINATION [0=PROGRAM,1=DATA]         = PROGRAM          1<CR>
```

After you enter the last parameter, which displays as DATA, the cursor briefly moves to the right, then the ? prompt reappears.

3.1.1.2 Modify Memory Commands

The IDM (Inspect Data Memory) and IPM (Inspect Program Memory) commands also let you enter data into any selected data- or program-memory location. The monitor displays the current value in the selected location; and, as usual, you press RETURN to keep it, or enter the new value, then press RETURN, to change it.

For example, change address >325 in data memory to >1B1B with the MDM (Modify Data Memory) command:

```
DISPLAY                                                    ENTER
?                                                          MDM<CR>
MDM
 ADDRESS                = 0000                             325<CR>
 DATA                   = 0000                             1B1B<CR>
?
```

Now use the IDM (Inspect Data Memory) to verify the change.

```
DISPLAY                                                    ENTER
?                                                          IDM<CR>
IDM
 ADDRESS                = 0000                             325<CR>
 0325                   = 1B1B                             Q<CR>
```

If you press RETURN or the Space bar, you see the content of location >326 and you can repeat this as often as you like to see locations >327, >328, etc. If you press -, you see 324 and this can also be repeated. To end the display and the command, press ESC.

3.1.2 Memory Display (DDM, DPM Commands)

These commands let you see memory content, but not change it.

To verify the fill, use the DDM (Display Data Memory) command:

```
DISPLAY                                                    ENTER
?                                                          DDM<CR>
DDM
 START ADDRESS = 0000                                      300<CR>
 END ADDRESS   = 0000                                      330<CR>

0300=1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A .. .. .. .. .. .. .. ..
0308=1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A .. .. .. .. .. .. .. ..
0310=1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A .. .. .. .. .. .. .. ..
0318=1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A .. .. .. .. .. .. .. ..
0320=1A1A 1A1A 1A1A 1A1A 1A1A 1A1A xxxx xxxx .. .. .. .. .. .. .. ..
0328=xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx .. .. .. .. .. .. .. ..
0330=xxxx                                     ..
?
```

The memory display begins with the START ADDRESS (displayed on the first line) and continues through the END ADDRESS (displayed on the last line).

The display is eight words per line (last line may be shorter). The first column lists the address of the first word displayed on that line, followed by an equal sign and the content of that address as four hexadecimal digits. The next seven entries are four hexadecimal digits each for the remaining seven words on the line.

The final entry on the line is one display position for each of the eight words. If the value in any byte is between >32 and >80, that is if it is a displayable ASCII character, then that character appears in the position. For any position that does not contain a value in that range, a dot (decimal point) appears in that position. In the example, >1A is not within that range, so all positions contain decimal points. (Since positions >326 - >330 were not FILLed, content may be anything).

3.1.3 Locating Data in Memory (FIND Command)

The monitor-program FIND command is a convenient way to locate data in memory:

```
DISPLAY                                                    ENTER
?                                                          FIND<CR>
FIND
 START ADDRESS              = 0000                         320<CR>
 END ADDRESS                = 0000                         330<CR>
 DATA                       = 0000                         1A1A<CR>
 DATA MASK (ONES ENABLE)    = FFFF                         <CR>
 SOURCE [0=PROGRAM,1=DATA] = PROGRAM                       1<CR>
 ?
```

The data mask lets you define "don't-care" bits. Refer to Section 5.1.12 on Page 5-13 for a detailed explanation. For this example, press RETURN to accept FFFF, which means only the exact data entry will be found.

Now respond to the last prompt with 1<CR> to search data memory. The screen displays:

```
 00320=1A1A
 00321=1A1A
 00322=1A1A
 00323=1A1A
 00324=1A1A
?
```

In this example, locations from >0320 to >0324 contain data that matches.

Now, reenter the FIND command and press RETURN to accept all the values, except enter 1010<CR> for the data mask. This time the display includes location 325 (and may possibly include other locations).

Now, enter >5A5A in locations >326 through >335:

```
DISPLAY                                                    ENTER
?                                                          FILL<CR>
FILL
 START ADDRESS                        = 0300               326<CR>
 END ADDRESS                          = 0325               335<CR>
 DATA                                 = 1A1A               5A5A<CR>
 DESTINATION [0=PROGRAM,1=DATA]       = DATA               <CR>
```

Since a preceding example executed the FILL command, calling it again displays the parameters entered during that execution as the current values. You accept or change them as shown.

Verify the new values in memory by executing the DDM command again:

```
?                                                          DDM<CR>
DDM
 START ADDRESS                        = 0300               <CR>
 END ADDRESS                          = 0330               <CR>
0300=1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A .. .. .. .. .. .. .. ..
0308=1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A .. .. .. .. .. .. .. ..
0310=1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A .. .. .. .. .. .. .. ..
0318=1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 1A1A .. .. .. .. .. .. .. ..
0320=1A1A 1A1A 1A1A 1A1A 1A1A 1A1A 5A5A 5A5A .. .. .. .. .. .. ZZ ZZ
0328=5A5A 5A5A 5A5A 5A5A 5A5A 5A5A 5A5A 5A5A ZZ ZZ ZZ ZZ ZZ ZZ ZZ ZZ
0330=5A5A                                     ZZ
```

You can also execute a short version of the display/enter sequence as follows:

```
DISPLAY                                                    ENTER
?                                                          FIND<CR>
FIND
 START ADDRESS                        = 0320               <CR>
 END ADDRESS                          = 0330               <CR>
 DATA                                 = 1A1A               <CR>
 DATA MASK (ONES ENABLE)              = 1010               FFFF<CR>
 SOURCE [0=PROGRAM, 1=DATA]           = DATA               <CR>
    0320=1A1A
    0321=1A1A
    0322=1A1A
    0323=1A1A
    0324=1A1A
```

If the example had defined the data mask with a first digit (first four bits) set to 0 (don't care), all values within the specified range of addresses would display as follows:

```
DISPLAY                                                    ENTER
?                                                          FIND<CR>
FIND
 START ADDRESS                  = 0320                     <CR>
 END ADDRESS                    = 0330                     <CR>
 DATA                           = 1A1A                     <CR>
 DATA MASK (ONES ENABLE)        = FFFF                     OF0F<CR>
 SOURCE [0=PROGRAM,1=DATA] = DATA<CR>
      0320=1A1A
      0321=1A1A
      0322=1A1A
      0323=1A1A
      0324=1A1A
      0325=1A1A
      0326=5A5A
      0327=5A5A
      0328=5A5A
      0329=5A5A
      032A=5A5A
      032B=5A5A
      032C=5A5A
      032D=5A5A
      032F=5A5A
      0330=5A5A
?
```

3.2 Assembler and Reverse-Assembler Applications

You normally enter programs into the XDS emulator with the XA (Execute Assembler) command. This lets you use the emulator assembly language, which is similar to, but not quite the same as, regular TMS320C2x Assembly Language.

After program entry, you can use the XRA (Execute Reverse Assembly) command to see entries that might have produced the object code. Reverse assembly does not include comments, directives, or labels.

3.2.1 XA Command

The XA command displays prompts for starting address and whether or not you want to keep an existing symbol table, then sets up source-code entry into the emulator assembler.

The program developed in this topic stores sequential values from -10 to +10 in on-chip block B1 starting at >300. If you have not already done so, you must first turn emulator memory on with the ITR command.

```
DISPLAY                                                    ENTER
?                                                          ITR<CR>
ITR
 PROGRAM MEMORY [0=OFF,1=ON]          = OFF                1<CR>
 DATA MEMORY    [0=OFF,1=ON]          = OFF                <CR>
?                                                          XA<CR>
XA
  START ADDRESS                       = 0000               20<CR>
  USE OLD SYMBOL TABLE (0=NO, 1=YES) = NO                  <CR>
```

The screen displays:

```
LINE   ADD    DATA   LABEL  MNEM     OPERANDS   COMMENT
0001   0020          []
```

Press the Space bar several times, or press CTRL and T together to move the cursor under the MNEM heading, then type IDT. Now press the Space bar several times or press CTRL and T together one time to move the cursor under the OPERANDS heading, then type /EXAMPLE/ and press RETURN.

You have defined the module EXAMPLE. The screen displays:

```
LINE   ADD    DATA   LABEL  MNEM     OPERANDS   COMMENT

0001   0020          []     IDT      /EXAMPLE/
0002   0020
```

Type INIT, space the cursor to under MNEM and type EQU, space it to under OPERANDS and type $, then press RETURN.

The screen displays:

```
LINE   ADD    DATA   LABEL  MNEM     OPERANDS   COMMENT

0001   0020          []     IDT      /EXAMPLE/
0002   0020          INIT   EQU      $
0003   0020          []
```

Space the cursor under MNEM and type LRLK, then space it under OPERANDS and type AR2,-10, then space it under COMMENTS and type INIT start, then press RETURN.

Further presentation is in the same manner. The command fills in the appropriate operation code for the mnemonic you have just entered under the DATA heading, then displays a line with line number and address, with the cursor at the first character position in the label field.

You enter a label, if required or desired; mnemonic; operands, if required; and a comment, if desired. You can also enter an entire line as a comment by entering an asterisk at the first label character position.

The rest of the program follows.

```
LINE    ADD     DATA    LABEL   MNEM      OPERANDS  COMMENT
0003    0020    D200            LRLK      AR2,-10   INIT start[11]
        0021    FFF6
0004    0022    D100            LRLK      AR1,>300  INIT starting pointer
        0023    0300
0005    0024    D000            LRLK      AR0,>314  INIT ending point
        0025    0314
0006    0026    5589            LARP      AR1       ; Save values in AR1
0007    0027            *
0008    0027            LOOP    EQU       $
0009    0027    72AA            SAR       AR2,*+,AR2Save value
0010    0028    55A9            MAR       *+,AR1    Increment value
0011    0029    CE52            CMPR      >2        if not done
0012    002A    F880            BBZ       LOOP      then repeat loop
        002B    0007
0013    002C    FF80    DONE    B         DONE 1    Finished
        002D    0000
        002D    >000C RESOLVED
0014    002E                    END

ERRORS    WARNINGS   UNRESOLVED

 0000      0000       0000

?
```

3.2.2 XRA Command

To display the reverse-assembled version of the program just entered, use the XRA (Execute Reverse Assembler) command:

DISPLAY

```
?
XRA
 START ADDRESS    = 0000 20<CR>
 LINES OF OUTPUT = 0000 9<CR>
   0020   D200   LRLK          AR2, >FFF6
   0022   D100   LRLK          AR1, >300
   0024   D000   LRLK          AR0, >314
   0026   5589   LARP          *,AR1
   0027   72AA   SAR           AR2,*+,AR2
   0028   55A9   MAR           *+,AR1
   0029   CE52   CMPR          >2
   002A   F880   BBZ           >0007,*
   002C   FF80   B             >000C,*
                 END
   ?
```

ENTER

```
XRA<CR>
```

[11] Comments actually appear in upper case just like the mnemonics; however this manual displays them in upper/lower case in regular font for legibility.

3.3 Other Program-Verification Methods

3.3.1 DPM Command

You could also verify the program with the DPM command:

```
DISPLAY                                                    ENTER
?                                                          DPM<CR>
DPM
 START ADDRESS = 0000                                      <CR>
 END ADDRESS   = 0000                                      D<CR>

0000=D200 FFF6 D100 0300 D000 0314 5589 72AA .. .. .. .. .. .. U. r.
0008=55A9 CE52 F880 0007 FF80 0000           U. R. .. .. .. ..
```

3.3.2 IPM Command

You can also display, enter or modify programs with the IPM (Inspect Program Memory) command. described in this section.

```
DISPLAY                                                    ENTER

?                                                          IPM<CR>
IPM
 ADDRESS = 0000                                            <CR>
  0000=D200                                                <CR>
 +0001=FFF6                                                <CR>
 +0002=D100                                                <CR>
 +0003=0300                                                 Q<CR>
```

To exit from the IPM command, type Q, then press RETURN, or press ESC. In either case, any value entered is changed.

For either display, use the IPM command to correct any errors. Just specify the address of the error and re-enter the correct value. Do this now, as the program must be correct in order for the remainder of the walkthrough to execute properly.

3.4 Register Applications

The emulator has commands that let you access the internal processor registers: Accumulator (ACC), auxiliary registers (AR0 - AR_X), program counter (PC), product register (P), stack registers (S_X), global-memory register (GREG), status registers (ST0, ST1), and temporary register (T).

3.4.1 Single Register Control

You can display the current value of any of these registers by simply typing the capitalized abbreviation for the register name with the ? prompt displayed. For example,

DISPLAY	ENTER
? PC=xxxx ?	PC <CR>

You can also change the value of any register by entering its name, an equal sign, and the new value. For example,

DISPLAY	ENTER
? ? PC=A ?	PC=A <CR> PC <CR>

3.4.2 Multiple Register Control

The IR (Inspect Register) and MR (Modify Register) commands each let you display the current content of each register and accept that value or change it. The difference is that the MR command also acts as a sort of register-value Master Reset; that is, executing the command with no parameters sets all the emulator registers to the values associated with the *last* execution of the command.

For example, set the PC by using the MR command:

DISPLAY	ENTER
? MR PC = 0000 20<CR> ST0 = 2604 <CR> ST1 = 47F0 <CR> T = 1234 <CR> P = 00569023 <CR> ACC = 00000001 <CR> ?	MR <CR>

Now set the PC to some other value with the IR command:

DISPLAY	ENTER
? IR PC = 0000 ST0 = E600 ST1 = 47F0 T = 1234 P = 00569023 ACC = 00000001 ?	IR <CR> 1<CR> Q<CR>

Verify register settings with the DR (Display Registers) command:

DISPLAY	ENTER
? DR PC =0001 T =1234 ST0=2604 P =00569023 ST1=47F0 ACC=00000001 ?	DR<CR>

Now enter:

DISPLAY	ENTER
? DR PC =0020 T =1234 ST0=2604 P =00569023 ST1=47F0 ACC=00000001 ?	MR;DR<CR>

to execute the MR command with no display of parameters, then display the registers. And notice that the PC value is again 0020.

Now, finally, enter:

DISPLAY	ENTER
? DR PC =0020 T =1234 ST0=2604 P =00569023 ST1=47F0 ACC=00000001 ?	IR;DR<CR>

and notice that the PC value is still 20; that is, the IR command did *not* change its value.

Another way to change multiple register values is to use command strings, for example:

```
PC=000A,ACC=2,AR0=11<CR>
```

3.5 Single-Step Program Execution

With the registers set, you can execute the program one instruction at a time with the SS (Single Step) command or several instructions at a time with the SS command and various repetition methods. Refer to Section 3.6 for an example of how to run the entire program.

3.5.1 One Instruction at a Time

To single-step the program:

```
DISPLAY                                                    ENTER

?                                                          SS<CR>
 PC= 0022 ST0= xxxx ST1=xxxx  NEXT: 0022 D100 LRLK AR1,>0300
```

3.5.2 Several Steps at a Time

You can re-execute the SS command by pressing <SP> as follows:

```
DISPLAY                                                    ENTER

?                                                          <SP>
SS
 PC= 0024 ST0= xxxx ST1=xxxx  NEXT: 0024 D000 LRLK AR0,>0314

?                                                          <SP>
SS
 PC= 0026 ST0= xxxx ST1=xxxx  NEXT: 0026 5589 MAR  *+,AR1

?                                                          <SP>
SS
 PC= 0027 ST0= xxxx ST1=xxxx  NEXT: 0027 72AA SAR  AR2,*+,AR2
```

A faster way is to enter *<nnnn> after the SS command name. Do this now, for nnnn = 5:

```
DISPLAY                                                    ENTER

?                                                          SS*5<CR>
SS*5
SS
  PC= 0028  ST0= xxxx ST1 = xxxx  NEXT: 0028 55A9 MAR  *+,AR!
SS
  PC= 0029  ST0= xxxx ST1 = xqxx  NEXT: 0029 CE52 CMPR >2
SS
  PC= 002A  ST0= xxxx ST1 = xxxx  NEXT: 002A F880 BBZ  >0027,*
SS
  PC= 0027  ST0= xxxx ST1 = xxxx  NEXT: 0027 72AA SAR  AR2,*+,AR2
SS
  PC= 0028  ST0= xxxx ST1 = xxxx  NEXT: 0028 55A9 MAR  *+,AR1
?
```

The program stops after the fifth execution.

3.6 Continuous Program Execution

3.6.1 The RUN Command

To run the program continuously, set the PC to >20, then enter the RUN command. You can do that with one command string as follows:

DISPLAY	ENTER
?	MR;RUN<CR>
RUN	
RUNNING	

Since this program has nothing to stop it, it continues to run with that display until you stop it by pressing any key or pressing RESET on the XDS/22 Operator Panel. If you press a key, you get this display:

```
KEY
PC=xxx ST0=xxxx ST1=xxxx
```

The XDS CRUN command (Page 4-22) also works.

3.6.2 Setting a Software Breakpoint

Your emulator offers a powerful program debugging tool called software breakpoint. What it does is stop program execution when your program reaches an address that you select with the SSB (Set Software Breakpoint) command. Let's set one at >0007.

DISPLAY	ENTER
?	SSB<CR>
SSB	
BREAKPOINT ADDRESS = xxxx	27<CR>
?	

Now let's start program execution at >20 again by entering:

DISPLAY	ENTER
?	MR;RUN<CR>
RUN	
RUNNING	
SBP	
PC=0027 ST0=xxxx ST1=xxxx	
?	

Now program execution stops at >27 with a display of SBP for software breakpoint.

You can also set up to nine more breakpoints by repeating the SSB command and entering a different address each time.

You can clear software breakpoints individually with the CSB (Clear Software Breakpoint) command as follows:

DISPLAY	ENTER
?	CSB<CR>
CSB	
ADDRESS = xxxx 27<CR>	
?	

You would also have to repeat this command for each breakpoint address. You can also clear all software breakpoints with the CASB (Clear all Software Breakpoints) command as follows:

DISPLAY	ENTER
? CASB ?	CASB<CR>

3.7 Breakpoint/Trace/Timing Examples

This example develops a Breakpoint/Trace/Timing sequence to examine execution and timing of the program entered in Section 3.2.

3.7.1 XDS Initialization

Initialize the XDS as follows:

```
DISPLAY                                                          ENTER
?                                                                INIT
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]     = INTERNAL  <CR>
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES] = NO        <CR>
      HEX NOTATION [0=NONE, 1=TI, 2=X, 3=HS, 4=HP]  = NONE      <CR>
  BP: NUMBER OF EXTENDED ADDRESS BITS (0-8)         = 0         <CR>
 EXP: MAP OVERLAP MODE [0=HIGH/LOW,1=PROGRAM/DATA]  = HIGH      <CR>
      WAIT FOR TARGET READY [0=NO,1=YES]            = NO        <CR>
      ENABLE BUS-REQUEST MODE [0=NO,1=YES]          = NO        <CR>
```

3.7.2 BTT Initialization

This example uses all power-up defaults for BTT operation; therefore, it is not necessary to execute the IBTT command.

3.7.3 BTT Command

Use the BTT command to specify the conditions for the features you selected (by default) with the IBTT command.

```
?                                                                BTT
BTT
 SELECT [S0, S1, S2, S3, ALL, COUNT, TIME]          = ALL        <CR>

STATE 0: BREAKPOINT COUNTER
 EVENT COUNT (0-FFFF)                               = 0001       <CR>

STATE 0: BREAKPOINT
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF       IAQ<CR>
 ADDRESS #1                                         = 0000       2C<CR>
 ADDRESS #2                                         = 0000       2C<CR>
 ADDRESS MASK (ONES ENABLE)                         = FFFF       Q<CR>

STATE 0: TRACE
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF       1<CR>
 ADDRESS #1                                         = 0000       <CR>
 ADDRESS #2                                         = 0000       FFFF<CR>
 ADDRESS MASK (ONES ENABLE)                         = FFFF       Q<CR>

GLOBAL COUNT VALUES
 SEQUENCE COUNT (1-FFFF)                            = 0001       Q<CR>
?
```

You have set the Breakpoint/Trace/Timing card to cause a hardware breakpoint on the first IAQ at address 000C, and trace all cycles from address 0000 to >FFFF.

3.7.4 DBTT Command

After programming the BTT card states, use the DBTT command to see the status of each state as follows:

```
?                                                    DBTT<CR>
DBTT

XDS TIMING AND ANALYSIS VERSION x.x
STATE 0: START AND STOP
    BP (END SEQ)  IF  0001[IAQ & INCL AD(002C-002C)]
    TRACE ON              [ALL & INCL AD(0000-FFFF)]
STATE 1:
    ***INACTIVE
STATE 2:
    ***INACTIVE
STATE 3: STOP
    ***INACTIVE
SEQUENCE COUNT=0001   DELAY COUNT=0000    TRACE COUNT=0000
NOTE: TMF CANNOT OCCUR
?
```

The DBTT command reminds you that a trace-memory-full (TMF) halt cannot occur because you left the trace count at 0 (infinite) by not executing the IBTT command. If the program had been such that a breakpoint halt condition was impossible, a HBP CANNOT OCCUR message would display. If you selected time-out operation, you also get the message STOP ON TIME (SEC ... NS).

3.7.5 Program Execution

Now run the program and check the trace-display results. Use the MR command to set the Program Counter (PC) to >20.

```
?                                                    MR<CR>
MR
 PC  = 0000                                          <CR>
 STO = 0000                                          Q<CR>
?                                                    MR;RUN<CR>
MR
RUN
  RUNNING
  HBP
     CYCLE QUAL EXTQUALS ADDR DATA RVRS ASSEMBLY
     *EVT  PR   11111111 002C FF80 B    >000C,*
?
```

The display shows that after a period of running, the Breakpoint/Trace/Timing card halted the Emulator card with a hardware breakpoint at address >000C.

3.7.6 Trace Display

You can display the trace buffer with either the IT or DT command. This example uses the IT command.

Power-up default trace-buffer format is NORMAL: Trace-sample index, BTT state where sample taken, sample cycle type, processor qualifier, interrupt line, $\overline{\text{BIO}}$ line, and the reverse-assembled instruction.

You further check program timing by redefining the trace function with the IBTT command. Trace time displays include:

Time	Show time trace stored.
Delta	Show time between traces.
Mark	Shows trace times before and after selected trace.

3.7.6.1 Normal Trace Display

You can display the trace buffer with either the IT or the DT command. The example uses the IT command to show some of the samples.

Power-up default trace-buffer format is NORMAL: Trace-sample index, BTT state where sample taken, sample cycle type, processor qualifier, interrupt line, $\overline{BIO}$ line, and the reverse-assembled instruction.

Note that a "L" tagged to the BTT state indicates that this is the last sample taken within that state. This occurs only if the breakpoint event and the trace events happen together.

```
?
IT
 FIRST SAMPLE (0-7FE, 0=OLDEST SAMPLE) = 000                <CR>

INDEX   S CYCLE QUAL   EXTQUALS   ADDR   DATA   RVRS ASSEMBLY

0000    0       IA     11111111   ----   ----
0001    0       IAQ    11111111   0020   D200   LRLK AR2,>FFF6
0002    0       PR     11111111   0021   FFF6
0003    0       IAQ    11111111   0022   D100   LRLK AR1,>0300
0004    0       PR     11111111   0023   0300
0005    0       IAQ    11111111   0024   D000   LRLK AR0,>0314
0006    0       PR     11111111   0025   0314
0007    0       IAQ    11111111   0026   5589   MAR  *,AR1
0008    0       IAQ    11111111   0027   72AA   SAR  AR2,*+,AR2
0009    0       IAQ    11111111   0028   55A9   MAR  *+,AR1
0010    0       IAQ    11111111   0029   CE52   CMPR >2
0011    0       IAQ    11111111   002A   F880   BBZ  >0007,*
0012    0       PR     11111111   002B   0007
0013    0       PR     11111111   002C   FF80
0014    0       IAQ    11111111   0027   72AA   SAR  AR2,*+,AR2
0015    0       IAQ    11111111   0028   55A9   MAR  *+,AR1
0016    0       IAQ    11111111   0029   CE52   CMPR >2
0017    0       IAQ    11111111   002A   F880   BBZ  >0007,*
0018    0       PR     11111111   002B   0007
[]
```

A TMS320C25 uses the first IA cycle to return to the RUN mode. If you use this program with a TMS32020 device, your index 0000 will be the LRLK instruction at address 0000.

You can see more traces by pressing RETURN or end the command by typing Q (RETURN not necessary) or pressing ESC.

3.7.6.2 Trace TIME Display

The Time trace-time option displays each sample with the time that it occurred.

```
                ?                                                                IBTT
                IBTT

                 INITIALIZE [0=OPTION, 1=BTT, 2=TRACE]           = OPTION      2<CR>
                 TRACE DISPLAY [0=NORM, 1=TIME, 2=DELTA, 3=MARK] = NORM        1<CR>

                ?                                                                IT
                IT

                 FIRST SAMPLE (0-7FE, 0=OLDEST SAMPLE) = 000                     <CR>
INDEX    S  CYCLE  QUAL   HRS  MI  SEC  MS   US   NS   ADDR   DATA   RVRS ASSY

[]       Q
?
```

3.7.6.3 Trace DELTA Display

The Delta mode lets you see the time difference between each trace sample taken.

```
                ?
                IBTT
                 INITIALIZE [0=OPTION, 1=BTT, 2=TRACE]           = TRACE       <CR>
                 TRACE DISPLAY [0=NORM, 1=TIME, 2=DELTA, 3=MARK] = TIME        2<CR>
                ?                                                              IT<CR>
                IT
                 FIRST SAMPLE (0-7FE, 0=OLDEST SAMPLE) = 000                   <CR>

INDEX    S  CYCLE  QUAL   HRS  MI  SEC  MS   US   NS   ADDR   DATA   RVRS ASSY

0000     0              IA   ... ..   ..   ...  ...  400    ----   ----
0000     S0             IAQ  ... ..   ..   ...  ...  200    0020   D200          LRLK
         AR2,>FFF6
0001     0              PR   +.....   ..   ...  ...  400    0021   FFF6
0002     0              IAQ  +.....   ..   ...  ...  400    0022   D100          LRLK
         AR1,>0300
0003     0              PR   +.....   ..   ...  ...  400    0023   0300
0004     0              IAQ  +.....   ..   ...  ...  400    0024   D000          LRLK
         AR0,>0314
0005     0              PR   +.....   ..   ...  ...  400    0025   0314
0006     0              IAQ  +.....   ..   ...  ...  400    0026   5589 MAR  *,AR1
0007     0              IAQ  +.....   ..   ...  ...  400    0027   72AA      SAR
         AR2,*+,AR2
0008     0              IAQ  +.....   ..   ...  ...  400    0028   55A9 MAR  *+,AR1
0009     0              IAQ  +.....   ..   ...  ...  400    0029   CE52 CMPR >2
0010     0              IAQ  +.....   ..   ...  ...  400    002A   F880 BBZ  >0007,*
0011     0              PR   +.....   ..   ...  ...  400    002B   07
0012     0              IAQ  +.....   ..   ...  ...  400    002C   72AA      SAR
         AR2,*+,AR2
0013     0              IAQ  +.....   ..   ...  ...  400    0027   55A9 MAR  *+,AR1
0014     0              IAQ  +.....   ..   ...  ...  400    0028   CE52 CMPR >2
0015     0              IAQ  +.....   ..   ...  ...  400    0029   F880 BBZ  >0007,*
0016     0              PR   +.....   ..   ...  ...  400    002A   0007
0017     0              IAQ  +.....   ..   ...  ...  400    002B   72AA      SAR
         AR2,*+,AR2
[]       Q
?
```

3.7.6.4 Trace MARK Display

The Mark mode lets you set any one trace sample as t_0, then display all other traces with their signed time differential from that trace: - for those occurring before, + for those occurring after.

This command first gives a display exactly like the Time option. You then select the trace that you want to be t_0, enter its number, press RETURN, and get the desired display.

```
?                                                                    IBTT
IBTT
 INITIALIZE [0=OPTION, 1=BTT, 2=TRACE]              = TRACE          <CR>
 TRACE DISPLAY [0=NORM, 1=TIME, 2=DELTA, 3=MARK] = DELTA             3<CR>

?                                                                    IT
IT
 FIRST SAMPLE (0-7FE, 0=OLDEST SAMPLE) = 000                         <CR>
```

```
INDEX   S  CYCLE  QUAL   HRS  MI  SEC  MS   US   NS   ADDR  DATA    RVRS ASSY
0000    0        IA     ...  ..  ..   ...  ...  400  ----  ----
0000    S0       IAQ    ...  ..  ..   ...  ...  600  0020  D200    LRLK AR2,>FFF6
0001    0        PR     +... ..  ..   ...  ..1  000  0021  FFF6
0002    0        IAQ    +... ..  ..   ...  ..1  400  0022  D100    LRLK AR1,>0300
0003    0        PR     +... ..  ..   ...  ..1  800  0023  0300
0004    0        IAQ    +... ..  ..   ...  ..2  200  0024  D000    LRLK AR0,>0314
0005    0        PR     +... ..  ..   ...  ..2  600  0025  0314
0006    0        IAQ    +... ..  ..   ...  ..3  000  0026  5589    MAR  *,AR1
0007    0        IAQ    +... ..  ..   ...  ..3  400  0027  72AA    SAR  AR2,*+,AR2
0008    0        IAQ    +... ..  ..   ...  ..3  800  0028  55A9    MAR  *+,AR1
0009    0        IAQ    +... ..  ..   ...  ..4  200  0029  CE52    CMPR >2
0010    0        IAQ    +... ..  ..   ...  ..4  600  002A  F880    BBZ  >0007,*
0011    0        PR     +... ..  ..   ...  ..5  000  002B  07
0012    0        IAQ    +... ..  ..   ...  ..5  400  002C  72AA    SAR  AR2,*+,AR2
0013    0        IAQ    +... ..  ..   ...  ..5  800  0027  55A9    MAR  *+,AR1
0014    0        IAQ    +... ..  ..   ...  ..6  200  0028  CE52    CMPR >2
0015    0        IAQ    +... ..  ..   ...  ..6  600  0029  F880    BBZ  >0007,*
0016    0        PR     +... ..  ..   ...  ..7  000  002A  0007
0017    0        IAQ    +... ..  ..   ...  ..7  400  002B  72AA    SAR  AR2,*+,AR2

[]
```

Assume that you want index number 10 to be t_0. Type M10 at the block cursor, then press RETURN. The screen displays:

```
INDEX   S  CYCLE  QUAL   HRS  MI  SEC  MS   US   NS   ADDR  DATA    RVRS ASSY

0000    0        IA  -  ...  ..  ....  ..3  800  ----  ----
0000    S0       IAQ -  ...  ..  ..    ...  ..3  600  0020  D200    LRLK AR2,>FFF6
0001    0        PR  -  +... ..  ..    ...  ..3  200  0021  FFF6
0002    0        IAQ -  +... ..  ..    ...  ..2  800  0022  D100    LRLK AR1,>0300
0003    0        PR  -  +... ..  ..    ...  ..2  400  0023  0300
0004    0        IAQ -  +... ..  ..    ...  ..2  000  0024  D000    LRLK AR0,>0314
0005    0        PR  -  +... ..  ..    ...  ..1  600  0025  0314
0006    0        IAQ -  +... ..  ..    ...  ..1  200  0026  5589    MAR  *,AR1
0007    0        IAQ -  +... ..  ..    ...  ...  800  0027  72AA    SAR  AR2,*+,AR2
0008    0        IAQ -  +... ..  ..    ...  ...  400  0028  55A9    MAR  *+,AR1
0009    0        IAQ -  +... ..  ..    ...  ...  ...  0029  CE52    CMPR >2
0010    0        IAQ 0  +... ..  ..    ...  ...  400  002A  F880    BBZ  >0007,*
0011    0        PR  +  +... ..  ..    ...  ...  800  002B  07
0012    0        IAQ +  +... ..  ..    ...  ..1  400  002C  72AA    SAR  AR2,*+,AR2
0013    0        IAQ +  +... ..  ..    ...  ..1  800  0027  55A9    MAR  *+,AR1
0014    0        IAQ +  +... ..  ..    ...  ..2  200  0028  CE52    CMPR >2
0015    0        IAQ +  +... ..  ..    ...  ..2  600  0029  F880    BBZ  >0007,*
0016    0        PR  +  +... ..  ..    ...  ..2  000  002A  0007
0017    0        IAQ +  +... ..  ..    ...  ..3  200  002B  72AA    SAR  AR2,*+,AR2

[]      Q
```

3.7.7 Performance Analysis

Another valuable Breakpoint/Trace/Timing card feature is performance analysis. The card offers:

- The DTIME command to display the time recorded by the range or point timers that you set with the IBTT and BTT commands
- Trace time-stamping to display traces with accumulated, relative, or offset time information
- The XTIME command to show total time and percentage of usage for ranges that you define.

3.7.7.1 DTIME Command

Use the DTIME command to display timer information. Average time is total time divided by the number of timer starts and stops.

The time resolution of the BTT board depends on the processor cycle time.

```
?
DTIME

                  HRS MI SEC MS  US  NS
   TIMER 1:       ... .. .. ... .12 000
   TIMER 1 AVG:   ... .. .. ... ..1 200
   TIMER 2:       ... .. .. ... ... ...
```

3.7.7.2 XTIME Command

Use the XTIME command to see where your program spent its time.

First set the program counter to the beginning of the example program, then configure the XTIME command to analyze from the start address to the end address. Selecting the IAQ prompt shows only instruction-execution time.

```
?
PC=000
XTIME
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF    1<CR>
 START ADDRESS                                       = 0000   <CR>
 END ADDRESS                                         = 0000   F<CR>
 ADDRESS INCREMENT (0=ONLY 1 SAMPLE)                 = 0000   2<CR>
 SAMPLE TIME (0=NORM--7FFF)                          = 0000   <CR>

  ADDR RANGE  AVG: MI SEC MS  US  NS  PERCENT
  0000-0002     ... .. .. ... ... ...  0.0% .
  0003-0005     ... .. .. ... ... ...  0.0% .
  0006-0008     ... .. .. ... ... ...  0.0% .
  0009-000B     ... .. .. ... ... ...  0.0% .
  000C-000E     ... .. .. ... ... 200  100.0% ******************
  000F-0011     ... .. .. ... ... ...  0.0% .
                                TOTAL=100.0%
```

Now change the XTIME prompt to D and see the data-memory access.

```
?
PC=F100

  ?
XTIME
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF    1<CR>
 START ADDRESS                                       = 0000   <CR>
 END ADDRESS                                         = F      <CR>
 ADDRESS INCREMENT (0=ONLY 1 SAMPLE)                 = 0      2<CR>
 SAMPLE TIME (0=NORM--7FFF)                          = 0      <CR>

   ADDR RANGE  AVG: MI SEC MS  US  NS  PERCENT
   0000-0002     ... .. .. ... ... ...  0.0% .
   0003-0005     ... .. .. ... ... ...  0.0% .
   0006-0008     ... .. .. ... ... ...  0.0% .
   0009-000B     ... .. .. ... ... ...  0.0% .
   000C-000E     ... .. .. ... ... ...  0.0% .
   000F-0011     ... .. .. ... ... ...  0.0% .
                                TOTAL=  0.0%
```

Pagination

4. Emulator Commands

This section contains a command quick-reference, a list of typographical conventions, and a description of each TMS320C2x Emulator command listed in alphabetical order.

You can enter each command in the Prompt mode described in previous parts of this manual (type command name, press RETURN, accept or change each parameter's current value, command executes after last parameter) or you can use the methods described in Section 5.1.

4.1 Emulator Command Functional Grouping

This Section groups emulator commands by function. Table 4-1 on Page 4-3 is a brief summary listing for the experienced user. The remaining sections cover the functional groups in more detail.

For a detailed command listing in alphabetical order, refer to Section 4.3.

4.1.1 Summary

Table 4-1. Emulator Commands Grouped by Function

Initialization		Mode	
DEC	Decimal Format	ARM	Initiate Alternate-Run Mode
HEX	Hexadecimal Format	BGND	Initiate Background Mode
ICC	Initialize Cursor Control	DISARM	End Alternate-Run Mode
IHC	Initialize Host Control	HOST	Initiate Host Mode
IMD	Initialize Multiprocessing	IMP	Initiate Multiprocessing Mode
INIT	Initialize Emulation	#n	Select Emulator #n
IPORT	Initialize Parameters	**Hardware Breakpoint/Trace**	
IPRM	Initialize Parameters	BTT	Set BTT Parameters
ITR	Initialize Target RAM	DBTT	Show BTT Parameters
LOAD	Load Stored Parameters	DT	Display Trace
MAG	Magnitude Format	DTIME	Show Time Settings
MAP	Map Expansion Memory	FT	Find Trace
RCC	Restore Cursor Controls	IBTT	Initialize BTT
RESTART	Restart	IT	Inspect Trace
SNAP	Freeze Display	XTIME	Time Analysis
Memory		**Software Breakpoint**	
DDM	Display Data Memory	CASB	Clear Software Breakpoints
DPM	Display Program Memory	CSB	Clear one Software Breakpoint
FILL	Fill Memory with Data	DSB	Display all Breakpoints
FIND	Find Data in Memory	SSB	Set one Breakpoint
IDM	Inspect Data Memory	**Assembler**	
IPM	Inspect Program Memory	XA	Execute Assembler
MDM	Modify Data Memory	XRA	Reverse Assembler
MPM	Modify Program Memory		
Communications		**Status**	
DIO	Display I/O	DES	Display Emulator Status
MIO	Modify I/O	DHS	Display Halt Status
Offload		DMAP	Display Memory Map
DL	Download	DPS	Display Processor Status
UL	Upload	DTS	Display Trace Status
		/n	Display Emulator MP Status
Run		**Miscellaneous**	
CRUN	Continue Run	DV	Display Values
GHALT	Group Halt (MP Mode)	HELP	Display Command Menu
GRUN	Group Run (MP Mode)	ID	Display Firmware Revision
RTR	Run on Target Reset	LOG	Print Commands and Response
RUN	Continuous Execution	SAVE	Save Parameters
SS	Single-Step Execution	**Registers**	
STOP	Stop Alternate-Run Mode	DR	Display Registers
THALT	Total Halt (MP Mode)	IR	Inspect Registers
TRUN	Total Run (MP Mode)	MR	Modify Registers

4.1.2 Initialization Commands

Initialization commands set display format, cursor-control keystrokes, memory conditions, and load and save parameters.

DEC Decimal Format. Following command shows parameter values in decimal if decimal selected with INIT command.

HEX Hexadecimal Format. Following command shows parameter values in hexadecimal.

ICC Initialize Cursor Control. Selects keystrokes to move cursor up, down, left, and right.

IHC Initialize Host Control. Selects control characters to meet host-computer requirements.

IMD Initialize Multiprocessing. Selects independent, master, or slave status for each emulator in microprocessing application.

INIT Initialize Emulation. Selects clock source, automatic software-breakpoint deletion, numeric mode, extended-addressing, map-overlap mode, wait for target ready, and bus-request mode.

IPORT Initialize Port. Selects parameters for emulator Port C or Port D.

IPRM Initialize Parameters. Resets command power-up parameters which become default values at *next* command execution.

ITR Initialize onboard target RAM. Lets emulator target data and program memory work.

LOAD Loads parameters saved with SAVE command. Does not load programs.

MAG Magnitude Format. Following command shows parameter values in magnitude (unsigned decimal, with sign bit converted as most-significant bit) if decimal selected with INIT command.

MAP Map Expansion Memory. Selects type for expansion memory.

RCC Restore Cursor Controls. Sets cursor-control keystrokes to default values: + (up), - (down), < (left), > (right).

RESTART Software restart. Equivalent to pressing RESET on Operator Panel.

SNAP Freeze Display. Causes last 22 lines on screen to "freeze", that is, parameters displayed on those lines remain the same, but their "current" values change as your program executes.

4.1.3 Memory Commands

Memory commands let you display and change data and program memory.

DDM Display Data Memory. Hexadecimal and ASCII display of selected data-memory location(s) content.

DPM Display Program Memory. Hexadecimal and ASCII display of selected program-memory location(s) content.

FILL Fill Memory with Data. Fill selected data- or program-memory location(s) with selected value.

FIND Find Data in Memory. Find selected value in selected data- or program-memory location(s).

IDM Inspect Data Memory. Inspect or change one selected data-memory location and consecutive adjacent locations.

IPM Inspect Program Memory. Inspect or change one selected program-memory location and consecutive adjacent locations.

MDM Modify Data Memory. Inspect or change selected data-memory location.

MPM Modify Program Memory. Inspect or change selected program-memory location.

4.1.4 Communications Commands

DIO Display I/O. Display target-system I/O ports.

MIO Modify I/O. Change target-system I/O ports.

4.1.5 Offload Commands

Offload commands let you transfer data between the XDS and a host computer or PROM Programmer.

DL Download. Transfers object code from host computer attached to XDS into emulator memory or PROM programmer attached to XDS.

UL Upload. Transfers object code from XDS or target memory to host computer or programmer attached to XDS.

4.1.6 Run Commands

Run commands let you start and stop emulator processing in single or multiprocessing applications.

CRUN Continue Run. Restarts program with trace buffer intact after halt.

GHALT Group Halt (MP Mode). Causes all emulators configured as group in multiprocessing application to halt.

GRUN Group Run. Causes all emulators configured as group in multiprocessing application to start running.

RTR Run on Target Reset. Causes emulator to start execution only after target-system reset.

RUN Start Program Execution. Starts program running after clearing trace buffer, if applicable. In multiprocessing application, independent emulators start immediately, group emulators start when all are ready.

SS Single-Step Execution. Executes instruction at program-counter location, then shows results and reverse-assembly of content of next location.

STOP Stop Execution (ARM Mode). Stops execution in Alternate-Run mode.

THALT Total Halt (MP Mode). Stops all emulators in multiprocessing application. Independent emulators stop after execution of instruction in progress; group stops together.

TRUN Total Run (MP Mode). Starts emulators in multiprocessing application. Independent emulators start immediately; group starts when all are ready.

4.1.7 Register Commands

Register commands let you see or change the content of the processor registers.

DR Display Registers. Display only of content of selected registers.

IR Inspect Registers. Display or modify content of selected registers.

MR Modify Registers. Display or modify content of selected registers.

4.1.8 Mode Commands

Mode commands establish basic emulator operating considerations.

ARM Initiate Alternate-Run Mode.

BGND Initiate Background Mode. Sets emulator currently in foreground into background without bringing any other emulator into foreground.

DISARM Disarm. Ends Alternate-Run Mode after execution of STOP command.

HOST Initiate Host Mode. Match host and emulator parameters for data transfer.

IMP Initiate Multiprocessing Mode.

#n Select Emulator #n. Brings emulator #n into foreground and sets emulator currently in foreground into background.

4.1.9 Hardware Breakpoint/Trace Commands

Hardware breakpoint and trace commands help in program debugging by letting you store selected machine cycles, with or without stopping emulation.

BTT Set BTT Parameters. Sets parameters for resources selected with IBTT command.

DBTT Display BTT Parameters. Shows parameters selected with BTT and IBTT commands.

DT Display Trace. Shows content of trace buffer in format selected with IBTT command.

DTIME Display Time Settings. Shows how long BTT timers ran.

FT Find Trace. Finds selected trace and displays trace buffer in its immediate area.

IBTT Initialize BTT. Select states, resources, and trace display for BTT operation.

IT Inspect Trace. Shows content of trace buffer in format selected with IBTT command.

XTIME Display Execution Time. Runs your program, then shows time spent in each selected address increment.

4.1.10 Software Breakpoint Commands

Software breakpoint commands let you stop program execution at up to 10 selected addresses and see the program-counter value and status-registers content at the time of the halt.

CASB Clear all Software Breakpoints.

CSB Clear Software Breakpoint at selected address.

DSB Display All Software Breakpoints. Display all addresses set with SSB command(s).

SSB Set one Software Breakpoint. Set address for program to halt upon reaching. Repeat up to 10 times.

4.1.11 Assembler Commands

XA Execute Assembler. Selects starting address and old symbol-table retention for emulator assembler.

XRA Reverse Assembler. Shows assembler mnemonics (no labels or symbols) that might have generated values in selected program-memory range.

4.1.12 Status Commands

Status commands let you inspect various emulation aspects at any time.

DES Display Emulator Status. Shows clock period, communication parameters, etc.

DHS Display Halt Status. Shows what caused halt, and program-counter and status-registers content at halt.

DMAP Display Memory Map. Shows memory map.

DPS Display Processor Status. Shows program-counter and status-registers content at halt.

DTS Display Trace Status. Shows state, number of events remaining, trace count, etc.

/n Display Emulator MP Status. Shows status of emulator #n.

4.1.13 Miscellaneous Commands

Miscellanous commands are useful emulator commands that fit none of the previously-described categories.

DV Display Values. Receives two decimal, hexadecimal, or mixed numbers, and shows algebraic sum and absolute value of difference.

HELP Command List. Shows list of Emulator commands.

ID Firmware Revision. In single-emulator operation, shows emulator-card firmware-revision level. In multiprocessing, shows emulator's place in multiprocessing chain.

LOG Print. Starts or stops printer attached to XDS.

SAVE Save. Saves parameter values and alternate buffer in emulator-card non-volatile memory.

4.2 Typographical Conventions

This manual uses the following typographical conventions for command entry:

1) Lower-case text enclosed in angle brackets <> is something that you type in from the keyboard, such as <value> which means type in a value.

2) Upper-case text enclosed in angle brackets means press a terminal key, such as <ESC>, which means press the key marked ESC, <SP>, which means press the Space bar, <CR>, which means press RETURN, or <TERM>, which means press any of the terminators found in Table 5-1 on Page 5-10.

3) A list enclosed in square brackets lets you enter any item from the list by simply typing its list number (remember that the list starts at 0 and that some prompt displays do not include the list number for all items), then pressing RETURN. You can also type in the whole list item, or, if it is a word, just its first letter.

   For example, if a list item is 0=PROGRAM, you can enter any of 0 (zero), P, or PROGRAM.

4) <CNTL/*x*> means press CNTL and *x* together, where *x* is some alphabetic key. Another abbreviation used in tables is ^x.

5) The only time that you can enter a decimal number in the hexadecimal mode is within the XA command.

6) When entering a hexadecimal number in the decimal mode, you must precede the hexadecimal number with one of >, X, or H, or follow it with H, with no intervening space.

7) When entering a hexadecimal number in the XA command, you must precede the number with one of >, 0X, or 0H, or precede it with 0 and follow it with H, with no intervening space.

4.3 Command Descriptions

The command-description format is

Syntax How to enter the command in Prompt mode (type command name, press RETURN) or Parenthesis mode (Section 5.1.3.3).

Description
What the command does.

Parameters
Describes each command parameter.

Associated commands
Names commands that work with subject command.

Example(s)
Examples of command entry and sample displays.

> **Note:**
>
> In hexadecimal mode, all command entry and display is in hexadecimal numbers, with no special notation. In decimal mode, all command entry is assumed to be *decimal* numbers, however, the monitor program converts all addresses, data values, etc. *to* and displays them *as* hexadecimal numbers. You can change the display to decimal, magnitude, or leave it as hexadecimal as described generally in Section 5.1.7 and specifically with each command.

> **Note:**
>
> Reproduction of XDS screen displays in this manual may differ slightly from an actual screen display.

Table 4-2 is a table of contents for the command section.

Table 4-2. Command Table of Contents

Table 4-2. Command Table of Contents (Concluded)

Syntax

Prompt Mode : ARM<TERM>
Parenthesis Mode : ARM(<value>)<TERM>

Description

The ARM command enables the Alternate-Run Mode, which lets the emulator continue running after hardware breakpoints, keyboard halts, or trace-memory-full interrupts.[12] The emulator displays the halt status (refer to the DHS command, Page 4-29) and RUNNING to indicate that the processor is still running.

The monitor program suspends all breakpoint and trace functions and returns to the command entry mode so that you can inspect traces and execute other commands.

Between execution of the ARM command and execution of any of the Run-type commands (RUN, GRUN, or TRUN), you can execute any command; however, once the processor starts, you can only enter the commands in the table below.

With ARM enabled, only a STOP command, software breakpoint, or memory-parity error can stop TMS320C25 program execution.

To cancel the alternate-run mode, wait until the processor halts for a breakpoint or trace condition, or press any key, then type STOP DISARM. In either case, your program continues to run with the monitor in the command-entry mode. You can then enter either the CRUN (Continue Run) or RUN command to change to the normal RUN mode.

Parameters

Memory-Commands-Permitted Parameter: Value of 0 (default) means you cannot enter any memory commands with the emulator running in Alternate-Run mode. Value of 1 means you can enter the commands listed in the first two columns of the table below.

COMMAND AVAILABILITY IN ALTERNATE-RUN MODE

ALWAYS AVAILABLE			AVAILABLE WHEN MEMORY COMMANDS PERMITTED		NEVER AVAILABLE		
ARM	HELP	RUN	DDM	IPM	ACC‡	FSM¶	PC‡
BGND	IBTT	SAVE	DIO	MDM	AR(n)‡	GREG‡	PM¶
BTT	ICC	SNAP	DPM	MPM	ARB¶	HM¶	RTR
CRUN	ID	SOR	FILL	XA	ARP§	HOST	S(n)
DBTT	IHC	STOP	IDM	XRA	C¶	INIT	SS
DES	IMD	THALT			CASB	INTM§	SSB
DHS	IPORT	XTIME			CSB	IR	SXM¶
DISARM	IT				CNF¶	MAP	T‡
DPS†	LOAD				DL	MR	TC¶
FT	LOG				DP§	OV§	TXM¶
GHALT	RCC				DR	OVM§	UL
GRUN	RESTART				FO¶	P‡	XF¶

†No values displayed while running.
‡Register.
§Value in Status-Register 0 (ST0)
¶Value in Status-Register 1 (ST1)

[12] A TMS32020 also continues running after software breakpoints, but a TMS320C25 halts.

> **Caution:**
>
> **Be aware that execution of memory commands in the AVAILABLE WHEN MEMORY COMMANDS PERMITTED group can alter program conditions and may affect program execution. In addition, execution of these commands requires that the processor halt during each memory access, which, in turn, affects program execution time.**

Associated Commands DISARM, Page 4-31

Syntax Prompt Mode : BGND<TERM>

Description **Use this command only in Multiprocessor Mode.** The BGND command places the foreground emulator in the multiprocessor chain into background operation to let that emulator operate off-line. In background operation, the emulator still responds to group commands.

To place another emulator into the foreground, type #n<CR>, where n is the number assigned at chain initialization.

To display the status of any emulator in the chain, type /n<CR>, where, again, n is the number assigned at chain initialization.

Parameters None

Associated Commands IMP (Initialize Multiprocessor Mode), Page 4-74

Example

DISPLAY			ENTER
?			BGND<CR>
BACKGROUND	or	AUTOPOLLING	(Display response depends
??		???	on the state of the EMU when IMP command was executed.)

Syntax

Prompt Mode : BTT<TERM>
Parenthesis Mode : Not available

Description

This command requires that you have the Breakpoint/Trace/Timing card installed. The BTT command lets you select breakpoint, jump, timer start/stop, and trace parameters for the options you selected with the IBTT command.

Parameters

The initial BTT prompt

```
SELECT [S0, S1, S2, S3, ALL, COUNT, TIME] = ALL
```

lets you select:

- An individual state (only the prompts for that state appear)
- All states (prompts appear for each state selected with the IBTT command (Page 4-63)
- Just the sequence (global) counts
- A time-out function (sets maximum emulator run time).

A detailed explanation of each follows:

S0,S1,S2,S3 Enter 0 to select state 0 for display or change, 1 to select state 1, etc. You can use only the state that you select.

4=ALL This is the default value and lets you access all BTT states and the global counters. You can use the primitive function keys: !, @, and # as follows:

- ! Use ! (usually SHIFT and 1 together) to go to the next option within a prompt.
- @ Use @ (usually SHIFT and 2 together) to go to the next state.
- # Use # (usually SHIFT and 3 together) to save and end the BTT command.

Refer to Section 6.2 for further information.

5=COUNT Selecting 5 lets you display only the global counters so that you can view or change the individual counters without having to go through all the other BTT command prompts.

6=TIME Selecting 6 lets you specify a maximum amount of emulator run time. This function is independent of the state timers within the BTT command.

> **Note:** The following example/displays assume that you have selected TRIX and EXTERNAL QUALS with the IBTT (Initialize Breakpoint/Trace/Timing) card, Page 4-63; that all BTT functions qualify both address and data; and that you selected only one function per state. The displays show power-up values.

Prompts for the Breakpoint Function

```
STATE 0: BREAKPOINT COUNTER
 EVENT COUNT (0-FFFF) = 0001

STATE 0: BREAKPOINT
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF
 ADDRESS #1                                          = 0000
 ADDRESS #2                                          = 0000
 ADDRESS MASK (ONES ENABLE)                          = FFFF
 ADDR TYPE [INCL, EXCL, SNGL]                        = INCL
 DATA #1                                             = 0000
 DATA #2                                             = 0000
 DATA MASK (ONES ENABLE)                             = 0000
 DATA TYPE [INCL, EXCL, SNGL]                        = INCL
 EXTERNAL PROBES DATA VALUE                          = 00
 EXTERNAL PROBES DATA MASK (ONES ENABLE)             = 00
```

Event Count This prompt defines the number of breakpoint events that must occur before sequencing to the next state. Each state containing one or more breakpoint functions contains one event count. Any qualified breakpoint within that state can decrement the event counter. When the count reaches 0, the BTT card sequences to the next state.

Qualifier The QUALifier lets you select a memory-cycle operation to qualify an event if it accesses your selected address(es). To turn the breakpoint counter off, select OFF (0). To select any memory cycle, select ALL (1).

The remaining material lets you build a very specific qualification with just a few keystrokes. The colons separate that material into groups, with the abbreviation of the access space (what gets accessed) before the parenthesis and the abbreviation of the access type inside the parenthesis.

The access-space abbreviations are:

P	Program memory
D	Data memory
IO	I/O access
I	Internal cycle

and the access-type abbreviations are:

A	All
IAQ	Instruction Acquisition
R	Read
W	Write

If you simply enter an access-space abbreviation (P, D, IO, or I), you qualify all its access types, for example, entering "P" is the same as entering Program-Memory Read, Program-Memory Write, and Program-Memory instruction acquisition as qualifiers.

On the other hand, if you enter an access type, you qualify that access type regardless of the accessed space; for example, entering "R" is the same as entering Program-Memory Read, Data-Memory Read, and IO Read as qualifiers. You don't get anything from the I group because it doesn't have an R.

You can also enter various combinations, for example, entering PRW is the same as entering Program-Memory Read and Program-Memory Write.

And you can also pick and choose from the various groupings by entering one or more items separated by your choice of a colon,

slash, or ampersand separation character. For example, PAQ/IOR qualifies Program-Memory instruction acquisition and IO read.

Address Address(es) for qualification. For a range, enter the beginning address as #1, the ending address as #2. For a single address, enter it as *both* #1 and #2; because the monitor program checks both. For two independent addresses, enter the first to be checked as #1, the second as #2.

In decimal mode, you can enter values as decimal numbers, but the monitor program converts them to and displays them as hexadecimal numbers.

Address Mask

The address mask lets you set the exact address bits to be included or excluded within the event qualification and also lets you turn a single value into a range of values. Each mask bit set to logic 1 lets only the exact corresponding address bit enter the breakpoint event criteria, each bit set to logic 0 lets either value of that bit enter. Refer to Section 5.1.12 on Page 5-12.

Address Type

This prompt lets you select an inclusive range, exclusive range, single address, or two independent single addresses.

With an inclusive range, 0=INCL, the BTT card checks each memory location starting at Address #1 and continuing through Address #2 for event qualification.

With an exclusive range, 1=EXCL, the BTT card checks all memory locations, *except* the locations from Address #1 to Address #2 (but including Address #1 and Address #2), for event qualification.

With a single address, or with two independent addresses, 2=SNGL, the program checks address #1, then checks address #2.

External Probes Data and External Probes Data Mask :

These prompts appear only if you enabled the TRIX or EXTERNAL DATA prompts with the IBTT command. If you enabled external probes, the BTT card uses the value on the external cable probes (as refined by the external-probe data mask) as part of event qualification.

Prompts for the Trace Function

```
STATE 0: TRACE
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF
 ADDRESS #1                                          = 0000
 ADDRESS #2                                          = 0000
 ADDRESS MASK (ONES ENABLE)                          = FFFF
 ADDR TYPE [INCL, EXCL, SNGL]                        = INCL
 DATA #1                                             = 0000
 DATA #2                                             = 0000
 DATA MASK (ONES ENABLE)                             = 0000
 DATA TYPE [INCL, EXCL, SNGL]                        = INCL
 EXTERNAL PROBES DATA VALUE                          = 00
 EXTERNAL PROBES DATA MASK (ONES ENABLE)             = 00
```

Prompts for the Jump Condition

```
STATE 0: JUMP CONDITION DESTINATION
 JUMP TO STATE (0-3) = 0

STATE 0: JUMP CONDITION
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF
 ADDRESS #1                                          = 0000
 ADDRESS #2                                          = 0000
 ADDRESS MASK (ONES ENABLE)                          = FFFF
 ADDR TYPE [INCL, EXCL, SNGL]                        = INCL
 DATA #1                                             = 0000
 DATA #2                                             = 0000
 DATA MASK (ONES ENABLE)                             = 0000
 DATA TYPE[INCL, EXCL, SNGL]                         = INCL
 EXTERNAL PROBES DATA VALUE                          = 00
 EXTERNAL PROBES DATA MASK (ONES ENABLE)             = 00
```

Prompts for the Range Timer

```
STATE 0: RANGE TIMER 1 START
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF
 ADDRESS #1                                          = 0000
 ADDRESS #2                                          = 0000
 ADDRESS MASK (ONES ENABLE)                          = FFFF
 ADDR TYPE[INCL, EXCL, SNGL]                         = INCL
 DATA #1                                             = 0000
 DATA #2                                             = 0000
 DATA MASK (ONES ENABLE)                             = 0000
 DATA TYPE[INCL, EXCL, SNGL]                         = INCL
 EXTERNAL PROBES DATA VALUE                          = 00
 EXTERNAL PROBES DATA MASK (ONES ENABLE)             = 00
```

You then get an exact duplicate of the Range-Timer prompts, *except* that the first line reads:

```
STATE 0: RANGE TIMER 1 STOP
```

The start and stop qualifications do *not* have to be the same.

Prompts for the Point Timer

```
STATE 0: POINT TIMER 1
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF
 START ADDRESS                                       = 0000
 STOP ADDRESS                                        = 0000
 ADDRESS MASK (ONES ENABLE)                          = FFFF
 ENABLE [BOTH, START, STOP]                          = BOTH
 DATA #1                                             = 0000
 DATA #2                                             = 0000
 DATA MASK (ONES ENABLE)                             = 0000
 DATA TYPE[INCL, EXCL, SNGL]                         = INCL
 EXTERNAL PROBES DATA VALUE                          = 00
 EXTERNAL PROBES DATA MASK (ONES ENABLE)             = 00
```

The point timer prompt, ENABLE [BOTH, START, STOP], lets you select whether a qualified event will:

1) Toggle the present point-timer run condition (start or stop)
2) Start the point timer
3) Stop the point timer

The 0 option lets you start and stop the timer any number of times within the same state. The START option in a state requires that a higher-numbered state include a point timer configured with BOTH or STOP.

Prompts for the Global Counters

```
GLOBAL COUNT VALUES
 SEQUENCE COUNT (1-FFFF)            = 0001
 DELAY COUNT (0-7FF)                = 000
 TRACE COUNT (1-7FF,0=INFINITE)     = 000
```

Prompts for the time-out function

```
SECONDS      IN DECIMAL (000-999) = 000
MILLISECONDS IN DECIMAL (000-999) = 000
MICROSECONDS IN DECIMAL (000-999) = 000
NANOSECONDS  IN DECIMAL (000-999) = 000
```

Associated Commands IBTT (Initialize Breakpoint/Trace/Timing Card), Page 4-63.

Example Refer to Section 3.7 on Page 3-17.

Syntax Prompt Mode : CASB<TERM>

Description The CASB command clears all existing software breakpoints. Note that using the Delete-Software-Breakpoint mode of the INIT command (Page 4-75) may make this command unnecessary.

Parameters None

Associated Commands DSB (Display all Software Breakpoints), Page 4-41. SSB (Set Single Software Breakpoint), Page 4-102.

Example This example clears breakpoints set at >10, >40, and >600 with the SSB command, then uses the DSB (Display Software Breakpoints) command to show that clearing occurred.

DISPLAY	ENTER
?	DSB<CR>
0010 0040 0600	
?	CASB<CR>
CASB	
?	DSB<CR>
DSB	
?	

Syntax Prompt Mode : CRUN<TERM>

Description The CRUN command continues program execution from where a halt occurred, and continues adding trace samples to the existing trace buffer, which it leaves intact.

This permits the following actions:

- After a software breakpoint (SDP) halt, the CRUN command moves your program to the next hardware or software breakpoint with tracing continued during that execution time.
- After a trace-memory-full (TMF) halt, the CRUN command suppresses further TMF halts. Tracing continues, with wrap around as required. Hardware breakpoint can occur.
- After a hardware-breakpoint halt, the CRUN command suppresses further HBP halts, but tracing continues and a TMF halt can occur.
- After a key halt, the CRUN command starts program execution at the next program-counter location. Tracing continues, and either a hardware breakpoint or TMF halt can occur.

Changing any BTT- or IBTT-command parameter clears the trace buffer and restarts it at 0.

Parameters None

Associated Commands BTT (Breakpoint/Trace/Timing), Page 4-16. IBTT (Initialize BTT), Page 4-63.

Example Under the assumption of no BTT or IBTT-command parameter changes, this example starts executing at the current program-counter value and continues to add samples to the trace buffer.

DISPLAY
```
?
CRUN
   RUNNING
```

ENTER
```
CRUN<CR>
```

Syntax

Prompt Mode : CSB<TERM>
Parenthesis Mode : CSB(<value>)<TERM>

Description

The CSB command clears the software breakpoint at a selected address. You get an error message if that address has no software breakpoint set.

In decimal mode, you can enter the address as a decimal number, but the monitor program converts it to hexadecimal.

Note that responding with 1=YES to the Delete-Software-Breakpoint parameter of the INIT command (Page 4-75) may make this command unnecessary.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Breakpoint Address	0 - FFFF	0

Breakpoint Address: Sets memory address of the software breakpoint to be cleared.

Associated Commands SSB (Set Software Breakpoint), Page 4-102.

Example

DISPLAY	ENTER
?	CSB<CR>
CSB BREAKPOINT ADDRESS = 0000	1A<CR>
?	SSB<CR>
SSB BREAKPOINT ADDRESS = 001A	<CR>
?	

This program clears a software breakpoint previously set at location >001A, then resets it. Notice that >1A entered for the CSB command becomes the default address for the SSB command.

Syntax Prompt Mode : DBTT<TERM>

Description **This command is available only with a Breakpoint/Trace/Timing card installed.** The DBTT command shows which parameters you have entered in the Breakpoint/Trace/Timing sequence. It also shows the sequence flow (start state, stop state, and jump-to state) of all BTT-card enable conditions; however, it does not show the address mask, data mask, and external qualifiers.

Each enabled state displays its data; each inactive state displays ****INACTIVE. The sequence, delay, and trace counts also display, along with a message stating that hardware breakpoint or trace-memory-full cannot occur, if the conditions assigned by the IBTT command would make them impossible (for example, selection of an option that does not include breakpoint). TIMEOUT and its corresponding time also appear if timeout is enabled.

Parameters None

Associated Commands BTT, Page 4-16. IBTT, Page 4-63.

Example

```
? DBTT<CR>
DBTT

XDS TIMING AND ANALYSIS VERSION 1.0
STATE 0: START
   BP TO STATE 1 IF  0010 [IAQ && SNGL AD(F012-F012)]
STATE 1:
 BP TO STATE 2 IF  0001[IAQ && INCL AD(F010-F020)]
 JUMP TO STATE 0   IF  [IAQ && SNGL AD(F013-F013)]
STATE 2: STOP
 BP (END SEQ)  IF  0001[MA && SNGL AD(F009-F009)]
STATE 3:
 ****INACTIVE
SEQUENCE COUNT=0001    DELAY COUNT=0000    TRACE COUNT=0000
NOTE: TMF CANNOT OCCUR
STOP ON TIME (SEC .. NS) = 010 000 000 000

?
```

STATE ***DESCRIPTION***

State 0 The BTT card sequences to state 1 after >0010 breakpoint events. The breakpoint event qualifier is instruction-acquisition at address >F012.

State 1 The BTT card sequences to state 2 after one breakpoint event. The breakpoint qualifier is an instruction acquisition on any address from >F010 through >F020. Or, the BTT card immediately jumps to state 0 if instruction acquisition at >F013 occurs, because the jump condition has priority over the breakpoint function.

State 2 This is the end-of-sequence state. The BTT card sequences to the next state or sends a hardware breakpoint to halt the emulator, if a memory access to address >F009 occurs.

Counts The global counts indicate that state 2 (end state) must only be entered and exited once (Outer loop count = 1), and that the trace count is set to infinite; therefore a trace-memory-full condition cannot occur.

STOP ...) The BTT card also halts the emulator after 10 seconds of run time if a hardware breakpoint does not occur first.

Syntax

Prompt Mode : DDM<TERM>
Parenthesis Mode : DDM(<value>,<value>)<TERM>

Description

The DDM command displays the content of a single selected data-memory location, two locations, or the range from one location to another location.

In decimal mode, you can enter the addresses as decimal numbers, but the monitor program converts them to and displays them as hexadecimal. You can display the content as decimal or magnitude numbers by preceding DDM with DEC or MAG, or leave them as hexadecimal numbers by simply entering DDM.

If a range display scrolls off the screen, press any key except ESC to stop the scrolling. Then press any key except ESC to start it again. Press ESC to end the display and the command.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Start Address	0 - FFFF	0000
End Address	0 - FFFF	0000

Start Address: First location in data memory to have content displayed. In decimal mode, you can enter this as a decimal number, but the monitor program changes it to and displays it as hexadecimal.

End Address: Last location in data memory to have content displayed. This can also be entered as a decimal number in decimal mode. If end address greater than start address, display includes content of both addresses and all addresses in between. If end address less than start address, only the content of the start location displays.

Display: Display is in lines of eight addresses each. The first of each line displays the address, an equal sign, and the content of that address. The display for the next seven bytes is just the content. The final part of the display is two characters per address: ASCII characters for any byte that is a displayable ASCII character, a dot (.) otherwise.

In hexadecimal mode, content displays as hexadecimal numbers. In decimal mode, content displays as hexadecimal mode with the selected hex notation, unless you precede DDM with DEC for decimal display or MAG for magnitude display.

Associated Commands DPM (Display Program Memory), Page 4-36.

Example 1 You want to display the content of data-memory locations >400 through >410 as hexadecimal numbers.

```
DISPLAY                                          ENTER
?                                                DDM<CR>
DDM
 START ADDRESS = 0000                            0400<CR>
 END ADDRESS   = 0000                            0410<CR>

0400=4449 5350 4C41 5920 4F46 2041 5343 4949  DI SP LA Y. OF .A SC II
0408=E8E9 F020 E832 F35E AEE9 5566 9280 D133  .. .  .2 .  .. Uf .. .3
0410=8754                                       .T

?
```

The first line contains the content of eight addresses, in hexadecimal and ASCII, starting at the address that you entered (displayed in the leftmost column). The ASCII display is the character, where displayable, or a dot, where it isn't.

The second line contains the content of the next eight addresses. The last line has a single entry for location >0010.

Example 2 You want to display the content of the same memory locations as decimal numbers after selecting decimal mode (TI display) with the INIT command.

```
DISPLAY                                          ENTER
?                                                DEC DDM <CR>
DEC
DDM
 START ADDRESS = 0400                            <CR>
 END ADDRESS   = 0410                            <CR>

0400=+17481 +21238 +19521 +22816 +20294 +8257  +21315 +18761
0408= -5911  -4064  -6094  -3234 -20759 +21862 -28032 -11981
?
```

The first line contains the decimal content of eight addresses. It has no ASCII display.

The second line contains the content of the next eight addresses.

Syntax Prompt Mode : DEC<TERM>

Description The DEC command causes the value of the *next command only* in a command string to display as a signed decimal number, **if and only if** you have selected decimal mode with the INIT command. If you have not, the DEC command causes an error message.

This format change affects only the selected command and remains in effect until changed by a HEX or MAG command preceding the same command or until you start a new session (default display format is hexadecimal).

Parameters None

Associated Commands HEX (Hexadecimal), Page 4-61. MAG (Magnitude), Page 4-88.

Example

```
DISPLAY                                                          ENTER

?                                                                INIT
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]     = INTERNAL  <CR>
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES] = NO        <CR>
      HEX NOTATION [0=NONE, 1=TI, 2=X, 3=HS, 4=HP]  = NONE      1<CR>
  BP: NUMBER OF EXTENDED ADDRESS BITS (0-8)         = 0         <CR>
 EXP: MAP OVERLAP MODE [0=HIGH/LOW,1=PROGRAM/DATA]  = HIGH/LOW  <CR>
      WAIT FOR TARGET READY [0=NO,1=YES]            = NO        <CR>
      ENABLE BUS-REQUEST MODE [0=NO,1=YES]          = NO        <CR>
?                                                                PC=20<CR>
?                                                                PC<CR>
 PC=20
?                                                                DEC PC<CR>
 ERROR 0603 DECIMAL MODE NOT INITIALIZED -- USE INIT
?                                                                INIT(0  0  1
              Q)<CR>
?                                                                PC<CR>
PC=+32
?                                                                HEX PC<CR>
PC=>20
?                                                                DEC PC
PC=+32
?
```

Further display of the PC in the DHS, DPS, DR, IR, or MR commands will be in decimal format unless:

1) You change it back to hexadecimal with the HEX command, Page 4-61,

2) You start a new session (defaults to hexadecimal),

3) You change it to magnitude with the MAG command, Page 4-88

4) You change a format within one of the above commands with CNTL/X.

Syntax Prompt Mode : DES<TERM>

Description The DES command displays the current emulator status including clock-cycle time, wait cycles, alternate-run mode status, and port-communication status.

Cycle time
Measured cycle time of device being emulated.

Automatic first wait state
Emulator automatically inserts wait state on all off-device memory accesses if you use the MAP command with cycle time less than 200 ns whether or not that location falls within the MAP-command block.

ARM ON, if in alternate-run mode; OFF, if not.

Ports Status of RTS, CTS, and DSR at XDS side of each XDS port. Port is Online if both CTS and DSR read 1, Offline otherwise.

Parameters None

Associated Commands ARM (Alternate-Run mode), Page 4-13. DISARM (Disable Alternate-Run mode), Page 4-31. INIT (Initialize), Page 4-75. MAP (Map Emulator Memory), Page 4-89.

Example Display the emulator status.

DISPLAY
```
?
DES
```

ENTER
```
DES<CR>
```

```
                    CYCLE TIME = 200 NSEC
  AUTOMATIC FIRST WAIT STATE = OFF
                          ARM = OFF


      PORT A      RTS:1      CTS:1      DSR:1    ONLINE
      PORT C      RTS:1      CTS:0      DSR:1    OFFLINE
      PORT D      RTS:1      CTS:1      DSR:1    ONLINE

?
```

Syntax Prompt Mode : DHS<TERM>

Description The DHS command displays a code or codes that indicate(s) the reason(s) for the last halt condition.

> **Note:**
>
> For an interrupt in Alternate-Run mode, the monitor program runs in the command-entry mode. When execution halts, the screen displays a Halt code from the table below, then displays RUNNING.
>
> The Breakpoint/Trace/Timing card ceases trace operations and lets you use the FT (Find Trace), IT (Inspect Trace), and DT (Display Trace) commands.

Parameters None

Associated Commands None.

Example

```
DISPLAY                                          ENTER
?                                                RUN<CR>
RUN
 RUNNING
                                                 <any key>
   KEY
   PC= 0063 ST0= E600 ST1= 03F0

?                                                DHS<CR>
KEY†
PC= 0063    ST0= E600    ST1= 03F0

?
```

†Halt codes are:

CODE	DEFINITION
ABORT	<ESC> entered after RUN command entered, but before execution started.
HBP	Hardware breakpoint interrupted program execution. (HBP display includes trace sample being processed when interrupt occurred.)
KEY	Keyboard entry interrupted program execution.
MULTI	Multiprocessing configuration resulted in interrupt.
PERR	Parity error halted program execution.
POR	System power up.
RES	RESET on XDS Operator Panel pressed.
RUNNING	Processor still executing instructions.
SBP	Software breakpoint interrupted program execution.
SS	Single-step command entered.
STOP	Program running in ARM received a STOP command.
TIME	Execution halted by BTT time-out.
TMF	Trace memory full.

Syntax

Prompt Mode : DIO<TERM>
Prompt Mode : DIO(<value>)<TERM>

Description

The DIO command lets you see the value at a specified port.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
0 - F	0	0

In decimal mode, you can enter the address as a decimal number, however, the monitor program converts it to hexadecimal.

Associated Commands None

Example 1

This example shows the current value at Port #1.

DISPLAY	ENTER
?	DIO<CR>
DIO	
PORT ADDRESS = 0	1<CR>
DATA = 00FF	
?	

Example 2

This example assumes the decimal mode and shows the current value at Port #1 as a decimal number.

DISPLAY	ENTER
?	DEC DIO<CR>
DEC	
DIO	
PORT ADDRESS = >0	1<CR>
DATA = +255	
?	

Syntax Prompt Mode : DISARM<TERM>

Description The DISARM command disables the Alternate-Run Mode, but processor execution continues. You must use the STOP command (Page 4-104) to halt emulator operation in the Alternate-Run Mode.

To enter the DISARM command, wait for a processor halt at a hardware or software breakpoint, or press any key.

Parameters None

Associated Commands ARM, Page 4-13.

Example This example assumes the emulator running in the Alternate-Run Mode.

DISPLAY	ENTER
?	DISARM<CR>
?	

Syntax

Prompt Mode : DL<TERM>
Parenthesis Mode : DL(<value>,<value>, ..., <value>)<TERM>

Description

The DL command configures the emulator to receive object code from an external device, usually a host computer system.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Load Offset	0 - FFFF	0000
Destination	0 = PROGRAM 1 = PROM 2 = DATA 3 = ASM	PROGRAM
Protocol	0 = NONE 1 = TEK 2 = ASR 3 = VAX	NONE
Source	0 = HOST 1 = USER	HOST

Load Offset: Address where storage of downloaded data starts. Values other than zero apply only to TI formats with relocatable object files. In decimal mode, you can enter the address as a decimal number, but the monitor program converts it to hexadecimal.

Destination: Sets downloaded data destination. If downloaded data exceeds available memory, monitor program downloads what fits and ignores the rest.

Values:

0 = PROGRAM	Data downloads into program memory
1 = PROM	Data downloads directly into PROM programmer/logging device on Port C
2 = DATA	Data downloads into data memory
3 = ASM	Data downloads into assembler.

Protocol: Defines data-transfer handshaking protocols including codes for beginning and ending downloads, beginning and ending uploads, and pass-through characters. Values are:

0 = NONE — No handshake protocol. You must define control characters by executing the IHC command before using this value.

1 = TEK — Enables Tektronix protocol. You must define control characters by executing the IHC command before using the DL command with protocol code of 1. Messages returned are:

0 = Record transferred without error.
7 = Error detected; retransmit record.

2 = ASR — Enables ASR terminal protocol. Controls are:

CNTL/R = Start download
CNTL/S = End download
CNTL/Q = Start upload
CNTL/@ = End upload
CNTL/P = Pass-through character

3 = VAX — Enables VAX/PDP-11 system protocol. Controls are:

CNTL/A = Start upload
CNTL/Z = End upload
CNTL/V = Start download
CNTL/W = End download
CNTL/P = Pass-through character

Source: Defines downloaded data source. Values are:

0 = HOST — Sets host computer as data source for download into emulator.

1 = USER — Sets user's terminal as data source for download into emulator for intelligent terminals or personal computers as terminals. Data stored in these terminals can then be downloaded through Port A.

Associated Commands IHC (Initialize Host Control Characters), Page 4-70.

Example This example downloads an object file from a host computer into the XDS unit starting at location >200 and using the VAX protocol.

```
DISPLAY                                                          ENTER
?                                                                 DL<CR>
DL
 LOAD OFFSET                                          = 0000      200<CR>
 DESTINATION [0=PROGRAM,1=PROM,2=DATA,3=ASM]          = PROGRAM   <CR>
 PROTOCOL [0=NONE, 1=TEK, 2=ASR, 3=VAX]               = NONE      3<CR>
 SOURCE [0=HOST, 1-USER]                              = HOST      <CR>

?
```

Syntax

Prompt Mode : DMAP<TERM>
Parenthesis Mode : DMAP(<value>)<TERM>

Description

The DMAP command lets you display the TMS320C2x emulation data- and program-memory map.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Display	0=MEM 1=TYPE 2=STATE	MEM

0=MEM: Displays memory map in 1K blocks with symbol "M" for either expansion or emulator memory and symbol "-" for no memory selected.

1=TYPE: Displays type of memory being mapped as follows:

T = Onboard Target RAM
A = Memory expansion configured as RAM
O = Memory expansion configured as ROM
- = No substitution

2=STATE: Displays wait-state information and memory type.

Associated Commands MAP, Page 4-89

Example 1

Assume that execution of the ITR and MAP commands has mapped the following memory blocks into the emulator:

- Four 1K blocks of RAM mapped into Data memory at address >6000.
- Eight 1K blocks of ROM mapped into Program memory at addresses >2000 and >C000.
- Four 1K blocks of onboard target RAM initialized into program memory at address >0000.

Now display the location of all memory with 0=MEM parameter.

```
DISPLAY                                                    ENTER
?                                                          DMAP <CR>
DMAP
 DISPLAY [0=MEM, 1=TYPE, 2=STATE] = MEM                    <CR>

ADDR          DATA          ADDR                  PROGRAM

0000 ---- ---- ---- ----    0000 MMMM MMMM MMMM MMMM
4000 ---- ---- MMMM ----    4000 ---- ---- ---- ----
8000 ---- ---- ---- ----    8000 ---- ---- ---- ----
C000 ---- ---- ---- ----    C000 MMMM MMMM ---- ----
               M: MEMORY          -: NO SUBSTITUTION

?
```

Example 2

Now execute DMAP again and set parameter to 1=TYPE to display the memory type at each location. For the memory blocks described in the first example, the following occurs:

```
DISPLAY                                                    ENTER
?                                                          DMAP <CR>
DMAP
 DISPLAY [0=MEM, 1=TYPE, 2=STATE] = MEM                    1<CR>

ADDR          DATA          ADDR                  PROGRAM

0000 ---- ---- ---- ----    0000 TTTT ---- OOOO OOOO
4000 ---- ---- AAAA ----    4000 ---- ---- ---- ----
8000 ---- ---- ---- ----    8000 ---- ---- ---- ----
C000 ---- ---- ---- ----    C000 OOOO OOOO ---- ----

T:ONBOARD TARGET RAM  A:RAM  O:ROM  -:NO SUBSTITUTION

?
```

Example 3

Now execute DMAP again and set parameter to 2=STATE to display the number of wait states and memory type at each location. For the memory blocks described in the first example, the following occurs:

```
DISPLAY                                                    ENTER
?                                                          DMAP <CR>
DMAP
 DISPLAY [0=MEM, 1=TYPE, 2=STATE] = TYPE                   2<CR>

ADDR          DATA          ADDR                  PROGRAM

0000 ---- ---- ---- ----    0000 TTTT ---- OOOO OOOO
                                                3333 3333
4000 ---- ---- AAAA ----    4000 ---- ---- ---- ----
                1111
8000 ---- ---- ---- ----    8000 ---- ---- ---- ----

C000 ---- ---- ---- ----    C000 OOOO OOOO ---- ----

                                     4444 4444

T:ONBOARD TARGET RAM A:RAM O:ROM -:NO SUBSTITUTION
 0-7:NUMBER OF WAIT STATES

?
```

Syntax

Prompt Mode : DPM<TERM>
Parenthesis Mode : DPM(<value>,<value>)<TERM>

Description

The DPM command displays the content of a selected area of TMS320C2x program memory.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Start Address	0 - FFFF	0000
End Address	0 - FFFF	0000

Start Address: First memory location to be displayed. In decimal mode, you can enter this as a decimal number, but the monitor program changes it to and displays it as hexadecimal.

End Address: Last memory location to be displayed. The command displays the content of all addresses from the Start Address to the End Address, except that, if the Start Address is greater than the End Address, it displays only the content of the Start Address. The end address can also be entered as a decimal number in decimal mode.

Display: Display is in lines of eight addresses each. The first of each line displays the address, an equal sign, and the content of that address. The display for the next seven bytes is just the content. The final part of the display is two characters per address: ASCII characters for any byte that is a displayable ASCII character, a dot (.) otherwise.

In hexadecimal mode, content displays as hexadecimal numbers. In decimal mode, content displays as hexadecimal mode with the selected hex notation, unless you precede DPM with DEC for decimal display or MAG for magnitude display.

Associated Commands

Example 1

You want to display the content of program-memory locations >0100 through >0110 as hexadecimal numbers.

```
DISPLAY                                          ENTER
?                                                DPM<CR>
DPM
 START ADDRESS = 0000                            0100<CR>
 END ADDRESS   = 0000                            0110<CR>

0100=4449 5350 4C41 5920 4F46 2041 5343 4949  DI SP LA Y. OF .A SC      II
0108=E8E9 F020 E832 F35E AEE9 5566 9280 D133  .. .  .2 .  .. Uf ..      .3
0110=8754                                       .T

?
```

The first line contains the content of eight addresses, in hexadecimal and ASCII, starting at the address that you entered (displayed in the leftmost column). The ASCII display is the character, where displayable, or a dot, where it isn't.

The second line contains the content of the next eight addresses. The last line has a single entry for location >0010.

Example 2

You want to display the content of the same memory locations as decimal numbers.

DISPLAY	ENTER
?	INIT(.,.,1,Q)<CR>
?	DEC DPM<CR>
DEC	
DPM	
START ADDRESS = 0100	<CR>
END ADDRESS = 0100	>10<CR>

```
0000=+17481 +21238 +19521 +22816 +20294 +8257  +21315 +18761
0008= -5911  -4064  -6094  -3234 -20759 +21862 -28032 -11981
?
```

Because the decimal displays can be longer, the command display in decimal mode does not include ASCII.

Syntax Prompt Mode : DPS<TERM>

Description The DPS command displays the current value of all registers in the digital signal processor currently mounted on the TMS320C2x Emulator card. The display includes a bit-by-bit display of both Status Registers.

The TMS320C2x registers are:

PC	Program Counter
ST0	Status Register 0 (detailed below)
ST1	Status Register 1 (detailed below)
T	Temporary Register
P	Product Register
ACC	Accumulator
AR0 - ARn	Auxiliary registers. TMS32020 has AR0 - AR4; TMS320C25 has AR0 - AR7.
Sn	Stack. For emulation, TMS32020 has three stack positions: S0 - S2; and TMS320C2x device has seven: S0 - S6. For either device, the emulator uses one stack position: S3, or S7.

Status-register 0 (ST0) contains the following:

ARP	Auxiliary-Register Pointer
OV	Overflow Flag
OVM	Overflow Mode
INTM	Interrupt Mode
DP	Data-Memory Page Pointer

Status-register 1 (ST1) contains the following:

ARB	Auxiliary-Register Pointer Buffer
CNF	RAM Configuration
TC	Test/Control Flag
SXM	Sign-Extension Mode bit
C	Carry bit (TMS320C2x only)
HM	Hold-mode bit (TMS320C2x only)
FSM	Frame-synchronization mode (TMS320C2x only)
XF	XF pin (general-purpose output)
FO	Format bit
TXM	Transmit-Mode bit
PM	Product-Shift Mode

Parameters None.

Associated Commands DR (Display Register), Page 4-40.

Example 1 TMS32020, Hexadecimal mode

DISPLAY

```
?
DPS
 PC =0000 T  =30D9      AR0=0001  AR4=40E6  S0=0000
 ST0=276C P  =002C44A8  AR1=C9EE            S1=0000
 ST1=2BF1 ACC=00000001  AR2=C4AA            S2=0000
                        AR3=366F

ARP OV OVM INTM DP        ARB CNF TC SXM XF FO TXM PM
 1  0   0   1   16C        1  0   1   0  1  0   0  1

?
```

ENTER

```
DPS<CR>
```

Note that these values are typical, but not necessarily what you will see on your screen.

Example 2 TMS32020, Decimal mode

```
DISPLAY                                                    ENTER
?                                                          DEC DPS<CR>
DEC
DPS
 PC = +0       T =  +12505     AR0= +1       AR4= +36   S0= +0
 ST0= +10092   P =  +2901160   AR1= +13842              S1= +0
 ST1= +11249   ACC= +1         AR2= +15190              S2= +0
                               AR3= +13935
ARP OV OVM INTM DP          ARB CNF TC SXM XF FO TXM PM
 +1 +0  +0  +1  +364         +1 +0  +1  +0 +1 +0  +0 +1

?
```

Example 3 TMS320C25, Hexadecimal Mode

```
DISPLAY                                                    ENTER
?                                                          DPS<CR>
DPS
 PC =0004  T  =FFFF       AR0=FFFF  AR4=1000  S0=FF00  S3=0000
 ST0=8FFF  P  =00000001   AR1=FFFF  AR5=FFFF  S1=0000  S4=0000
 ST1=EFF0  ACC=00000001   AR2=FFFF  AR6=FFFF  S2=0000  S5=0000
                          AR3=FFFF  AR7=EBFF  S6=0000

ARP OV OVM INTM DP          ARB CNF TC SXM C  HM  FSM  XF FO TXM PM
 7  0   0   1   1FF          7   0  1   1  1  1    1   1  0   0   0

?
```

Syntax Prompt Mode : DR<TERM>

Description The DR command displays the current content of the PC, ST0, ST1, T, P, and ACC registers. To see the individual items in Status Registers 0 and 1, use the DPS (Display Processor Status) command on Page 4-38.

In the decimal mode, you can display all register values in hexadecimal, decimal, or magnitude numbers if you precede the command name with HEX, DEC, or MAG. This display mode temporarily overrides any individual register or register-value assignments, but lasts only until you end the DR command.

The registers are:

PC	Program Counter
ST0	Status Register 0 (detailed below)
ST1	Status Register 1 (detailed below)
T	Temporary Register
P	Product Register
ACC	Accumulator

Parameters None

Associated Commands None.

Example 1 Hexadecimal display.

DISPLAY
```
?
DR
 PC =0000   T   =0000
 ST0=276C   P   =002C44A8
 ST1=03F0   ACC=00000001

?
```

ENTER
```
DR<CR>
```

Example 2 Decimal-mode display.

DISPLAY
```
?
DEC
DR
 PC = +0        T   = +12505
 ST0= +10092    P   = +2901160
 ST1= +11249    ACC= +1

?
```

ENTER
```
DEC DR<CR>
```

Syntax Prompt Mode : DSB<TERM>

Description The DSB command displays all current software breakpoints. Since this command only displays addresses, it is *not* affected by decimal mode.

Parameters None

Associated Commands SSB (Set Software Breakpoint), Page 4-102.

Example This example assumes previous setting of several software breakpoints by SSB (Set Single Breakpoint) commands.

DISPLAY	ENTER
?	DSB<CR>
DSB	
0002 003F 0810 0860	

The example shows display of breakpoints set at four locations.

Syntax

Prompt Mode : DT<TERM>
Parenthesis Mode : DT(<value>,<value>)<TERM>

Description

The DT command lets you display up to 2047 trace samples beginning at a selected sample.

Display format is as follows:

Index Trace-sample number, starting at 0001.

Cycle Flag. Entry of *EVT indicates last breakpoint event recorded. Entry of EVNT indicates conditions that have satisfied all breakpoint-event qualifications.

QUAL Type of memory access: IAQ, DMR, DMW, IA, IACK, PI, PMR, PMW. IA occurs for TMS320C25 only.

EXTQUALS
The binary value of the extended-address probes programmed by the INIT command as *data*, not address.

ADDR Number on address bus.

DATA Number on data bus.

RVRS ASSEMBLY
The assembly-language statement that produced the code.

For example,[13]

```
INDEX  S    CYCLE  QUAL   EXTQUALS  ADDR   DATA   RVRS ASSEMBLY
0133   SO   *EVT   IAQ    11111111  000C   FF80   B  >000C,*
```

indicates that a breakpoint event occurred on the 133rd program step which was an instruction-acquisition read at address >000C. The op code was >FF80, a branch instruction.

If you selected trace time-stamping with the IBTT command, then use the DT command, the header becomes:

```
INDEX S  CYCLE QUAL HRS MI SEC MS  US  NS   ADDR  DATA  RVRS ASSEMBLY
 0133 SO *EVT IAQ  ... .. ..  ... .26 800  000C  FF80  B  >000C,*
```

Entry of DT with no samples in the trace buffer causes a display of the header line along with any entries set by the BTT command, or the default values if the command hasn't been executed.

In the decimal mode, entering DEC DT or MAG DT causes any arguments in the reverse assembly to display as decimal or magnitude numbers; otherwise, it has no effect.

[13] Assuming the program entered in Section 3.7.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
First Sample	0 - 7FF (0=Oldest Sample)	000
Number Of Samples	0 - 7FE	000

First Sample: Sets number of first sample to be displayed from 0 for oldest sample to >7FE for most-recent sample to >7FF to display the last *nnn* samples where *nnn* is the number set in the next prompt. In the decimal mode, you can enter this as a decimal number, however, the monitor program converts it to hexadecimal.

Number of Samples: Sets number of samples to be displayed. The display scrolls if necessary to display all traces. If you select more samples than the trace buffer contains, the display ends at the last buffer entry. In the decimal mode, you can enter this as a decimal number, but the monitor program converts it to hexadecimal.

Pressing any key except ESC during scrolling freezes display. Pressing any key with the display frozen causes scrolling to continue.

Associated Commands IBTT (Initialize BTT), Page 4-63. IT (Inspect Trace), Page 4-83. FT (Find Trace), Page 4-53.

Example 1 This example uses the program entered in Section 3.7.

```
DISPLAY                                                    ENTER

?                                                          DT<CR>
DT
 FIRST SAMPLE (0-7FE, 0=OLDEST SAMPLE) = 000               7FE<CR>
 NUMBER OF SAMPLES                     = 000               C<CR>

INDEX  S  CYCLE QUAL EXTQUALS ADDR DATA RVRS ASSEMBLY

0077   0          IAQ  11111111 0008 55A9 MAR*+,AR1
0078   0          IAQ  11111111 0009 CE52 CMPR >2
0079   0          IAQ  11111111 000A F880 BBZ  >0007,*
0080   0          IAQ  11111111 0007 72AA SAR  AR2,*+,AR2
0081   0          IAQ  11111111 0008 55A9 MAR  *+,AR1
0082   0          IAQ  11111111 0009 CE52 CMPR >2
0083   0          IAQ  11111111 000A F880 BBZ  >0007,*
0084   0          IAQ  11111111 0007 72AA SARAR2,*+,AR2
0085   0          IAQ  11111111 0008 55A9 MAR*+,AR1
0086   0          IAQ  11111111 0009 CE52 CMPR >2
0087   0          IAQ  11111111 000A F880 BBZ  >0007,*
0088  S0 *EVT     IAQ  11111111 000C FF80 B>000C,*
?
```

Example 2 This example uses the IBTT command to time-stamp the traces shown in the previous example.

```
DISPLAY                                                    ENTER

?                                                          IBTT<CR>
IBTT
 INITIALIZE [0=OPTION, 1=BTT, 2=TRACE]    = OPTION         2<CR>
 TRACE DISPLAY [NORM, TIME, DELTA, MARK] = NORM            1<CR>
```

```
?                                                                DT<CR>
DT
 FIRST SAMPLE (0-7FE, 0=OLDEST SAMPLE) = 7FE                     7FE<CR>
 NUMBER OF SAMPLES                     = 000                     C<CR>

INDEX  S  CYCLE QUAL HRS MI SEC MS US NS  ADDR DATA RVRS ASSEMBLY

 0077  0        IAQ  ... .. ..  .. 48 200 0008 55A9 MAR*+,AR1
 0078  0        IAQ  ... .. ..  .. 48 600 0009 CE52 CMPR >2
 0079  0        IAQ  ... .. ..  .. 49 000 000A F880 BBZ  >0007,*
 0080  0        IAQ  ... .. ..  .. 50 200 0007 72AA SAR  AR2,*+,AR2
 0081  0        IAQ  ... .. ..  .. 50 600 0008 55A9 MAR  *+,AR1
 0082  0        IAQ  ... .. ..  .. 51 000 0009 CE52 CMPR >2
 0083  0        IAQ  ... .. ..  .. 51 400 000A F880 BBZ  >0007,*
 0084  0        IAQ  ... .. ..  .. 52 600 0007 72AA SAR  AR2,*+,AR2
 0085  0        IAQ  ... .. ..  .. 53 000 0008 55A9 MAR*+,AR1
 0086  0        IAQ  ... .. ..  .. 53 400 0009 CE52 CMPR >2
 0087  0        IAQ  ... .. ..  .. 53 800 000A F880 BBZ  >0007,*
 0088 S0 *EVT    IAQ  ... .. ..  .. 55 000 000C FF80 B>000C,*
 ?
```

Syntax Prompt Mode : DTIME<TERM>

Description **This command works only with a Breakpoint/Trace/Timing card installed.** The DTIME command displays the total time counted by each of the two BTT card timers. Any timer not enabled appears as zeros. The command also displays the average time for timer 1 along with the timer 1 data. The timer 2 average is not available.

Average time = (timer 1 total)/(# of timer 1 starts)

> **Note:**
>
> The XTIME command destroys the time values displayed by this command.

> **Note:**
>
> The XTIME command displays time values rounded up to 2/F where F is the emulation-clock frequency.

Parameters None

Associated Commands XTIME (), Page 4-113.

Example This example assumes prior selection of either a point timer or a range timer with the IBTT command and program execution.

```
@TIME<CR> DTIME

                HRS MI SEC MS  US  NS
   TIMER 1:     ... .. .2 503 817 750
   TIMER 1 AVG: ... .. .. ... .19 250
   TIMER 2:     ... .. .5 477 068 000
```

Syntax Prompt Mode : DTS<TERM>

Description The DTS command displays current trace count, number of breakpoint events remaining, number of delay counts remaining, current Breakpoint/Trace/Timing-card operating state, and number of sequence counts remaining.

Decimal mode does not affect this command.

Trace Cnt Displays current content of trace counter. This counter increments by 1 with each trace-sample saved, up to a limit of >7FF.

Events Left Shows current breakpoint/event counter reading. This counter decrements by 1 each time a breakpoint event occurs if the number set by the BP (Breakpoint) command was greater than 0. When the count reaches 0, the delay count starts.

Delays Left Shows current count in delay counter. After the event counter reaches 0, this counter decrements by 1 each time the monitor program takes a trace sample if the number set by the BP (Breakpoint) command was greater than 0. When this count reaches 0, program execution halts.

If the number of events left (above) is greater than 0, than the number displayed here is the number set by the BP command.

State Current BTT operating state.

Seq Cnt Current BTT sequence count. Count decrements by 1 after each breakpoint sequence.

Parameters None

Associated Commands BTT (Breakpoint/Trace/Timing), Page 4-16. DT (Display Trace), Page 4-42. IT (Inspect Trace), Page 4-83.

Example This example uses the program from Section 3.7.

DISPLAY	ENTER
?	DTS<CR>
DTS	
TRACE CNT EVENTS LEFT DELAYS LEFT STATE SEQ CNT	
00F 0002 0005 3 0003	
?	

The DTS command shows that the trace buffer has 15 samples and that the BTT card has two breakpoint events and three sequences remaining in State 3. Also, halt won't occur until after five traces.

Syntax

Prompt Mode : DV<TERM>
Parenthesis Mode : DV(<value>,<value>)<TERM>

Description

The DV command lets you do addition and subtraction of two decimal or two hexadecimal numbers or one of each. The command displays the algebraic sum and the absolute value of the difference.

With hexadecimal mode selected, all entry and display is in hexadecimal.

With decimal mode displayed, the monitor program assumes entry to be decimal, unless you precede a hexadecimal number with >, 0, 0X, or H, follow it with H, or precede it with 0 and follow it with H. Regardless of entry format, the command displays the entered values as hexadecimal numbers in the selected format.

Sum and difference values also display as hexadecimal numbers unless you precede the command with DEC or MAG.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Value 1	0 - FFFFFF	000000
Value 2	0 - FFFFFF	000000

Each of value 1 and value 2 is an unsigned number of up to six hexadecimal digits, six decimal digits with sign, or seven decimal digits without sign. The command:

- Converts decimal numbers to hexadecimal
- Aligns the least-significant digits of each number
- Adds the numbers
- Subtracts the smaller value from the larger value
- Displays both results as follows:

SUM >nnnnnn DIFFERENCE >nnnnnn

Both SUM and DIFFERENCE display with sign with decimal mode selected unless you precede the command with HEX or MAG.

Associated Commands None.

Example 1

DISPLAY	ENTER
?	DV <CR>
DV	
VALUE = 000000	4444 <CR>
VALUE = 000000	55555 <CR>
SUM >059999 DIFFERENCE >051111	
?	

Example 2

DISPLAY	ENTER
?	INIT(.,.,1,Q) DV <CR>
INIT	
DV	
VALUE = >000000	4444 <CR>
VALUE = >000000	55555 <CR>
SUM -5537 DIFFERENCE -14425	
?	HEX DV<CR>
HEX	
DV	
VALUE = >00115C	<CR>
VALUE = >00D903	<CR>
SUM >EA5F DIFFERENCE >C7A7	
?	MAG DV<CR>
MAG	
DV	
VALUE = >00115C	<CR>
VALUE = >00D903	<CR>
SUM 59999 DIFFERENCE 51111	

Syntax

Prompt Mode : FILL<TERM>
Parenthesis Mode : FILL(<value>,<value>,<value>,<value>)<TERM>

Description

The FILL command lets you enter a specific value into all memory locations in a selected range. Filling the entire emulator memory takes about 6 minutes.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Start Address	0 - FFFF	0
End Address	0 - FFFF	0
Data	0 - FFFF	0
Destination	0=PROGRAM 1=DATA	PROGRAM

Start Address: Sets beginning address to be filled with data. In decimal mode, you can enter this as a decimal number, however, the monitor program converts it to a hexadecimal number.

End Address: Sets ending address of range to be filled with data. In decimal mode, you can enter this as a decimal number, however, the monitor program converts it to a hexadecimal number.

Data: Sets hexadecimal value to be placed in memory. In decimal mode, you can enter this as a decimal number, but the monitor program always converts it to and enters it as hexadecimal.

Destination: Selects memory to be filled.

Associated Commands None

Example 1

Fill program memory from address >0000 to >000F with >AA:

```
DISPLAY                                                    ENTER
?                                                          FILL<CR>
FILL
 START ADDRESS                          = 0000             <CR>
 END   ADDRESS                          = 0000             F<CR>
 DATA                                   = 0000             AA<CR>
 DESTINATION [0=PROGRAM, 1=DATA] = PROGRAM                 <CR>
```

Verify fill by entering the DPM (Display Program Memory) command with the same range.

```
?                                                          DPM<CR>
DPM
 START ADDRESS = 0000                                      <CR>
 END ADDRESS   = 0000                                      F<CR>
0000=00AA 00AA 00AA 00AA 00AA 00AA 00AA 00AA  .. .. .. .. .. .. .. ..
0008=00AA 00AA 00AA 00AA 00AA 00AA 00AA 00AA .. .. .. .. .. .. .. ..

?
```

Example 2

Fill program memory from address +0 to +15 with +255.

```
DISPLAY                                                    ENTER
?                                                          DEC FILL<CR>
DEC
FILL
 START ADDRESS                          = >0000            <CR>
 END   ADDRESS                          = >0000            15<CR>
 DATA                                   = >0000            255<CR>
 DESTINATION [0=PROGRAM, 1=DATA] = PROGRAM                 <CR>
```

Verify fill by entering the DPM (Display Program Memory) command with the same range.

```
?                                                          DEC DPM<CR>
DEC
DPM
 START ADDRESS = >0000                                     <CR>
 END ADDRESS   = >0000                                     15<CR>
 0000=+255 +255 +255 +255 +255 +255 +255 +255
 0008=+255 +255 +255 +255 +255 +255 +255 +255

?
```

Syntax

Prompt Mode : FIND<TERM>
Parenthesis Mode : FIND(<value>, <value>, ..., <value>)<TERM>

Description

The FIND command searches a selected part of memory for a selected value and the address of all locations containing that value, with the screen scrolling if necessary. Each address display also includes the selected value.

Press any key to stop scrolling. Press any key again to resume scrolling. Press ESC to end the command and the display at any time.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Start Address	0 - FFFF	0
End Address	0 - FFFF	0
Data	0 - FFFF	0000
Data Mask (Ones Enable)	0 - FFFF	FFFF
Source	0=PROGRAM 1=DATA	PROGRAM

Start Address: Sets starting address for data search. In decimal mode, you can enter this address as a decimal number, but the monitor program converts it to a hexadecimal number.

End Address: Sets ending address for data search. In decimal mode, you can also enter this number as a decimal number.

Data: Sets search value. In decimal mode, you can enter this as a decimal number, but the monitor program converts it to hexadecimal.

Data Mask (Ones Enable): Refer to Page 5-12.

Source: 0=PROGRAM causes the command to search program memory, 1=DATA causes the command to search data memory.

Associated Commands None

Example 1

Displays all locations in address range >0000->0025 that contain >1A:

```
DISPLAY                                          ENTER
?                                                FIND<CR>

FIND
 START ADDRESS                 = 0000            <CR>
 END ADDRESS                   = 0000            25<CR>
 DATA                          = 0000            1A<CR>
 DATA MASK (ONES ENABLE)       = FFFF            <CR>
 SOURCE [0=PROGRAM,1=DATA]     = PROGRAM         <CR>
     0000=001A
     0001=001A
     0002=001A
     0003=001A
     0004=001A
```

Example 2 Displays all locations in address range +0 - +37 that contain +31:

```
DISPLAY                                                ENTER
?                                                      DEC FIND<CR>

DEC
FIND
 START ADDRESS                 = >0000                 <CR>
 END ADDRESS                   = >0000                 25<CR>
 DATA                          = >0000                 1A<CR>
 DATA MASK (ONES ENABLE)       = -1                    <CR>
 SOURCE [0=PROGRAM,1=DATA]     = PROGRAM               <CR>
    >0005= +31
    >0006= +31
?
```

Syntax

Prompt Mode : FT<TERM>
Parenthesis Mode : FT(<value>, <value>, ...,<value>)<TERM>

Description

The FT command searches the trace buffer for a particular trace sample.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Qualifier	0 = ANY 1 = IAQ 2 = PMR 3 = PMW 4 = DMR 5 = DMW 6 = I 7 = IACK 8 = PI 9 = IOR † = IOW	ANY
Trace Address #1 Number Of Ext. Address Bits = 0 Number Of Ext. Address Bits = 8	 0 - FFFF 0 - FFFFFF	 0
Trace Address #2 Number Of Ext. Address Bits = 0 Number Of Ext. Address Bits = 8	 0 - FFFF 0 - FFFFFF	 0
Range Indicator	0 = NO 1 = YES	NO
EMU Data-Compare Word	0 - FFFF	0000
EMU Data Mask (Ones Enable)	0 - FFFF	0000
Extended Data-Compare Byte Number Of Ext. Address Bits = 0 Number Of Ext. Address Bits = 8	 0 - FF 0	 0
Extended Data Mask Number Of Ext. Address Bits = 0 Number Of Ext. Address Bits = 8	 0 - FF 0	 0

†In hexadecimal mode, must enter this as IOW. Can enter as hexadecimal A (A with any of the hex symbols) in decimal mode.

Qualifier: Defines primary qualifying criteria for trace-memory search. Values are:

0 = ANY Searches all trace-sample types.
1 = IAQ Searches program-memory instruction-acquisition samples only.
2 = PMR Searches program-memory read samples only.
3 = PMW Searches program-memory write samples only.
4 = DMR Searches data-memory read samples only.
5 = DMW Searches data-memory write samples only.
6 = I Searches internal program type samples.
7 = IACK Searches interrupt-acknowledge trace samples only.
8 = PI Searches program-interrupt trace samples only.
9 = IOR Searches I/O read samples only.
>A = IOW Searches I/O write samples only. In the hexadecimal mode, you must enter this as IOW. In the decimal mode, you can enter A as a hexadecimal number (use one of the recognized hexadecimal symbols).

Trace Address #1: Selects trace-memory address to begin search, if Range Indicator OFF; or selects beginning of trace-memory-address range, if Range Indicator ON. In decimal mode, you can enter both trace-address #1 and trace-address #2 as decimal numbers, however, the monitor program converts them to hexadecimal numbers.

Trace Address #2: Selects second trace-memory address to be located, if Range Indicator OFF; or selects end of memory-address range, if Range Indicator ON.

Range Indicator: If 1=YES selected, Trace Address #1 and Trace Address #2 define a range of addresses to be searched; however, if Address #1 greater than Address #2, the command searches *all* memory addresses *except* the addresses between Trace Address #1 and Trace Address #2.

EMU Data-Compare Byte: The command compares this value to the trace sample. A match satisfies the search. In decimal mode, you can enter this value as a decimal number, however, the monitor program converts it to hexadecimal.

EMU Data Mask (Ones Enable): Masking lets you further refine the data-compare byte match. Refer to Section 5.1.12 on Page 5-12.

Extended Data-Compare Byte: The command compares this value to each trace-sample data value on the extended-address cable. A match qualifies a search sample. In decimal mode, you can enter this value as a decimal number, however, the monitor program converts it to hexadecimal.

Extended Data Mask (Ones Enable): Refer to Section 5.1.12 on Page 5-12.

INTERACTIVE OPTIONS FOR THE FT COMMAND		
FT COMMAND PARAMETER	**COMMAND AFFECTED**	**OPTION AFFECTED**
ANY,IAQ,PMR,PMW,DMR,DMW,I,IACK,IOR,IOW	IT	F
Trace Address #1	IT	F
Trace Address #2	IT	F
Range Indicator	IT	F
EMU Data-Compare Word	IT	F
EMU Data Mask	IT	F
Ext. Data Comp Byte	IT	F
Extended Data Mask	IT	F

INTERACTION WITH OTHER COMMANDS/PARAMETERS		
INTERACTING COMMAND	**INTERACTING PARAMETER**	**FT PARAMETER AFFECTED**
INIT	Number of external-address bits	Trace address #1 Trace Address #2 External data-compare byte External data mask

When the FT command "finds" a sample, the command calls the IT (Inspect Trace) command to display 19 trace samples with the "found" sample in the middle and flagged by an asterisk. Each time that you enter the F option in the IT command, search begins after the last sample found.

Associated Commands IT (Inspect Trace), Page 4-83.

Example

This example initializes onboard target RAM, enters a small assembly-language program, reverse-assembles that program, initializes the program counter, sets up tracing, then uses the FT command to find a particular trace.

> **Note:**
>
> Unplug the target-system cable to avoid bus conflicts.

```
DISPLAY                                                    ENTER
?                                                          ITR<CR>
ITR
 PROGRAM MEMORY RAM [0=OFF, 1=ON] = OFF                    1<CR>
 DATA MEMORY RAM    [0=OFF, 1=ON] = OFF                    <CR>

?                                                          XA <CR>
XA
 START ADDRESS                          = 0000             100<CR>
 USE OLD SYMBOL TABLE [0=NO, 1=YES] = NO                   <CR>
```

```
LINE   ADD    DATA   LABEL  MNEN     OPERANDS  COMMENT
0001   0100   C804          LDPK     4
0002   0101   D001          LALK     >100
       0102   0100
0003   0103   CE06          RSXM
0004   0104   6800   LOOP   SACH     0
0005   0105   5900          TBLW     0
0006   0106   D102          ADLK     >8000,1
       0107   8000
0007   0108   FF80          B        LOOP
       0109   0004
0008   010A                 END

ERRORS    WARNINGS  UNRESOLVED
0000      0000      0000
?
```

Verify correct program entry by means of the XRA command.

```
DISPLAY                                                    ENTER
?                                                          XRA
XRA
 START ADDRESS    = 0000                                   100<CR>
 LINES OF OUTPUT = 0000                                    6<CR>
     0000  C804  LDPK  004
     0001  D001  LALK  0100,0
     0003  CE06  RSXM
     0004  6800  SACH  00,0
     0005  5900  TBLW  00
     0006  D102  ADLK  8000,1
                 END
     ?
```

Now initialize the program counter and set the number of trace samples to 100

```
DISPLAY                                                          ENTER
?                                                                PC=100
        <CR>
?                                                                BTT
BTT
 SELECT [S0, S1, S2, S3, ALL, COUNT, TIME]            = ALL      <CR>

STATE 0: BREAKPOINT COUNTER
 EVENT COUNT (0-FFFF)                                 = 0001     <CR>

STATE 0: BREAKPOINT
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF      Q<CR>

STATE 0: TRACE
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = ALL      <CR>
 ADDRESS #1                                           = 0000     <CR>
 ADDRESS #2                                           = FFFF     <CR>
 ADDRESS MASK (ONES ENABLE)                           = FFFF     Q<CR>

GLOBAL COUNT VALUES
 SEQUENCE COUNT (1-FFFF)                              = 0001     <CR>
 DELAY COUNT (0=7FF)                                  = 000      <CR>
 TRACE COUNT (1-7FF, 0=INFINITE)                      = 000      100<CR>
```

Now execute the RUN command.

```
DISPLAY                                                          ENTER
?                                                                RUN<CR>
RUN
 RUNNING
 TMF
 PC= 0104   ST0=xxxx   ST1=xxxx
```

The FT command now searches the trace buffer for the first program-memory write cycle that enters >11.

```
DISPLAY                                                          ENTER

?                                                                FT<CR>
FT
 QUALIFIER [ANY,PR,PW,IAQ,DR,DW,IOR,IOW,IA,IACK] = ANY           <CR>
 TRACE ADDRESS #1                                = 0000          <CR>
 TRACE ADDRESS #2                                = 0000          200<CR>
 RANGE INDICATOR [0=NO, 1=YES]                   = YES           <CR>
 EMU DATA-COMPARE WORD                           = 0000          11<CR>
 EMU DATA MASK (ONES ENABLE)                     = 0000          FFFF<CR>
 EXT DATA-COMPARE BYTE                           = 00            <CR>
 EXT DATA MASK (ONES ENABLE)                     = 00            <CR>

            INDEX S CYCLE QUAL EXTQUALS  ADDR  DATA  RVRS ASSEMBLY

              0153 0       PW   11111111  0100  0010
              0154 0       PR   11111111  0107  8000
              0155 0       IAQ  11111111  0108  FF80  B    >0104
              0156 0       PR   11111111  0109  0004
              0157 0       IA   11111111  ----  ----
              0158 0       IAQ  11111111  0104  6800  SACH >00,0
              0159 0       IAQ  11111111  0105  5900  TBLW >00
              0160 0       IAQ  11111111  0106  D102  ADLK >8000,1
              0161 0       IA   11111111  ----  ----
            * 0162 0       PW   11111111  0100  0011
              0163 0       PR   11111111  0107  8000
              0164 0       IAQ  11111111  0108  FF80  B    >0104,*
              0165 0       PR   11111111  0109  0004
              0166 0       IA   11111111  ----  ----
              0167 0       IAQ  11111111  0104  6800  SACH >00,0
              0168 0       IAQ  11111111  0105  5900  TBLW >00
              0169 0       IAQ  11111111  0106  D102
              0170 0       IA   11111111  ----  ----  ADLK >8000,1
              0171 0       PW   11111111  0100  0012
            []
```

†The FT command found sample 162, displayed a "window" of trace data centered around the sample, then called the IT (Inspect Trace) command (Page 4-83).

Syntax Prompt Mode : GHALT<TERM>

Description The GHALT command stops execution of all emulators in a multiprocessing group.

Using this command when the XDS/22 is not in the multiprocessing mode causes an error message.

Parameters None

Associated Commands IMP (Initialize Multiprocessing Mode), Page 4-74.

Example

DISPLAY	ENTER
?	GHALT<CR>

?

The GHALT command stops all running emulators in the group. The emulator currently in the foreground retains control.

Syntax Prompt Mode : GRUN<TERM>

Description **This command used in Multiprocessing Mode only.** The GRUN command starts the Run condition in all emulators configured for synchronized start by execution of the IMD command. It also starts any unit configured as a master but not configured as start-synchronous.

Parameters None

Associated Commands IMD (Initialize Multiprocessor Mode), Page 4-72.

Example

DISPLAY	ENTER
?	GRUN<CR>

The display depends on the current status of the emulators in the chain. All emulators configured as start-synchronous by the IMD command start running.

Syntax Prompt Mode : HELP<TERM>

Description The HELP command displays the menu of the mnemonic names of all emulator commands. This display, plus a line displaying the device type, XDS, and firmware-revision level, also appears at power-up.

Parameters None

Associated Commands None.

Example

```
COMMANDS:
   INIT      IPM     IR      RUN       BTT     IT      HOST    IMP
   IPORT     IDM     DR      CRUN      IBTT    FT      IHC     IMD
   IPRM      DPM     MR      SS        DBTT    DT      UL      ID
   ICC       DDM     DIO     RTR                       DL      BGND
   RCC       MPM     MIO
   RESTART   MDM

   LOAD      FILL    DPS     ARM       SSB     DTIME   LOG     GRUN
   SAVE      FIND    DES     DISARM    DSB     XTIME   SNAP    TRUN
   HEX       ITR     DHS     STOP      CSB             HELP    GHALT
   DEC       MAP     DTS               CASB    XA      DV      THALT
   MAG               DMAP              XRA

VARIABLES:
   PC        AR      ACC     ARP       INTM    ARB     XF      C
   ST        S       T       OV        DP      CNF     FO      HM
             GREG    P       OVM               TC      TXM     FSM
                                               SXM     PM

   ?
```

Syntax Prompt Mode : HEX<TERM>

Description The HEX command causes the value of the *next command only* in a command string to display as a hexadecimal number in the format selected with the INIT command. This format change affects only the selected command and remains in effect until you change it with a DEC or MAG command.

If you enter this command with the format already set to hexadecimal, the value displays as a number with no hexadecimal sign.

Parameters None

Associated Commands DEC (Decimal), Page 4-27. INIT (Initialize), Page 4-75. MAG (Magnitude), Page 4-88.

Example

```
?                                                                INIT
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]     = INTERNAL  <CR>
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES] = NO        <CR>
      HEX NOTATION [0=NONE, 1=TI, 2=X, 3=HS, 4=HP]  = NONE      1<CR>
  BP: NUMBER OF EXTENDED ADDRESS BITS (0-8)         = 0         <CR>
 EXP: MAP OVERLAP MODE [0=HIGH/LOW,1=PROGRAM/DATA]  = HIGH/LOW  <CR>
      WAIT FOR TARGET READY [0=NO,1=YES]            = NO        <CR>
      ENABLE BUS-REQUEST MODE [0=NO,1=YES]          = NO        <CR>

?                                                                PC<CR>
 PC=0dddH

?                                                                HEX PC<CR>
 PC=dddH
```

After declaring the decimal mode with the INIT command, the PC display is a signed decimal number. After entering the HEX command, the PC display is a hexadecimal number followed by H (since that was the display format selected with the INIT command). Further display of the PC in the DHS, DPS, DR, IR, or MR commands will be in hexadecimal unless:

1) You change it back to decimal with the DEC (Decimal) command, Page 4-27,

2) You change it to magnitude with the MAG command, Page 4-88, or

3) You change the format with one of the above commands with CNTL/X.

Syntax Prompt Mode : HOST<TERM>

Description The HOST command places the emulator in Terminal Mode for communication with a computer system connected to Port D. The command displays a set of control characters to start and stop data transfer between the XDS unit and the host computer.

The command also displays a "pass-through" character, that is, a character that causes the XDS to transfer the *next* character without acting upon it.

The actual character display depends upon the protocol defined by the last DL (Download) or UL (Upload) command executed, or is the VAX protocol if neither command has been executed.

Parameters None

Associated Commands IPORT (Initialize Port), Page 4-79. IHC (Initialize Host Controls, Page 4-70.

Example

DISPLAY
?

ENTER
HOST<CR>

```
HOST
      EXIT          DOWNLOAD          UPLOAD          PASS
      <^E>          <^V><^W>          <^A> <^Z>       <^P>
```

where ^E, for example, means press CNTL and E together. Calling the HOST command with no previous use of the DL or UL command displays the VAX (default) values.

Syntax

Prompt Mode : IBTT<TERM>
Parenthesis Mode : Not applicable

Description

This command is available only with a Breakpoint/Trace/Timing card installed. The IBTT command lets you configure the BTT card by selecting combinations of its breakpoint, jump, timer, and trace resources. You can configure each of States 0-3 independently. You can also select an end-of-sequence state, external qualifications, TRIX (Trace on Instruction Acquisition) mode, and one of four trace-display configurations.

> **Note:**
>
> You cannot use either the parameter-display-only (!) or parenthesis modes of command entry described in Section 5.1 with this command.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Initialization	0=OPTION 1=BTT 2=TRACE	OPTION
State Option Selection	0 (0=OFF--1F) 1 (0=OFF--1F) 2 (0=OFF--1F) 3 (0=OFF--1F)	02 00 00 00
End-of-Sequence State (0-3)		0
TRIX Mode TRIX Mode	0=NO 1=YES	NO
External Qual- ifiers	0=NO 1=EXT	NO

Initialization: 0=OPTION: If you accept or select the default, 0, you get this display:

ADDRESS & DATA MODE

OPTION:	1	2	3	4	5	6	7	8
BP	2	1	1	1				
TR/TRIX		1			2	1		
JUMP			1					
R TIMER							1	
P TIMER				1		1		2

ADDRESS-ONLY MODE

OPTION:	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
BP	4	3	3	3	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1			
TR/TRIX		1			2				1	1		3	2	2	1	1					4	2	
JUMP			1			2			1		1		1		2		2	1	1				
R TIMER							1									1		1		1		1	2
P TIMER				1				2		1	1			1			1		2	1			

BP is hardware breakpoint. TR/TRIX is tracing. JUMP means that you can move from one state to a non-adjacent state.

`R Timer` means Range Timer which lets you start or stop a timer any time that program execution reaches a single memory address, two independent addresses, an inclusive range of addresses, or an exclusive range (all of memory *except* a certain range).

`P Timer` means Point Timer which lets you toggle (start or stop) a timer any time that program execution reaches a single memory address or either of two independent addresses. You can toggle as many times as toggle conditions occur within a single state or start in one state and stop in another.

Note that options 1 - 8 include both address and data-qualification, but that the remaining options include only address qualification.

After deciding what resources you need, enter the option number that corresponds to that selection at the State-Option prompts described in later text.

1=BTT: If you select 1=BTT, you get the same options, but no display. As with *0=OPTION*, you enter the option numbers at the prompts described in later text.

2=TRACE: If you select 2=TRACE, you get the following display:

```
TRACE DISPLAY [0=NORM, 1=TIME, 2=DELTA, 3=MARK] = NORM
```

0=NORM: In this default mode, the DT (Display Trace) or IT (Inspect Trace) commands give you the standard trace display (index, cycle, external qualifiers, address , data, reverse-assembly).

1=TIME: In this mode, the trace line consists of index, cycle type, time stamp, address, data, and reverse-assembly. The time stamp is the absolute time from start of Run mode to the time that the trace sample occurred.

If the trace buffer has not wrapped around, the first sample has the time of the first instruction; all other samples have a higher value.

2=DELTA: In this mode, the trace display is like the 1=TIME display above, except that the time given for each sample except the first sample is the elapsed time from the previous sample. For the first sample, the time given is the time from zero (start of instruction).

3=MARK: In this mode, you Mark a particular trace sample, and its collection time becomes time=0. Execution of the DT or IT command then displays all trace samples preceding the Marked sample display with a negative time offset from time=0; all following the Marked sample display with a positive time offset from time=0.

The initial display is with the first sample Marked, which means that all remaining samples have a positive offset. To Mark any other sample, enter M*nn* at the BTT cursor [] where *nn* is the index number of the trace sample of interest.

The trace-display command re-executes with that sample marked as time=0; all preceding samples marked with a negative time offset from time=0; and all following samples marked with a positive time offset from time=0.

If the index you enter doesn't exist, nothing happens.

You can execute the IBTT command repeatedly to change the trace display to look at the same trace data in a variety of ways to extract different types of information.

State Option Selection: Lets you select an option (combination of breakpoints, jumps, tracing, and timers) for State x. You can configure a state whether or not you use it during execution.

You can also select an option but not use all its features. For example, select option 1, but use only the first breakpoint. Refer to the BTT-command discussion on Page 4-16.

End-of-Sequence State (0-3): Lets you select the last or end state for BTT operation. You cannot use any states past the end-of-sequence state, but you can still configure them.

> **Note:**
>
> State 3 is always an end-of-sequence state.

> **Note:**
>
> It is not good practice to define an end-of-sequence state past the last state that you activate with the BTT command.

TRIX Mode [0=NO,1=YES]: Lets you enable or disable TRIX (Trace on Instruction Acquisition) prompts. If enabled, you receive TRIX prompts within each BTT command event qualifier. If disabled, no TRIX prompts appear.

External Quals [0=NO,1=EXT]: Lets you enable or disable External-Qualifier prompts. If enabled, you receive these prompts within each BTT command event qualifier. If disabled, no prompts appear.

Associated Commands BTT (Breakpoint/Trace/Timing), Page 4-16. DBTT (Display BTT), Page 4-24.

Example 1

This example configures State 1 to contain three breakpoints and one tracing function on addresses only and also sets State 1 as the end-of-sequence state. To save space, the example uses initialization option 1, no display.

```
DISPLAY                                                    ENTER
?                                                          IBTT<CR>
IBTT
 INITIALIZE [OPTION, 1=BTT, 2=TRACE]  = OPTION    1 <CR>
 STATE 0 OPTION (0=OFF--1F)           = 02        <CR>
 STATE 1 OPTION (0=OFF--1F)           = 00        0A<CR>
 STATE 2 OPTION (0=OFF--1F)           = 00        <CR>
 STATE 3 OPTION (0=OFF--1F)           = 00        <CR>
 END OF SEQUENCE STATE (0-3)          = 0         1<CR
 ALLOW TRIX MODE [0=NO,1=YES]         = NO        <CR>
 EXTERNAL QUALS [0=NO,1=EXT]          = NO        <CR>

?
```

Selection of >0A configures State 1 to contain three breakpoints and one tracing function on address only. Selection of State 1 as the end-of-sequence state lets the BTT card sequence through States 0 and 1. States 2 and 3 and the TRIX and EXTERNAL QUALS remain off.

Example 2 This example selects display of trace samples in the Delta mode.

```
?                                                              IBTT
IBTT
 INITIALIZE [OPTION, 1=BTT, 2=TRACE]                 = OPTION<CR>
 TRACE DISPLAY [0=NORM, 1=TIME, 2=DELTA, 3=MARK] = NORM  2<CR>

     ?
```

COMMENT: The selected trace-display mode remains in force until you enter a new mode.

Syntax

Prompt Mode : ICC<TERM>

Description

The ICC command adjusts your terminal cursor-control keys to the TMS320C2x Emulator's requirements (standard ASCII control codes).

> **Note:**
>
> Before using the SNAP command (Page 4-100), you must execute this command even if you simply enter the power-up values shown below.

If your terminal generates multiple-character sequences for cursor movement, this command will *not* work.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Cursor Up	<Up Arrow>	+
Cursor Down	<Down Arrow>	-
Cursor Left	<Left Arrow>	<
Cursor Right	<Right Arrow>	>

> **Note:**
>
> If your terminal has arrow keys, press the appropriate arrow key for each parameter. If your terminal uses control sequences, that is, if you must press CNTL (CTRL) along with some other key to move the cursor, then press the appropriate keys together for each parameter.
>
> If your terminal doesn't respond properly to this command, use the RCC (Restore Cursor Control) command and power-up values for cursor movement.

Associated Commands RCC (Restore Cursor Control), Page 4-95.

Example

DISPLAY
```
?
```

ENTER
```
ICC<CR>
```

```
PRESS KEY FOR CURSOR UP    =     Press Up Arrow key
PRESS KEY FOR CURSOR DOWN  =     Press Down Arrow key
PRESS KEY FOR CURSOR LEFT  =     Press Left Arrow key
PRESS KEY FOR CURSOR RIGHT =     Press Right Arrow key

?
```

The screen displays OK after each key depression to show that your entry now controls the cursor.

Executing the RCC command restores the original power-up cursor-control values.

After command execution, the program displays your entry of any power-up cursor-control value, but does NOT move the cursor.

If you define <CNTL>/I or <CNTL>/D as cursor sequences, they cannot be used for editing insert or delete.

Syntax Prompt Mode : ID<TERM>

Description The ID command displays the XDS device type and firmware release number. In the multiprocessing mode, it also displays the foreground emulator number.

Parameters None

Associated Commands None

Example

DISPLAY	ENTER	COMMENT
?	ID<CR>	Only one emulator running
TMS320C2x XDS VERSION x.x.x		
?		
	OR:	
#3		Unit #3 in MP chain is in foreground.
TMS320C2x XDS VERSION x.x.x		
?		

In either example, xx completes the device type being emulated, for example C25, for TMS320C25 emulation.

Syntax

Prompt Mode : IDM<TERM>
Prompt Mode : IDM(<value,>)<TERM>

Description

The IDM command displays the content of a selected data-memory location for review or change. You can then review or change adjacent locations without repeating the command.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Address	0 -FFFF	0000

Address: Selects address to be displayed. Keep the displayed displayed value and look at the next address by pressing +, Space bar, or RETURN. Keep the value and look at the previous address by pressing -. Keep the value and end the command by pressing ESC or typing Q, then pressing RETURN.

Change the displayed content by typing a new value, wiping out any remnant of the old value with the Space bar or the terminal DELETE key, then pressing RETURN. After entry, you can select adjacent locations as described above.

In decimal mode, you can enter a decimal value for the address, the new entry, or both; however, the monitor program converts any such values to hexadecimal.

Associated Commands IPM (Inspect Program Memory), Page 4-78. MAP (Map Expansion Memory), Page 4-89.

Example 1

This example displays the content of memory locations >0F0D through >0F0F, then changes the content of >0F0F.

DISPLAY	ENTER
?	IDM<CR>
IDM ADDRESS = 0000	0F0D <CR>
0F0D = 34F1 0F0E = 4251 0F0F = 425D 0F10 = 4250	<CR> <CR> 425E<CR> <ESC>
?	

Example 2

This is the same example, but with the decimal mode selected. You enter the addresses as +3853 through +3855, then enter the data change at +3855 as +16990.

DISPLAY	ENTER
? DEC IDM	DEC IDM<CR>
ADDRESS = >0000	3853 <CR>
+3853 = +13393 +3854 = +16977 +3855 = +16989 +3856 = +16976	<CR> <CR> 16990<CR> Q<CR>
?	

Syntax

Prompt Mode : IHC<TERM>

Description

The IHC command defines special control-key sequences to inform the emulator monitor program of download start or end, upload start or end, or that the *next* character should pass to the host with no XDS action.

For NONE or TEK protocol selection, execute this command before using the DL and UL commands. It is not required for the VAX or ASC protocols since these modes have predefined control characters.

> **Note:**
>
> This command does not honor the Preview mode (Section 5.1.4 on Page 5-5).

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Download Start	CNTL/(key)	CNTL/V
Download End	CNTL/(key)	CNTL/W
Upload Start	CNTL/(key)	CNTL/A
Upload End	CNTL/(key)	CNTL/Z
Pass-Through Character	CNTL/(key)	CNTL/P

> **Note:**
>
> The XDS unit sends the *next* character after the pass-through character on to the host with no XDS action.

Associated Commands DL (Download), Page 4-32. HOST (Host), Page 4-62. IPORT (Initialize Port), Page 4-79. UL (Upload), Page 4-107.

Example 1

Host computer uses <CNTL/U> as Upload-End character. User selects NONE as UL-command protocol.

```
DISPLAY                                              ENTER
?                                                    IHC<CR>
IHC
 DEPRESS KEY FOR DOWNLOAD START     = <CR>
 DEPRESS KEY FOR DOWNLOAD COMPLETE  = <CR>
 DEPRESS KEY FOR UPLOAD START       = <CR>
 DEPRESS KEY FOR UPLOAD COMPLETE    = <CNTL/U><CR>
 DEPRESS KEY FOR PASS THROUGH CHAR  = <CR>

?
```

The screen displays OK after each character input.

Example 2

You want to send <CNTL/V>, the normal download-start character, to the host computer with no XDS action.

```
DISPLAY                                                    ENTER
?                                                           IHC<CR>
IHC
 DEPRESS KEY FOR DOWNLOAD START      = <CR>
 DEPRESS KEY FOR DOWNLOAD END        = <CR>
 DEPRESS KEY FOR UPLOAD START        = <CR>
 DEPRESS KEY FOR UPLOAD END          = <CNTL/U><CR>
 DEPRESS KEY FOR PASS THROUGH CHAR   = <CNTL/P<CR>

?                                                          ^P ^V
```

The host computer receives <CNTL/V>.

Syntax

Prompt Mode : IMD<TERM>
Parenthesis Mode : IMD(<value>,<value>)<TERM>

Description

This command used in Multiprocessing Mode only. The IMD command sets the operational mode of the foreground emulator in a multiprocessing chain as one of independent, master, or slave. You must use the IMP (Initialize Multiprocessor mode) command before using this command. Each emulator must be initialized before multiprocessing.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Processor Mode	0 = IND 1 = MASTER 2 = SLAVE	IND
Start Synchronous	0 = NO 1 = YES	NO

Processor Mode: Defines operational mode for foreground emulator in multiprocessing chain as one of independent, master, or slave.

An Independent unit can start independently or synchronously and does not halt when other emulators halt.

A Master unit can also start independently or synchronously and does not halt when other emulators halt; but also responds to the GRUN (Group Run) command whether or not it starts synchronously.

A Slave unit can only start synchronously and stops when any other emulator configured to start synchronously halts.

Start Synchronous: If processor mode is independent or master, entry of 0 for this parameter says this emulator starts immediately after a RUN command. Entry of 1 places emulator in a group. Emulators in a group start only after all other emulators in the group have received a RUN command or after a GRUN (Group Run) command. Emulators in the slave mode always start synchronously.

Associated Commands IMP (Initialize Multiprocessing Mode), Page 4-74. *#n* (with ? displayed) to place emulator *n* in foreground.

Example

The multiprocessing chain contains four emulators with #1, the unit connected to the terminal, in the foreground. Previous execution of the IMP command has initialized the MP Mode. The example initializes the foreground unit, then changes unit #2 to the foreground and initializes it, then unit #3, then finally unit #4.

```
DISPLAY                                                          ENTER
?                                                                IMD<CR>
IMD
 PROCESSOR MODE [0=IND, 1=MASTER, 2=SLAVE]    = IND             MASTER<CR>
 START SYNCHRONOUS [0=NO, 1=YES]              = NO              YES<CR>

?                                                                #2<CR>

?                                                                IMD<CR>
IMD
 PROCESSOR MODE [0=IND, 1=MASTER, 2=SLAVE]    = IND             SLAVE<CR>
 START SYNCHRONOUS [0=NO, 1=YES]              = NO              <CR>

?                                                                #3<CR>

?                                                                IMD<CR>
IMD
 PROCESSOR MODE [0=IND, 1=MASTER, 2=SLAVE]    = IND             2<CR>
 START SYNCHRONOUS [0=NO, 1=YES]              = NO              <CR>

?                                                                #4<CR>

?                                                                IMD<CR>
IMD
 PROCESSOR MODE [0=IND, 1=MASTER, 2=SLAVE]    = IND             <CR>
 START SYNCHRONOUS [0=NO, 1=YES]              = NO              <CR>
```

The emulators are now configured for multiprocessing operations.

Syntax

Prompt Mode : IMP<TERM>
Parenthesis Mode : IMP(<value>)<TERM>

Description

The IMP command controls the Autopolling feature in the multiprocessing mode. This is the first command in the multiprocessing mode.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Autopoll	0 = NO 1 = YES	NO

Autopoll: Starts autopolling sequence. This feature lets an emulator with output information automatically switch from Background Mode to Foreground Mode and display the information on the terminal. All multiprocessing units must be in Background Mode for this feature to function.

Associated Commands BGND, Page 4-15.

Example

DISPLAY	ENTER
?	IMP<CR>
IMP AUTOPOLL (0=NO, 1=YES) = NO	1<CR>
TMS320C25 XDS VERSION 1.0.0 LAST EMULATOR = n	

This program initializes multiprocessing mode and starts autopolling. At this point, the user can control any emulator in the chain by entering #, then its ID number.

Syntax

Prompt Mode : INIT<TERM>
Parenthesis Mode : INIT(<value>,<value>, ..., <value>)<TERM>

Description

The INIT command lets you set several TMS320C2x emulation features, such as clock source and numeric mode.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
EMU: Clock Source	0=INTERNAL 1=TARGET 2=OSC	INTERNAL
Delete Software Breakpoint Mode	0=NO 1=YES	NO
Hex Notation	0=NONE 1=TI 2=X 3=HS 4=HP	NONE
BP: Number of Extended Address Bits	0 - 8	0
EXP: Map-Overlap Mode	0=HIGH/LOW 1=PROGRAM/ DATA	HIGH
Wait for Target Ready	0=NO 1=YES	NO
Enable Bus-Request Mode	0=NO 1=YES	NO

EMU:Clock Source: Power-up value 0 enables TMS320C2x emulator-card crystal oscillator as clock. Enter 1 to use target-system oscillator. Enter 2 to use the crystal-oscillator package installed at U9 on the Emulator card (refer to Section 9.4.1.2 on Page 9-30.). Use the DES command (Page 4-28) to check your selection.

Delete Software Breakpoint Mode: Select 0 (default) to leave software breakpoints enabled after execution, or select 1 to clear each software breakpoint after encountering it. In default mode, the monitor program executes one instruction after the breakpoint, but execution is *NOT* in real time.

In clear mode, the monitor stops the processor before executing the instruction at the breakpoint address, deletes the breakpoint, then returns control to the keyboard.

Hex Notation: Lets you select the XDS numeric mode.

- 0=NONE accepts and displays all numbers as hexadecimal.
- 1=TI accepts all numbers as decimal unless preceded by one of >, 0X, X, or H; followed by H; or preceded by 0 and followed by H. All hexadecimal numbers display preceded by >.
- 2=X does the same thing, except display preceded by X.
- 3=HS does the same thing, except display followed by H.
- 4=HP does the same thing, except display preceded by H.

Number of Extended-Address Bits: Lets you select the eight signals on the extended-address cable from the Breakpoint/Trace/Timing card individually as data or as addresses. Default is all eight as data.

Map-Overlap Mode: 0=HIGH/LOW lets you map both 32K memory segments on the XDS Memory Expansion/Communications card with the restriction that program and data memory cannot overlap. 1=PROGRAM/DATA allows overlapping, but all mapped addresses must either be between >0000 to >7FFF or between >8000 and >FFFF.

Wait for Target Ready: 0=NO lets the Memory-Expansion card end target-memory access during the second cycle of that access by setting memory-control signals ($\overline{PS}$, $\overline{DS}$, R/$\overline{W}$, and $\overline{STRB}$) to logic 1 after the first cycle. 1=YES forces the card to wait until the target system returns a READY signal, indicating memory-access complete.

Enable Bus-Request Mode: As indicated by the EXP prefix, this prompt interacts with the Memory Expansion/Communications card. The prompt controls the emulator card TMS320C2x global emulator-memory accesses (as defined in its GREG register) by driving its $\overline{BR}$ (Bus Request) and $\overline{DS}$ (Data Memory Strobe) pins low.

If your response to the prompt is the default of 0=NO, then that memory access can go to the Memory Expansion/Communications card, if it has been mapped-in with the MAP command (Page 4-89). If your response is 1=YES, then that access *must* go to target-system memory.

Note that the target system cannot directly access the Memory Expansion/Communications card regardless of this prompt.

Because the value in GREG (which you can set with the GREG register command on Page 4-115) sets a global-memory range from a variable starting address to >FFFF, and because the emulator MAP command can assign a base address from >0000 to >FC00, you can control the global memory access to the emulator memory.

Associated Commands

INTERACTION WITH OTHER COMMANDS OR PARAMETERS		
INTERACTING COMMAND	**INTERACTING PARAMETER**	**INIT PARAMETER AFFECTED**
Number of Extended-Address Bits	BTT	Address #1 Address #2 Address Mask External-Probes Data External-Probes Data Mask
Map-Overlap Mode	MAP	Memory Base Address Number of 1K Blocks

Example This example shows emulation using the target-system oscillator, with no extended-address bits needed.

```
DISPLAY                                                              ENTER
?                                                                    INIT
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]     = INTERNAL   <CR>
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES] = NO         <CR>
      HEX NOTATION [0=NONE, 1=TI, 2=X, 3=HS, 4=HP]  = NONE       <CR>
  BP: NUMBER OF EXTENDED ADDRESS BITS (0-8)         = 0          <CR>
 EXP: MAP OVERLAP MODE [0=HIGH/LOW,1=PROGRAM/DATA]  = HIGH       <CR>
      WAIT FOR TARGET READY [0=NO,1=YES]            = NO         <CR>
      ENABLE BUS-REQUEST MODE [0=NO,1=YES]          = NO         <CR>
?
```

Syntax

Prompt Mode : IPM <TERM>
Parenthesis Mode : IPM(<value>)<TERM>

Description

The IPM command displays the content of a selected program-memory location for review or change. You can then review or change adjacent locations without repeating the command.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Address	0 -FFFF	0000

Address: Selects address to be displayed. To keep the displayed value and look at the next address, press +, the Space bar, or RETURN. To keep that value and look at the previous address, press -. To keep that value and end the command, press ESC or type Q, then press RETURN.

To change the displayed content, type a new value, then press RETURN. If the new value is shorter than the existing value, erase the extra with the Space bar or the terminal DELETE key. After entry, you can step forward or backward through memory as described above.

In decimal mode, you can enter address and/or values as decimal numbers, however, the monitor program converts them to hexadecimal.

Associated Commands IDM (Inspect Data Memory), Page 4-69.

Example 1

This example displays the content of memory locations 0F0D through 0F0F, then changes the content of 0F0F.

DISPLAY	ENTER
?	IPM<CR>
IPM	
ADDRESS = 0000	0F0D <CR>
0F0D = 34F1	<CR>
0F0E = 4251	<CR>
0F0F = 425D	425E <CR>
0F10 = 4250	<ESC>
?	

Example 2

This is the same example, but with the decimal mode selected. You enter the addresses as +3853 through +3855, then enter the data change at +3855 as +16990.

DISPLAY	ENTER
?	DEC IPM<CR>
DEC	
IPM	
ADDRESS = >0000	3853 <CR>
>0F0D = +13393	<CR>
+>0F0E = +16977	<CR>
+>0F0F = +16989	16990<CR>
+>0F10 = +16976	Q<CR>
?	

Syntax

Prompt Mode : IPORT<TERM>
Parenthesis Mode : IPORT(<value>, ..., <value>)<TERM>

Description

The IPORT command sets baud rate, parity, number of stop bits, and number of bits per character for XDS Port C (Programmable Read-Only Memory (PROM) Programmer or logging device) or Port D (Host computer).

For Port C RS-232C configuration, refer to Section 7.4 or Section 7.5.

For Port D RS-232C configuration, refer to Section 9.2 for general Memory Expansion/Communications-card switch settings, and to Section 7.6, Section 7.7, Section 7.8, and Section 8.7.1 for specific settings.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Port	0 = HOST("D") 1 = LOG("C")	HOST
Baud	0 = 19.2K 1 = 9.6K 2 = 4.8K 3 = 2.4K 4 = 1.2K 5 = 600 6 = 300 7 = 110	19.2K
Parity	0 = OFF 1 = ODD 2 = EVEN	OFF
Stop Bits	0 = 2 1 = 1	2
Bits Per Character	0 = 7 1 = 8	7

Port: This parameter lets you select configuration of XDS I/O port C for a Programmable Read-Only Memory (PROM) programmer or logging device or select Port D for an external computer.

Baud: This parameter sets the transmit or receive data rate. At XDS system power-up, the autobaud sequence sets all ports to the baud rate of the terminal connected to port A.

For manual baud-rate setting, you can enter the parameter position number (0-7) or the actual values displayed in the prompt.

Parity: This parameter sets the parity bit. If no parity selected, transmitted data has no parity bit added and the monitor ignores the parity bit in received data.

Stop Bits: Sets number of stop bits.

Bits Per Character: Sets number of data bits. TI compressed format requires eight bits per character, other formats require either seven or eight.

Associated Commands None

Example This example configures the XDS to conform to the 4800-bps baud rate of a host computer system.

```
DISPLAY                                                          ENTER
?                                                                IPORT<CR>
IPORT
 PORT [0=HOST("D"), 1=LOG/PROM("C")]                 = HOST      <CR>
 BAUD[19.2K,9.6K,4.8K,2.4K,1.2K,600,300,110]         = 19.2K     2<CR>
 PARITY[0=OFF, 1=ODD, 2=EVEN]                        = OFF       <CR>
 STOP BITS [0=2, 1=1]                                = 2         <CR>
 BITS/CHAR [0=7, 1=8]                                = 7         <CR>

?
```

Syntax Prompt Mode : IPRM<TERM>

Description The IPRM command resets each command power-up value without executing anything or changing any hardware. This lets you restore these parameters without having to turn the machine on and off.

The restored values reappear as current values the *next* time you call the command in the Prompt mode.

Parameters None.

Associated Commands None.

Example This example assumes that you have used the INIT command (Page 4-75) to set the clock source to an external oscillator, then used the SAVE command (Page 4-99) to store that selection (as well as all the other things that command does).

```
DISPLAY                                                              ENTER
?                                                                    INIT(2,Q)
                SAVE<CR>
?                                                                    INIT<CR>
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]     = OSC           <CR>
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES] = NO            Q<CR>
?                                                                    IPRM<CR>
IPRM
?                                                                    INIT<CR>
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]     = INTERNAL      <CR>
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES] = NO            Q<CR>
?                                                                    LOAD
LOAD
?                                                                    INIT<CR>
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]     = INTERNAL      <CR>
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES] = NO            Q<CR>
```

The clock-selection parameter display is 0, the default value; however, the XDS continues to run with the external oscillator unless you now actually enter another value.

Syntax Prompt Mode : IR<TERM>

Description The IR command displays the content of all processor internal registers for review or change. Changes become the current value for the register when execution resumes.

In the decimal mode, register values display in decimal or magnitude numbers if you precede the command name with DEC or MAG.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
PC (Program Counter)	0-FFFF	0000
ST0 (Status Register 0)	0-FFFF	variable
ST1 (Status Register 1)	0-FFFF	variable
T (Temporary Register)	0-FFFF	variable
P (Product Register)	0-FFFFFFFF	variable
ACC (Accumulator)	0-FFFFFFFF	variable

Associated Commands MR (Modify Registers), Page 4-94.

Example 1 Hexadecimal mode

```
DISPLAY                          ENTER
?                                IR <CR>
IR
 PC    = 0000                    <CR>
 ST0   = E600                    <CR>
 ST1   = E3F0                    <CR>
 T     = 1234                    <CR>
 P     = 00569023                <CR>
 ACC   = 00000001                <CR>
?
```

Example 2 Decimal mode

```
DISPLAY                          ENTER
?                                DEC IR <CR>
IR
 PC    = 0000                    <CR>
 ST0   = E600                    <CR>
 ST1   = E3F0                    <CR>
 T     = 1234                    <CR>
 P     = 00569023                <CR>
 ACC   = 00000001                <CR>
?
```

Syntax

Prompt Mode : IT<TERM>
Parenthesis Mode : IT(<value>)<TERM>

Description

The IT command displays trace samples from the trace buffer, 19 lines at a time, starting with a selected trace. If the trace buffer contains more than 19 samples, pressing RETURN displays your selection of the next (default) or previous *nnnn* samples where *nnnn* is 0 - 2047.

> **Note:**
>
> The IT command always works in decimal, never in hexadecimal.

The monitor program algebraically adds the value of *nnnn* to the index number of the selected trace. The default value is +19, but you can change it at the end of any screen. For example, if you select trace 0 and press RETURN, you would see traces 0 -18 (the first 19), if the default still applies. If you now want to see trace 500, enter 500 at the block cursor and press RETURN. Your screen now displays traces 500 - 518, if the buffer has that many; or the last 18, if it doesn't.

Pressing RETURN again adds another 500 to the index, so that you now would see traces 1000 - 1018 (again assuming that they exist).

If you are paging forward through the trace buffer, and the last display has less than 19 samples, the command displays the last sample on line 19 and fills in the display with the preceding 18 lines. For example, if the trace buffer has 29 (decimal) samples, the first screen would have samples 0 - 18 and the next (and last) screen would have samples 10 (decimal) through 28 (decimal) with samples 10 through 18 repeated from the first screen.

If you are paging backward through the trace buffer, and the last display has less than 19 samples, the command displays the first sample on line 1 and fills in the display with the preceding 18 lines. Using the example above, the first screen would display samples 10 through 28, the second (and last) screen would display samples 0 through 18.

Pressing ESC at any time ends the command and the display. At the end of any screen display, you can enter the following controls:

B<CR> Displays bottom of trace buffer or a screen of trace samples ending with the last sample taken.

<CR> Displays next or previous *nnnn* trace samples depending on last ±*nnn* entry. Default is next 19 samples (+19).

F<CR> Searches trace buffer for next occurrence of conditions set by the FT (Find Trace) command. Refer to Page 4-53.

L<CR> Displays trace sample with Last-Event flag (*EVT) in its CYCLE column "windowed" with preceding and following samples.

Mnnnn Indicates sample to Mark with IBTT Trace Mark option.

Q Exits IT command.

T<CR> Displays top of trace buffer or a screen of trace samples starting with the first sample taken.

continued

+nnnn Indicates that pressing RETURN displays the sample number that is *nnnn* greater than the number of the sample presently displayed at the top of the screen (if that sample exists).

-nnnn Indicates that pressing RETURN displays the sample number that *nnnn* less than the number of the sample presently displayed at the top of the screen (if that sample exists).

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
First Sample (0 = Oldest Sample)	0 - 7FF	0

Sets first sample to be displayed, where 0 is the oldest sample (first sample taken) and >7FE (2046 decimal, which is how the indices display) is the most recent sample taken. Entry of >7FF causes the monitor program to display the last 19 samples taken.

Associated Commands DT (Display Trace), Page 4-42. FT (Find Trace), Page 4-53.

Example Refer to Section 3.7.5.

Syntax

Prompt Mode : ITR<TERM>
Parenthesis Mode : ITR(<value>,<value>)<TERM>

Description

The ITR command lets RAM installed on the Emulator card respond to memory-access cycles as if it were installed in your target system.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Program Memory RAM	0=OFF 1=ON	OFF
Data Memory RAM	0=OFF 1=ON	OFF

Program Memory RAM: ON lets onboard memory selected as program RAM respond to memory-access cycles. OFF prevents response.

Data Memory RAM: ON enables onboard memory selected as data RAM to respond to memory-access cycles. Selecting OFF prevents response.

Associated Commands None

Example

This example enables the Emulator-card program memory but leaves the data memory disabled.

```
DISPLAY                                          ENTER
?                                                ITR<CR>
ITR
 PROGRAM MEMORY RAM [0=OFF,1=ON] = OFF           1 <CR>
 DATA MEMORY RAM [0=OFF,1=ON]    = OFF           <CR>

?
```

Caution:

Be careful to avoid bus conflicts with the target system memory.

Syntax Prompt Mode : LOAD<TERM>

Description The LOAD command reads all command parameters stored in the TMS320C2x Emulator's 2K non-volatile memory back into the emulator and sets them up to be default values.

This command does *not* actually change anything. To restore parameters, you must execute the command and accept all the stored values as default values.

This command also does **not** restore values stored in data or program memory.

Parameters None

Associated Commands SAVE, Page 4-99.

Example This example reloads all parameters saved by means of the SAVE command.

DISPLAY
```
?
LOAD
 INIT.MAP
```

ENTER
```
LOAD<CR>
```

Syntax

Prompt Mode : LOG<TERM>
Parenthesis Mode : LOG(<value>,<value>)<TERM>

Description

The LOG command enables or disables the logging device, usually a printer.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Log Device	0 = OFF 1 = ON	OFF
Line Feed with Backspace	0 = NO 1 = YES	NO

Log Device: Enables or disables transmission to the logging device connected to XDS/22 Port C.

Line Feed with Backspace: Enables emulator to issue a line feed with each backspace to prevent overwriting information.

INTERACTION WITH OTHER COMMANDS OR PARAMETERS		
INTERACTING COMMAND	INTERACTING PARAMETER	LOG PARAMETER AFFECTED
DL	Destination (PROM)	Log Device (Set to 0)

Associated Commands SAVE, Page 4-99.

Example

Enables printer and specifies a linefeed with each output line.

```
DISPLAY                                                    ENTER
?                                                          LOG<CR>

LOG DEVICE [0=OFF, 1=ON]                 = OFF             1<CR>
LINEFEED WITH BACKSPACE [0=NO, 1=YES] = NO                 1<CR>
```

Syntax Prompt Mode : MAG<TERM>

Description The MAG command causes the value of the *next command only* in a command string to display as an unsigned decimal number formed by converting the value's binary representation, including the sign bit. The MAG command works *only* if you have selected the decimal mode with the INIT command.

This format change remains in effect until changed by a DEC or HEX command preceding the same command or until you start a new session (default display format is hexadecimal).

Parameters None

Associated Commands DEC (Decimal), Page 4-27. HEX (Hexadecimal), Page 4-61.

Example

```
?                                                                INIT
INIT
 EMU: CLOCK SOURCE [0=INTERNAL, 1=TARGET, 2=OSC]     = INTERNAL  <CR>
      DELETE SOFTWARE BREAKPOINT MODE [0=NO, 1=YES] = NO        <CR>
      HEX NOTATION [0=NONE, 1=TI, 2=X, 3=HS, 4=HP]  = NONE      1<CR>
  BP: NUMBER OF EXTENDED ADDRESS BITS (0-8)         = 0         <CR>
 EXP: MAP OVERLAP MODE [0=HIGH/LOW,1=PROGRAM/DATA]  = HIGH/LOW  <CR>
      WAIT FOR TARGET READY [0=NO,1=YES]            = NO        <CR>
      ENABLE BUS-REQUEST MODE [0=NO,1=YES]          = NO        <CR>
?                                                                PC<CR>
 PC=±ddddd

?                                                                MAG PC<CR>
  PC=ddd...ddd
?
```

After declaring the decimal mode with the INIT command, the PC display is a signed decimal number. After entering the MAG command, the PC display is an unsigned decimal number. If the signed decimal number is positive, the magnitude decimal number is the same; if the signed decimal number is negative, the magnitude decimal number may be much larger. Further program-counter display in the DHS, DPS, DR, IR, or MR commands will be in this format unless:

- You change it back to decimal (with the DEC command, Page 4-27)
- You change it to hexadecimal (with the HEX command, Page 4-61),
- You start a new session (always starts in hexadecimal).
- You change a format within the IR or MR commands with CNTL/X.

Syntax

Prompt Mode : MAP<TERM>
Parenthesis Mode : MAP(<value>,<value>,...<value>)<TERM>

Description

The MAP command lets you define expansion memory. You can substitute emulator expansion memory for target-system data- or program-memory, either RAM or ROM, in 1k word blocks. The command lowers any base address not specified as an exact multiple of 1K to the nearest such multiple.

Memory expansion can be used as substitution memory for onboard target RAM. It cannot be used for any memory that physically exists on a TMS320C2x device since the command does not generate an external access.

The XDS normally runs expansion-memory DRAM's with two-cycle accesses below 5 MHz or with four-cycle accesses above 5 MHz; however, any conflict between a regular access and a DRAM refresh access may add one cycle below 5 MHz or two cycles above 5 MHz.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Memory	0=Program 1=Data	Program
Base Address (1K Boundary)	0000-FC00	0000
Hexadecimal number of 1K-word blocks	0 - 40	00
Action	0=Unmap 1=ROM 2=RAM 3=Copy	Unmap
Number of additional wait states	0 - 7	0

Memory [0=Program, 1=Data]: Defines memory type to be used for substitution.

Base Address: Defines starting point for memory block. In decimal mode, you can enter this as a decimal number, however, the monitor program converts it to a hexadecimal number.

Hexadecimal number of 1K-word blocks: Defines size of mapped memory.

Action

- 0=Unmap this block
- 1=Map block as ROM
- 2=Map block as RAM
- 3=Map block as ROM and fill it with content of target-memory space.

> **Note:**
>
> If you assign a particular memory as RAM. enter data, then UNMAP the area (enter 0 in this prompt), then reassign the area as RAM (enter 2 in this prompt), your data *may* return intact.

Number of additional wait states: Number of additional wait states required for your target-system memory devices.

Associated Commands DMAP (Display Memory Map), Page 4-34. INIT (Initialize), Page 4-75.

INTERACTION WITH OTHER COMMANDS OR PARAMETERS		
INTERACTING COMMAND	**INTERACTING PARAMETER**	**PARAMETER AFFECTED**
INIT	Map-Overlap Mode Target Ready Bus Request	

Example

This example maps a 1K block of program memory as ROM into emulator expansion memory starting at 0. The memory requires three additional wait states.

```
DISPLAY                                                ENTER
?                                                      MAP<CR>
MAP
 BASE ADDRESS (1K BOUNDARY)              = 0           <CR>
 NO. OF 1K-WORD BLOCKS IN HEX            = 0           1<CR>
 MEMORY [0=PROGRAM, 1=DATA]              = PROGRAM     <CR>
 [0=UNMAP,1=ROM,2=RAM,3=COPY]            = UNMAP       1<CR>
 NO. OF ADDITIONAL WAIT STATES (0-7) = 0               3<CR>
```

Syntax

Prompt Mode : MDM<TERM>
Parenthesis Mode : MDM(<value>,value>)<TERM>

Description

The MDM command lets you change the content of a specified data-memory location.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Address	0-FFFF	0000
Data	0-FFFF	0000

Address: Selects data-memory address to be changed.

> **Note:**
>
> Be sure the address can really be in data memory. Refer to TMS320C25 User's Guide (SPRU012).

Data: New value for specified memory location.

Associated Commands MPM (Modify Program Memory), Page 4-93. MIO (Modify I/O), Page 4-92.

Example

This example places a value of >F980 in location >400.

DISPLAY	ENTER
?	MDM<CR>
MDM	
ADDRESS=0000	400 <CR>
DATA =0000	F980 <CR>
?	

Syntax

Prompt Mode : MIO<TERM>
Parenthesis Mode : MIO(<value>,<value>)<TERM>

Description

The MIO command lets you write a value to a specified port.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Port Address	0 - F	0
Data	0 - FFFF	0000

Port Address: Specifies port number (address).

Data: Value to go to specified port.

Associated Commands MDM (Modify Data Memory), Page 4-91. MPM (Modify Program Memory), Page 4-93.

Example

This example writes a value of >00FF to port 1.

DISPLAY	ENTER
?	MIO<CR>
MIO	
PORT ADDRESS = 0	1 <CR>
DATA = 0000	00FF<CR>
?	

Syntax

Prompt Mode : MPM<TERM>
Parenthesis Mode : MPM(<value>,<value>)<TERM>

Description

The MPM command lets you change the content of a specified program-memory location.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Address	0 - FFFF	0000
Data	0 - FFFF	0000

Address: Specifies program-memory address where content to be changed.

Data: New hexadecimal value for that address.

Associated Commands MDM (Modify Data Memory), Page 4-91. MIO (Modify I/O Port), Page 4-92.

Associated Commands

Example

This example places a value of >FF80 in program-memory location >0100.

DISPLAY	ENTER
?	MPM<CR>
MPM	
ADDRESS = 0100	<CR>
DATA = 0000	FF80<CR>
?	

Syntax

Prompt Mode : MR<TERM>
Parenthesis Mode : MR(<value>,<value>, ..., <value>)<TERM>

Description

The MR command lets you display or change the program counter, status registers, temporary register, product register, and the accumulator.

The monitor program also stores the parameters set by each execution of the MR command. At the next execution of the command *without* parameter changes, the program restores the register values from that *last* MR-command -- regardless of any changes made individually by the IR (Inspect Register) command or program execution in the meantime.

For that reason, note that many examples in this Manual use `MR;` preceding a RUN command. As described in Section 5.1.3.2, entering a semicolon after a command name causes it to execute with current parameters, but not to display those parameters.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
PC (Program Counter)	0-FFFF	0000
ST0 (Status Register 0)	0-FFFF	variable
ST1 (Status Register 1)	0-FFFF	variable
T (Temporary Register)	0-FFFF	variable
P (Product Register)	0-FFFFFFFF	variable
ACC (Accumulator)	0-FFFFFFFF	variable

Associated Commands IR (Inspect Register), Page 4-82.

Example

This example changes the values in some processor registers.

DISPLAY	ENTER
?	MR <CR>
MR	
PC = 0000	1<CR>
ST0 = 2604	<CR>
ST1 = 47F0	<CR>
T = 1234	<CR>
P = 00569023	<CR>
ACC = 00000001	<CR>
?	

To execute a program in a command procedure with a predefined set of register values, use the MR command as follows:

DISPLAY	ENTER	COMMENT
?	MR;RUN<CR>	
MR;RUN		
RUN		
RUNNING		

The program starts running at the program-counter location established by the *last* execution of the MR command, default of 0000.

Syntax Prompt Mode : RCC<TERM>

Description The RCC command resets cursor-control functions to their power-up values which cancels cursor-control functions defined by the ICC command.

Parameters None

Associated Commands ICC (Initialize Cursor Control), Page 4-67.

Example

DISPLAY	ENTER
?	RCC<CR>

?

After execution of the RCC command, move the cursor with the following keys:

- \- (minus sign) To recall preceding prompt.
- \+ To step forward to next prompt.
- < To move the cursor left.
- > To move the cursor right.

Syntax

Prompt Mode : RESTART<TERM>
Parenthesis Mode : RESTART(<value>)<TERM>

Description

The RESTART command resets the emulator to the power-up, cold-start condition without having to turn the emulator off.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Are You Sure ?	0 = NO 1 = YES	0

Associated Commands None.

Example

DISPLAY	ENTER
?	RESTART<CR>
RESTART ARE YOU SURE [0=NO, 1=YES] = NO	1<CR>
?	

The emulator is now set to power-up condition, with all parameters and registers set to power-up values. Complete the normal power-up sequence by pressing RETURN twice to set the terminal baud rate with the monitor-program Autobaud feature. The terminal screen then displays the power-up banner and the command menu.

Syntax Prompt Mode : RTR<TERM>

Description The RTR command starts the emulator running at target-system reset. The monitor program does not begin execution until the emulator logic detects a rising voltage on the target-connector $\overline{RS}$ pin; however, the screen displays RUNNING even before this level appears.

Execution of the RTR command does the following:

- **Normal Run Mode:** Clears trace buffer, begins a new trace, and re-enables all hardware breakpoints.
- **Alternate-Run Mode:** Reprograms Breakpoint/Trace/Timing card to the conditions set by the BTT and IBTT commands.

Parameters None

Associated Commands None

Example

DISPLAY	ENTER
? RTR RUNNING	RTR<CR>

Syntax Prompt Mode : RUN<TERM>

Description The RUN command executes the content of program memory starting at the current program-counter setting. Be sure to set it and any other applicable registers before execution.

Program execution ends upon any of the following:

- Reaches end of program
- Reaches end of program memory
- Reaches address set as software breakpoint (Section 6.10)
- Satisfies conditions set for hardware breakpoint (Section 6.2)
- Selected number of traces collected (Section 6.3)
- Any key on terminal keyboard pressed
- Parity error, etc. (abnormal end).

Execution of the RUN command does the following:

- **Normal Run Mode:** Clears trace buffer, begins a new trace, and re-enables all hardware breakpoints.
- **Alternate-Run Mode:** Reprograms breakpoint/trace function to the conditions set by the BP, BPM, TR, and TRM commands.

In multiprocessor mode, the RUN command starts the emulator in the foreground if it meets the other criteria described in Section 8.

Parameters None

Associated Commands CRUN (Continue Run), Page 4-22. GRUN (Group Run), Page 4-59.

Example

```
DISPLAY                          ENTER        COMMENT
?                                RUN<CR>
RUN
 RUNNING
  (Halt Code)
    .
  (REGISTER DISPLAY)
     .
?                                RUN<CR>
RUN
 RUNNING                         <any key>
 KEY
 PC= F01C ST0 = 2600 ST1= 43F0
?
```

Syntax Prompt Mode : SAVE<TERM>

Description The SAVE command stores all command parameters and the content of the alternate-command buffer in non-volatile memory so that they can be restored at the next system power-up.

Parameters None

Associated Commands LOAD, Page 4-86.

Example

DISPLAY	ENTER
?	SAVE<CR>

Syntax Prompt Mode : SNAP,<command string><TERM>

Description The SNAP command "freezes" display of parameter names, but not parameter values, on your terminal's screen display.

The SNAP command **can only be used** as the first command in a command string and **only makes sense** when one or more of the remaining commands display parameters changed by execution of your application program.

When so-used, the remaining commands execute in sequence with a normal display, including scrolling, until 22 lines have appeared on the screen. At that time, the screen stops scrolling and the parameter names displayed at that time remain displayed (which is also normal).

What is different, however, is that, if your application program continues to execute and changes any of the displayed parameter values, those values *do* change. Any changes on the lines that scrolled off the screen do not appear.

If the remaining commands cause less than 22 display lines, then the parameters, but not the values, on all displayed lines "freeze."

You must execute the ICC (Initialize Cursor Control) command before executing the SNAP command - even if you just accept its default values.

Parameters None

Associated Commands ICC (Initialize Cursor Control), Page 4-67.

Example

```
DISPLAY                                                    ENTER
?                                                          DR <CR>
DR
 PC =0000  T  =0000
 ST0=E600  P  =00000000
 ST1=03F0  ACC=FFFFFFFF
?                                                          SNAP,SS*25 <CR>
SS
 PC=0002  ST0=0000 ST1=03F0 NEXT: 0002 LRLK AR0, >0314
   .          .         .         .          .        .
   .          .         .         .          .        .
 PC=0027  ST0=0000 ST1=03F0 NEXT: 0002 LRLK AR0, >0314
?
```

where the display above is similar to, but not necessarily the same as, what you may see. The program counter does increment as shown, but the remaining values may or may not be the same and may or may not change, depending upon what's in memory at the time.

Syntax Prompt Mode : SS<TERM>

Description The SS command executes a program stored in emulator memory one instruction at a time and displays the resulting register content along with the address and reverse-assembly of the next instruction.

Execution of the SS command after a change of breakpoint event count, delay count, or trace-count parameters clears the trace buffer. The buffer then begins storing new traces with the current instruction being executed.

Press ESC to end single-step execution.

You can also use the XDS asterisk repeat feature (Section 5.1.3) with the SS command to execute a number of steps. Enter SS*<CR> to repeat until you press ESC to stop, or enter SS*nnnn<CR> to do *nnnn* steps.

If you have selected decimal mode with the INIT command and have set the program counter to display in decimal or magnitude, then that is its display format for this command.

Parameters None

Associated Commands None.

Example Execute a program starting in location >20A one step at a time.

DISPLAY	ENTER
?	PC=20A<CR>
?	SS<CR>

```
SS
 PC=020A ST0= 0000 ST1= 0000 NEXT: 020A  D001 LALK 0100,0
```

Repeat single-step execution of program from where last step executed:

DISPLAY	ENTER
?	SS*<CR>

```
 PC=020B ST0= xxxx ST1= xxxx NEXT: 020B  code mnemonic operand(s)
 PC=020C ST0= xxxx ST1= xxxx NEXT: 020C  code mnemonic operand(s)
 PC=020D ST0= xxxx ST1= xxx  NEXT: 020D  code mnemonic operand(s)
                                                      <ESC>
```

Pressing ESC ends the command.

Syntax

Prompt Mode : SSB<TERM>
Parenthesis Mode : SSB(<value>)<TERM>

Description

The SSB command sets one memory address as a software breakpoint. Program action upon reaching the address depends on the Delete Software Breakpoint mode set by the INIT command (Page 4-75).

1) If in the default mode, the program services the breakpoint upon reaching the address, then executes one more instruction. The breakpoint remains sets.

2) If in the Delete mode, the processor halts upon reaching the breakpoint, then automatically clears the breakpoint.

You can execute the command up to 10 times to define up to 10 software breakpoints. You get an error message if you try an 11th time.

To restart your program after a software breakpoint, enter a RUN or CRUN command. RUN resets the trace buffer (if you are also using hardware breakpoint or tracing), CRUN does not.

> **Note:**
>
> This command is a TMS320C2x Emulator card function. It does NOT require installation of a Breakpoint/Trace/Timing card.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Breakpoint Address	0 - FFFF	0

Breakpoint Address: Sets program memory location to be defined as a software breakpoint. The terminal screen displays:

```
1125  SOFTWARE BREAKPOINT OVERFLOW
```

if you attempt to set more than 10 breakpoints.

The XDS treats duplicate software breakpoints assigned to the same address as one breakpoint.

Entering a RUN or CRUN command continues program execution from the point where the program halted.

In Alternate-Run Mode (ARM), the following applies:

- A software breakpoint returns control to the monitor which lets you execute breakpoint/trace commands, instead of a RUN halt. A TMS320C25 processor stops executing code upon reaching a software breakpoint. You must enter another RUN-type command to restart it.
- You can execute the SSB command after execution of the ARM command if the Memory-Commands-Permitted parameter in the ARM command is enabled. Otherwise, end processor execution by means of the STOP, DISARM, or RUN (with a breakpoint enabled) commands before executing SSB.

Associated Commands CSB (Clear Software Breakpoint), Page 4-23.

Example

Enter software breakpoints at locations 205 and 220:

DISPLAY	ENTER
?	SSB<CR>
SSB BREAKPOINT ADDRESS = 0000	205<CR>
?	SSB<CR>
SSB BREAKPOINT ADDRESS = 0205	220<CR>
?	DSB<CR>
DSB 0205 0220	
?	

Syntax Prompt Mode : STOP<TERM>

Description The STOP command halts processor execution after execution of the RUN command if the emulator is operating in Alternate-Run Mode.

Parameters None

Associated Commands ARM (Alternate-Run Mode), Page 4-13.

Example

```
DISPLAY                                                    ENTER
?                                                          ARM<CR>
ARM
 MEMORY COMMANDS PERMITTED? (0=NO, 1=YES) = 0              <CR>

?                                                          RUN<CR>
RUN
 RUNNING
    .
    .                                                      <any key>
 KEY RUNNING
    .                                                      STOP<CR>
?
```

In the Alternate-Run mode, pressing any key does not halt processor execution. Only the STOP command causes a halt.

Syntax Prompt Mode : THALT<TERM>

Description The THALT command interrupts RUN or background operations of all emulators in a multiprocessing chain.

Parameters None

Example

DISPLAY	ENTER
?	THALT<CR>

?

All emulators in the multiprocessing chain halt. One emulator assumes control.

Syntax Prompt Mode : TRUN<TERM>

Description The TRUN command enables all emulators in a multiprocessing chain to begin running. Those configured to start simultaneously by the IMD command start within one instruction. All others start immediately.

In a single-emulator application, or if you have not initialized a multiprocessing chain, entering this command causes this error message to display:

```
ERROR 1701  MULTIPROCESSING NOT INITIALIZED
```

Parameters None

Associated Commands IMP (Initialize Multiprocessor Mode), Page 4-74.

Example

DISPLAY
```
?
TRUN
   BACKGROUND  or AUTOPOLLING
      ??             ???
```

ENTER
```
TRUN<CR>
```

All emulators in the multiprocessing chain start to run. The display depends on the parameter of the IMD (Initialize Multiprocessing) command.

Syntax

Prompt Mode : UL<TERM>
Parenthesis Mode : UL(<value>, <value>, ..., <value>)<TERM>

Description

The UL command sets parameters for uploading object code from the emulator to a host system or external device.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Start Address	0 - FFFF	0000
End Address	0 - FFFF	0000
Format (Object Code Format)	0 = TI 1 = TICOM 2 = TEK 3 = INTEL 4 = MR	TI
Data Format	0 = WORD 1 = HI BYTE 2 = LO BYTE	WORD
Source	0 = PROGRAM 1 = DATA	PROGRAM
Destination	0 = HOST 1 = PROM 2 = USER	HOST
Protocol	0 = NONE 1 = TEK 2 = ASR 3 = VAX	NONE

Start Address: Sets address in external device for storage of uploaded data to begin.

End Address: Sets address in external device for storage of uploaded data to end. The command fills any extra locations between start and ending addresses with zeros.

Format: Specifies object-code file format. Values are:

0 = TI	TI normal ASCII format
1 = TICOM	TI compressed format
2 = TEK	Tektronix hexadecimal format
3 = INTEL	Intel Intellec 8/MDS format
4 = MR	Motorola Exorcisor format

Data Format: Lets you upload one data word or one byte at a time for object-code formats other than the above. For byte transfer, 1=HI BYTE sends the high-order byte first, 2=LO BYTE sends the low-order byte first.

Source: Specifies memory to upload from.

Destination: Defines uploaded data destination. Values are:

0 = HOST — Establishes host computer as destination of data to be uploaded from emulator through Port D.

1 = PROM — Lets data be transmitted through Port C into PROM programmer or logging device.

2 = USER — Selects your terminal as destination of data to be uploaded from emulator through Port A. For intelligent terminals or personal computers as terminals.

Protocol: Sets protocol for data transfers. Values are:

0 = NONE — No protocol defined. You must define control characters by means of the IHC command before using this protocol.

1 = TEK — Tektronix protocol enabled. You must define control characters with the IHC command before using this protocol. Returned messages are:

ASCII 0 - Record transmitted without error.
ASCII 7 - Error detected; retransmit.

2 = ASR — ASR terminal protocol enabled. Control characters are:

CNTL/R - Start download
CNTL/S - End download
CNTL/Q - Start Upload
CNTL/@ - End Upload
CNTL/P - Pass through next control character.

3 = VAX — VAX or PDP-11 protocol enabled. Values are:

CNTL/A - Start Upload
CNTL/W - End Upload
CNTL/V - Start Download
CNTL/X - End Download
CNTL/P - Pass-through Character

Note:

For a TEK protocol, the emulator expects to receive ASCII 0 (zero) <CR> from the host if it received the record successfully, or ASCII 7 <CR> if it didn't. For the latter, the emulator monitor program retransmits the record up to 15 times. If it hasn't transferred after 15 times, the monitor program aborts the upload, displays "UPLOAD ABORTED", and redisplays the ? prompt.

Associated Commands IHC (Initialize Host Control), Page 4-70.

Example

Uploading to Host System Through Port D.

Assume the following:

- Object code resides in F006 - FFFF
- Standard TI format object code
- Destination: Host port
- Standard VAX protocol

```
DISPLAY                                                          ENTER
?                                                                UL<CR>
UL
 START ADDRESS                                          = 0000   F006<CR>
 END ADDRESS                                            = 0000   FFFF<CR>
 FORMAT [0=TI, 2=TEK, 1=TICOM, 3=INTEL, 4=MR]           = TI     <CR>
 DATA FORMAT [0=WORD,1=HI BYTE,2=LO BYTE]               = WORD   <CR>
 SOURCE [0=PROGRAM, 1=DATA]                             = PROGRAM <CR>
 DESTINATION  [0=HOST, 1=PROM, 2=USER]                  = HOST   <CR>
 PROTOCOL [0=NONE, 1=TEK, 2=ASR, 3=VAX]                 = NONE   <CR>

?
```

Syntax

Prompt Mode : XA<TERM>
Parenthesis Mode : XA(<value>,<value>)<TERM>

Description

The XA command lets you enter and assemble a TMS320C2x assembly-language source program from the keyboard.

The first two command prompts let you set the starting address for assembly and select whether you want to use the existing symbol table or not. This latter feature lets you add on to an existing program.

After responding to these prompts, the command displays:

```
LINE   ADD     DATA    LABEL   MNEM      OPERANDS   COMMENT
                       []
```

ready for entry of assembly-language statements.

An assembly-language statement consists of a label field, command field, operand field, and comment field.

The following rules apply when entering source code from the keyboard:

- The label field is optional, however, if used, its first character must go in the cursor position.
- If no label used, start command field in column 2 (Press Space bar one time); or, preferably, space cursor under MNEN heading.
- Separate first operand from command field with at least one space. Preferably enter it under the OPERANDS heading.
- Separate operands with a comma.
- Separate comment(s) from operand field with at least one space. Preferably enter under the COMMENT heading. Comments are not stored and do not appear in the reverse assembly.

To end assembly, enter an END directive as a mnemonic (preferred) or press CNTL and E together. Pressing ESC will *not* work.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Start Address	0 - FFFF	0
Use Old Symbol Table	0 = NO 1 = YES	NO

Start Address: Sets beginning address of object program to be assembled.

Use Old Symbol Table: Lets values previously entered into symbol table be used again, if desired.

Associated Commands XRA (Reverse Assembler), Page 4-112.

Example This program stores sequential values from -10 to +10 in on-chip block B1 starting at >300.

```
DISPLAY                                                    ENTER

?                                                          ITR<CR>
ITR
 PROGRAM MEMORY RAM [0=OFF,1=ON]      = OFF                1 <CR>
 DATA MEMORY RAM [0=OFF,1=ON]         = OFF                <CR>
?                                                          XA<CR>
XA
  START ADDRESS                       = 0000               20<CR>
  USE OLD SYMBOL TABLE [0=NO, 1=YES] = NO                  <CR>

LINE   ADD    DATA   LABEL   MNEM     OPERANDS  COMMENT

0001   0020                 IDT      /EXAMPLE/
0002   0020          INIT   EQU      $
0003   0020   D200          LRLK     AR2,-10   INIT start
       0021   FFF6
0004   0022   D100          LRLK     AR1,>300  INIT starting pointer
       0023   FFF6
0005   0024   D000          LRLK     AR0,>314  INIT ending point
       0025   FFF6
0006   0026   5589          LARP     AR1       Save values in AR1
0007   0027          *
0008   0027          LOOP   EQU      $
0009   0027   72AA          SAR      AR2,*+,AR2 Save value in data mem
0010   0028   55A9          MAR      *+,AR1     Increment value
0011   0029   CE52          CMPR     >2          if not done
0012   002A   F880          BBZ      LOOP        ;then repeat loop
       002B   0007
0013   002C   FF80   DONE   B        DONE 1     Finished
       002D   0000
       000D   >000C RESOLVED
0014   000E                 END

ERRORS   WARNINGS   UNRESOLVED

 0000     0000       0000

?
```

Syntax

Prompt Mode : XRA<TERM>
Parenthesis Mode : XRA(<value>,<value>)<TERM>

Description

The XRA command lets you inspect a memory location and see not only its content, but the assembly-language mnemonic and operands that probably led to that content.

The XRA command does not recreate any comments or labels that might have been included with the original statement.

Parameters

PARAMETER	PARAMETER VALUES	POWER-UP VALUE
Start Address	0 - FFFF	0000
Lines Of Output	0 - FFFF	0000

Start Address: Sets beginning address of program to be reverse-assembled.

Lines Of Output: Sets number of lines of object program to be reverse-assembled.

Associated Commands XA (Assembler), Page 4-110.

Example

Executes reverse assembler to show code for program shown in XA command example:

```
DISPLAY                                          ENTER
?                                                XRA<CR>
XRA
  START ADDRESS           = 0000                 <CR>
  LINES OF OUTPUT         = 0000                 6<CR>
  0020   D200   LRLK AR2,>FFF6
  0022   D100   LRLK AR1,>0300
  0024   D000   LRLK AR0,>0314
  0026   5589   MAR  *,AR1
  0027   72AA   SAR  AR2,*+,AR2
  0028   55A9   MAR  *+,AR1
               END

?
```

Note that the reverse-assembly does not include labels or comments.

Syntax

Prompt Mode : XTIME<TERM>
Parenthesis Mode : XTIME(<value>, ..., <value>)<TERM>

Description

This command is available only with a Breakpoint/Trace/Timing card installed. The XTIME command displays a sample of either memory use or program execution per address range. This command repeatedly enters the Run mode, accumulates time in timers 1 and 2, then halts and displays the percentage of total run time in each address range. This command starts from the current program-counter value; therefore, you must set the program counter with the IR (Inspect Registers), MR (Modify Registers), or PC (Set Program Counter) commands before using XTIME.

If you use the Alternate-Run mode to keep the processor running, the time display is samples of the program execution. In normal Run mode, the processor halts after each sample, then starts from the halt location for the new sample period.

Parameters

QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)]

1) OFF (0) essentially turns the command off.
2) ALL (1) enables any data- or program-memory or I/O access or internal cycle for analysis.
3) P (2) enables any program-memory access. PR enables any program-memory read, PW enables any program-memory write, PIAQ enables any program-memory instruction-acquisition cycle.
4) D (3) enables any data-memory access. DR enables any data-memory read, DW enables any data-memory write.
5) IO (4) enables any input/output access. IOR enables any input read, IOW enables any output write.
6) I (5) enables any internal access. IA enables internal TMS320C2x cycles. IIAQ enables any internal instruction acquisition.
7) R enables any kind of read operation.
8) W enables any kind of write operation.
9) IAQ enables any kind of instruction acquisition.

START ADDRESS, END ADDRESS, and ADDRESS INCREMENT (0=ONLY 1 SAMPLE): These addresses define the inclusive address range to qualify an event for time-sample storage. The command shows memory-area use for each address range from the start address to the end address (incremented with address increment). For the default of 0, the command displays one line consisting of time for the entire range.

For an entry of 1, the command displays a line for the first and second addresses in the range, then displays a line for the third and fourth addresses, then for the fifth and sixth, etc.

For an entry of 2, the command displays a line for the first three addresses in the selected range, then a line for the next three, then a line for the next three, etc.

This pattern continues up to an entry number that matches the address size. For any larger entry, the display consists of a single line covering from the first address in the range up to the entry number. For example, with a range from 0 to FF, an entry of 3FF gives a single line for time from address 0 to address 3FF.

The SAMPLE TIME parameter tells the command how long to stay in the Run mode for each of the steps defined above. The larger the number, the longer each step runs. The exact time depends on the emulation clock frequency, the address increment, and the program, so it is not necessarily true that a setting of, for example, 500, runs half as long as a setting of 1000.

Run time may or may not have an effect on the program-execution time displayed, depending on that time, the address step size, and the program itself. For this

reason, you need to coordinate your entries with your program; for example, entering an address increment of 0 and a sample time of 050 for a very long program would not give a meaningful display.

Texas Instruments recommends pressing RESET on the XDS Operator Panel for quick action if you want to end command execution early. Pressing ESC works, but the command does not end until after the current address increment runs.

Associated Commands DTIME (Display Time), Page 4-45.

Example 1 Multiple address ranges

```
DISPLAY                                                          ENTER
?                                                                XTIME<CR>
XTIME

 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF    <CR>
 START ADDRESS                                       = 0000   F000<CR>
 END ADDRESS                                         = 0000   F013<CR>
 ADDRESS INCREMENT (0=ONLY 1 SAMPLE)                 = 0000   1<CR>
 SAMPLE TIME (0=NORM--7FFF)                          = 0000   100<CR>

 ADDR RANGE  AVG: MI SEC MS  US  NS  PERCENT
 F000--F001   ... .. .. ... ... ...   00.0%   .
 F002--F003   ... .. .. ... ... ...   00.0%   .
 F004--F005   ... .. .. ... 301 750   96.8%   ******************
 F006--F007   ... .. .. ... ... ...   00.0%   .
 F008--F009   ... .. .. ... ..2 500   00.8%   .
 F00A--F00B   ... .. .. ... ... ...   00.0%   .
 F00C--F00D   ... .. .. ... ..2 500   00.8%   .
 F00E--F00F   ... .. .. ... ..2 500   00.8%   .
 F010--F011   ... .. .. ... ... ...   00.0%   .
 F012--F013   ... .. .. ... ..2 250   00.7%   .
                                        TOTAL = 99.9%
```

Example 2 Single address range

```
DISPLAY                                                          ENTER

?                                                                XTIME<CR>
XTIME
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF    <CR>
 START ADDRESS                                       = 0000   F000<CR>
 END ADDRESS                                         = 0000   F013<CR>
 ADDRESS INCREMENT (0=ONLY 1 SAMPLE)                 = 0000   <CR>
 SAMPLE TIME (0=NORM--7FFF)                          = 0000   10<CR>

   ADDR RANGE  AVG: MI SEC MS  US  NS  PERCENT
   F000--F013   ... .. .. 526 006 000  100.0%   ******************
                              TOTAL =100.0%
```

4.4 Register Commands

A TMS320C2x XDS/22 lets you display or change the content of the following TMS320C2x-device registers:

- Accumulator (ACC)
- Auxiliary registers (AR_x)
- Auxiliary-register pointer (ARP)
- Auxiliary-register pointer buffer (ARB)
- Global register (GREG)
- Product register (P)
- Program counter (PC)
- Stack (S_x, x = 0 - n, S_n reserved for emulator)
- Status registers (STx)
- Temporary register (T)

You can also display or change the content of the following status-register bits:

ST0

- DP, Data-memory page pointer, bits 0-8
- INTM, Interrupt Mode, bit 9
- OVM, Overflow Mode, bit 11
- OV, Overflow Flag, bit 12
- ARP, Auxiliary-register pointer, bits 13-15.

Bit 10 is always a 1.

ST1

- PM, Product-shift mode, bits 0 and 1
- TXM, Transmit mode, bit 2
- FO, Format mode, bit 3
- XF, XF pin status, bit 4
- FSM, Frame-synchronization mode, bit 5
- HM, Hold mode, bit 6
- C, Carry bit, bit 9
- SXM, Sign-extension mode, bit 10
- TC, Test/Control flag, bit 11
- CNF, onchip block B_0 control, bit 12
- ARB, Auxiliary-register pointer buffer, bits 13-15.

Bits 7 and 8 are always 1's.

> **Note:**
>
> For emulation, both registers and register content are called variables.

To change a variable, type its mnemonic, an equal sign (=), then the new value: in decimal or hexadecimal depending upon the current numeric mode, then press RETURN. The monitor program redisplays the ? prompt, but does not display the changed value. In decimal mode, you can enter the new value in decimal, but the monitor program converts it to hexadecimal. Subsequent display is also in hexadecimal (with the selected notation) unless you precede the variable name with DEC or HEX.

To display a variable value in hexadecimal, simply type its mnemonic, then press RETURN. The screen displays the mnemonic name, an equal sign, and the current value, then redisplays the ? prompt.

To display a variable value in decimal (in the decimal mode), type DEC or MAG, then the variable name, then press RETURN. The screen displays the mnemonic name,

an equal sign, and the current value in the selected numeric mode, then redisplays the ? prompt.

You can also display the content of the program counter, status registers, product register, temporary register, and accumulator with the DR (Display Register) command, and both display and change the same registers with the IR (Inspect Register) and MR commands. Each of these commands displays the status registers as four hexadecimal digits in the hexadecimal mode, or as the necessary number of decimal or magnitude digits in the decimal mode (with command name preceded by DEC or MAG as described above).

For a display of the individual status-register items *and* each register as a whole, use the DPS (Display Processor Status).

For example:

```
DISPLAY                          ENTER
?                                PC<CR>
 PC=0000
?                                PC=200
?                                PC<CR>
 PC=200
?
```

```
DISPLAY                          ENTER
?                                DR<CR>
DR
 PC =0000   T  =0000
 ST0=276C   P  =002C44A8
 ST1=03F0   ACC=00000001

?
```

```
DISPLAY                                                    ENTER
?                                                          DPS<CR>
DPS
 PC =0000 T  =30D9       AR0=0001  AR4=40E6   S0=0000
 ST0=276C P  =002C44A8   AR1=C9EE             S1=0000
 ST1=2BF1 ACC=00000001   AR2=C4AA             S2=0000
                         AR3=366F

ARP OV OVM INTM DP         ARB CNF TC SXM XF FO TXM PM
 1  0   0   1   16C         1  0   1   0  1  0   0  1

?
```

5. Operation - Emulation

This section describes XDS/22 operation from command entry and editing through target-system connection to running and ending an emulation session.

Section 6 describes operation of the XDS/22 breakpoint and tracing features if your program requires debugging. Section 8 describes multiprocessing operation.

Topics include:

5.1 Command Entry

This section describes advanced command entry and numeric display.

Topics include:

5.1.1 General

The commands you have been entering from your terminal's keyboard have been going into a buffer, called the current buffer, on the Emulator card. Using a buffer lets you enter more than one command at a time (called a string, Page 5-6) and also lets you repeat the last command or commands (Page 5-4).

The monitor program reads the current buffer and simultaneously echoes it (sends it back) to the screen for display. Letter display is always upper case (capital) letters, regardless of what you type.

The Emulator card also has a second buffer. Using two buffers lets you execute commands in sequence (Page 5-9).

5.1.2 Simple Command Entry

You have been using the simplest form of XDS command entry, the Prompt mode, all along. You typed a command name, then pressed RETURN, and the XDS monitor program prompted you for each parameter - memory addresses, data values, etc. - for which it needed input. The prompt always included the current parameter value and all the acceptable values for that parameter. You then accepted that current value by pressing RETURN or changed it by typing another permissable value, then pressing RETURN.

After you accepted or changed the last command parameter, the command executed immediately.

You have probably also noticed that some parameters with only a few possible values displayed those values as a list in square brackets; and you could enter an item by typing

- Its list number (starting from 0),
- The corresponding text entry
- Just the first letter of that text entry.

You may not have noticed that some longer prompts omit the list number; however, you can use it, whether or not it displays.

For example,

```
ADDR TYPE [0=INCL, EXCL, 2=SNGL] = INCL
```

from the BTT command (Page 4-16) lets you set the address type for the addresses entered in the two preceding prompts (not shown here) as one of Inclusive range, Exclusive range (all addresses except those between the two addresses), a single point, or two single points. To keep the default, INCL, just press RETURN. To select an exclusive range, type 1 or "E" or "EXCL", then press RETURN. To select a single address as a single point or two addresses as single points, type 2 or "S" or "SNGL", then press RETURN.

You do have to type in addresses, data values, etc. when required.

After command execution, with the question-mark prompt displayed, you can redisplay the current buffer for editing.

You can edit parameter-value entries before or after command execution:

Before Move the cursor to the line containing the error with the key or keys that move the cursor upward, then move the cursor to the error with the key(s) that move the cursor left and right, then move the cursor back to where you were with the key(s) that move the cursor downward.

After Type the command name, then press RETURN, to bring it up in the Prompt mode and correct the parameter when you get to it.

Or refer to Section 5.2.

5.1.3 Other Command-Entry Methods

5.1.3.1 Command Repetition

The TMS320C2x Emulator has several ways to repeat commands depending on what you want to repeat.

1) To repeat a command with current parameters, just press the Space bar once for each desired repetition.

2) To repeat a command in the Prompt mode as many times as desired, just type an asterisk (*) by pressing SHIFT and 8 together (on many terminals) after typing the command name and before pressing RETURN. To end command repetition, press ESC.

3) To repeat a command a selected number of times in the Prompt mode without having to count them, just type asterisk (*, press SHIFT and 8 together on many terminals), then a number from 1 to 9999, then press RETURN. The command repeats the entered number of times, then stops. You can also press ESC to stop it sooner.

5.1.3.2 Command Execution with Current Parameters

To execute a command with current parameters and see those parameters, type a period after the command name before pressing RETURN.

To execute a command with current parameters without seeing those parameters, type a semicolon after the command name before pressing RETURN.

To execute a command repeatedly with display of current parameters, type a period and an asterisk after the command name before pressing RETURN.

5.1.3.3 Changing Selected Parameters with a Single Entry

To change selected parameters with a single entry, follow the command name with a parameter list enclosed in parenthesis. Make the list as follows:

1) Follow each parameter value except the last with a space or comma

2) Replace each parameter value that you don't want to change with a dot (period)

3) Replace each parameter value that you do want to change with the new value.

This is called the Parenthesis mode. You really have to be careful with it, because the monitor program will execute *exactly* what you enter -- if it can. If not, the command reappears on the command line with the cursor at the first place the monitor program couldn't figure out what you wanted.

The list *must* include an entry for each command parameter, except that you can enter the letter "Q", then close parenthesis, after any changed parameter to tell the monitor program that you want to accept all remaining parameters at their current value.

You can repeat the command by entering an asterisk (or an asterisk followed by a number) *after* the closing parenthesis. Each repetition uses all the values in the parenthesis. You CANNOT include an asterisk inside the parentheses.

5.1.4 Checking Command Parameters

To see a command's current parameter values without changing them or executing the command, type an exclamation point (!) after the command name before pressing RETURN. This is called the Preview mode.

> **Note:**
>
> This mode does not work with the BTT, ICC, or IHC commands.

5.1.5 Multiple Command Entry

5.1.5.1 General

The monitor program lets you enter command strings – that is, several commands at a time. Such commands execute one at a time, in the exact order that you enter them; therefore, it is your responsibility to enter them in the correct order.

You can enter up to 79 characters in each string. That includes letters, numeral digits, spaces, separator characters, etc. Typing anything except numbers, letters, and the control characters discussed in this section causes an error message to display.

You can enter one command in the Prompt mode, another in the Parenthesis mode, etc.

Command entry is similar to what was previously described, except as follows.

5.1.5.2 Command-String Separators

You must separate each command name from the following command name with one of the following:

Separator	*Action*
Space	Command executes in Prompt mode
Comma	Command executes in Prompt mode
Right (closing) parenthesis	Command in parenthesis mode (Section 5.1.3.3) executes.
Period	Command executes with current parameters and displays them.
Semicolon	Command executes with current parameters, but does not display them.

5.1.5.3 Last Command

You can also include a control character after the last command in a string as follows:

Separator	*Action*
Period	Command executes with current parameters and displays them.
Semicolon	Command executes with current parameters, but does not display them.
Asterisk	Entire command string executes repeatedly until you press ESC.
Asterisk, number	Entire command string executes *number* times.

Note that after the last command is the only place that you can enter an asterisk or asterisk, number.

5.1.6 Ending a Command

The monitor program offers several ways to end a command. These work for the command-entry method just described as well as for the methods described later in this Section.

1) You can end a command at any time simply by pressing ESC at the terminal keyboard or pressing RESET on the XDS/22 Operator Panel.

   a) If you press ESC or RESET during parameter entry, the monitor program ignores any parameter changes that you may have made.

   b) If you press ESC or RESET after command execution, the monitor program ends the command, but keeps any parameter changes that you made during command entry.

2) You can also end most commands during parameter entry in the Prompt mode by simply typing Q any time that the monitor program displays a parameter value. The monitor program keeps any previous changes in that command; for example, if you change the first two parameters in a command, then type Q in response to the third parameter, the monitor program ends the command, but keeps your changes to the first two prompts.

3) You can only exit from the XA (Execute Assembler) command by entering an END directive or by pressing CNTL and E together.

5.1.7 Numeric Display and Notation

The XDS starts in an "assumed-hexadecimal" mode; that is, it assumes all input numbers to be hexadecimal (base 16), and displays all numbers as hexadecimal, but without any identifying symbol.

You can accept this mode or select numerical entry and display in decimal (base 10) notation. You can also display some values as hexadecimal numbers and other values as decimal numbers.

You control the basic XDS numerical input and display with the INIT command and further control display with the DEC, HEX, and MAG commands.

> **Note:**
>
> The monitor program displays most parameter-value ranges as hexadecimal numbers or as a list of decimal numbers enclosed in square brackets. If you select a hexadecimal display format, then the range values display in that format.

5.1.7.1 INIT Command

At system initialization, you can leave the XDS in its power-up hexadecimal numeric format or you can change it to decimal (but with four hexadecimal display-format options) by your response to the INIT-command hex-notation parameter:

```
HEX NOTATION [0=NONE, 1=TI, 2=X, 3=HS, 4=HP] = NONE
```

where:

NONE Default, all numerical input considered to be hexadecimal, but display accompanied with no symbols. For example, input of 40 means hexadecimal 40, and displays as 40.

TI All numerical input considered to be decimal, unless preceded by one of:

- >
- X, or OX
- H

followed by H, or preceded by O and followed by H.

Regardless of input, all hexadecimal numbers display preceded by >. For example, an entry of 40 displays as >28.

X Same as TI, except all hexadecimal numbers display preceded by X (except inside the XA, Execute Assembler, command)or OX (inside that command). For example, an entry of 40 displays as X28 or 0X28.

HS Same as TI, except all hexadecimal numbers display followed by H or preceded by O and followed by H as described for the X entry above. For example, an entry of 40 displays as 28H or 028H.

HP Same as TI, except all hexadecimal numbers display preceded by H or OH as described for the X entry above. For example, an entry of 40 displays as H28 or 0H28.

The XA (Execute Assembler) command requires the longer entry forms and generates the longer displays to eliminate any conflict between hexadecimal numbers and symbols.

1) Hexadecimal Operation

You can leave the monitor program in the hexadecimal mode by accepting the default of 0.[14] In the hexadecimal mode, the monitor program considers all numbers entered as hexadecimal numbers and displays all numbers as hexadecimal, but without any identifying symbol. It does *not* accept the four hexadecimal symbols described above.

You can enter decimal numbers within the XA (Execute Assembler) command.

[14] You can also return the XDS to the hexadecimal mode by re-executing the command and setting this parameter to 0.

2) Decimal Operation

If you select decimal operation (respond with any of 1-4 to the INIT-command prompt), the monitor program default action is to:

- Consider all input numbers preceded by + or - as signed decimal numbers
- Consider all input numbers not preceded by + or - as magnitude numbers (discussed in following text)
- Display all output numbers as hexadecimal numbers with the selected hexadecimal notation.

For signed numbers, the monitor program looks at the two's-complement binary value and supplies a + sign if its first bit (sign bit) is 0, or a - sign if its sign bit is 1. For magnitude numbers, the monitor program considers the sign bit as the most-significant binary digit, which means the magnitude number is the same as a decimal number for a positive number, but is much larger for a negative number. For example,

-1 decimal = 65535 magnitude = FFFF hexadecimal

With decimal notation selected, you, however, enter hexadecimal numbers by preceding each with one of >, X, or H, or by following it with H;[15] in other words, by using one of the hexadecimal notation signs. Bear in mind, however, that all hexadecimal numbers display in the format selected by the prompt, regardless of the entry format.

You can also switch any parameter value from decimal to hexadecimal to magnitude format by pressing CNTL and X together with that parameter value displayed. The monitor program remembers your selection and displays that value in that format until you change it or end the session.

> **Note:**
>
> You cannot change the parameter-value display format with CNTL/X from the power-up hexadecimal mode.

5.1.7.2 DEC Command

If you have selected decimal operation from the INIT command by selecting a value of 1, 2, 3, or 4 with the hex-notation prompt, you can switch a register value entered as hexadecimal to decimal by entering DEC *variable name*, then pressing RETURN. The monitor program continues to display that variable as a signed decimal number until you change it with a HEX or MAG command or end the session.

If you enter DEC before the name of a command that displays register content, but cannot change it, such as the DR command, then *all* that command display is decimal, regardless of what other commands you may have used. This change, however, goes away when you end the command.

---

[15] 0X (zero), 0H, or 0...H in assembly-language entry within the XA command and assembly-language displays in the DT, FT, IT, and XRA commands.

5.1.7.3 HEX and MAG Commands

The HEX and MAG commands work just like the DEC command except that the display is >*nnnn* for the HEX command or an unsigned decimal number (as previously discussed) for the MAG command.

5.1.8 Command Buffers and Sequenced Execution

You can display the previously-mentioned Emulator-card alternate buffer content at any time with the ? prompt displayed. At that time, the alternate and current buffers swap roles; that is, the alternate buffer becomes the current buffer and the current buffer becomes the alternate buffer.

You can only execute the current buffer.

For sequenced command entry, display the buffer that you want to execute first and press RETURN to execute it, then display the other buffer and press RETURN to execute it.

The key sequence to change buffers depends on your choice of numerical display mode.

- **Hexadecimal Mode**: Press + or -
- **Decimal Mode**: Press whatever keys you selected to move the cursor up and down with the ICC command (Page 4-67 or 5-19), CTRL and J (up) or CTRL and K (down) default.

You cannot change the hexadecimal-mode assignments.

5.1.9 Register Commands

Register commands work just like regular Emulator commands except that you precede a value change with an equal sign.

5.1.10 Other Emulator Displays

During execution of commands such as the FILL command, or during program operations, the screen cursor may move to the right and remain there until the operation is complete.

The output display of commands that can cause more than one screen of data to be displayed contains a [] prompt. Press RETURN to see further data display.

Upon reaching the final screen, the [] prompt remains, but the screen does not advance. You can redisplay the previous screen by entering -<CR>. You can repeat this entry until you reach the very first screen. You can scroll forward by entering +<CR>. Refer to the IT (Inspect Trace) command description (Page 4-83) for further details.

Press ESC or the Emulator RESET button to end the command and the display.

5.1.11 Summary of Command Entry

The following tables summarize command entry. Remember that you can only enter letters, numerals, and the modifier, repeater, and terminator characters into a command.

Table 5-1. Single or Multiple Command Terminators

ENTRY	ACTION
<CR>	Executes command or command string.
<+> <->	Stores command or command string in alternate buffer.
<ESC>	Ends execution, clears current buffer.
<?>	After command execution, redisplays current buffer for editing.
<!>	Displays command parameters without executing command.

Table 5-2. Single or Multiple Command Repeaters

ENTRY	ACTION
<*>	Executes last command or entire string repeatedly in Prompt mode until ESC pressed.
<*nnnn>	Executes last command or entire string *nnnn* times (or until ESC pressed) in Prompt mode.
<.*>	Executes last command or entire string repeatedly with current parameters until ESC pressed.
<;*nnnn>	Executes last command or entire string *nnnn* times (or until ESC pressed) with no display of parameters.
<SP>	Executes last command or entire string one time for each depression using current parameters.

Table 5-3. Single Command Modifiers

ENTRY	ACTION
<.>	Command executes one time with current parameters and displays those parameters.
<;>	Command executes one time with current parameters but does not display those parameters.
<(p1, p2, ..., pn)>	Enters and executes command without parameter display. Enter new values or enter a dot (decimal point) to keep existing values or enter Q after a change to keep all remaining parameters with existing values. Enter space or comma between each parameter.
<(p1, p2, ..., pn)*>	Enters and executes command repeatedly without parameter display. Enter new values or enter a dot (decimal point) to keep existing values or enter Q after a change to keep all remaining parameters with existing values. Enter space or comma between each parameter. Press ESC to end.
<(p1, p2, ..., pn)*nnnn>	Enters and executes command *nnnn* times without parameter display. Enter new values or enter dot (decimal point) to keep existing values or enter Q after a change to keep all remaining parameters with existing values. Enter space or comma between each parameter.

Table 5-4. Multiple Command Separators

ENTRY	ACTION
<,>	Separates commands.
<b> <.> <;> <!>	Blank space, period, semicolon, or exclamation point also separates commands.
<()>	For command entries in Parenthesis mode, right or left parenthesis also separates commands.

Table 5-5. Monitor Program Key Functions

FUNCTION DESCRIPTION	HEX MODE	DEC MODE
Move cursor right Move cursor left	> <	<CTRL/L> <CTRL/H>
Switch command-entry buffers	+ -	<CTRL/J> <CTRL/K>
Prompt input mode Next prompt Previous prompt	 + -	 <CTRL/J> <CTRL/K>
Change value to other mode	†	<CTRL/X>

†Cannot change values in Hexadecimal mode

5.1.12 Masking

Several TMS320C2x Emulator commands let you enter addresses or data and the monitor program performs certain actions when the command reaches one of those addresses or generates that data.

The TMS320C2x Emulator also offers a refinement of this address or data matching called masking. Masking lets you define "don't-care" states for various parts of the comparison value you entered, which means that a single value really becomes a range of addresses or values, since the program considers any of the modified values as matches.

A mask, and its associated data, always works as a binary number, whether entered in decimal or hexadecimal notation; however, you will probably be able to understand masking more easily if you stick to hexadecimal.

An entry of 0 for any mask bit tells the program that either a 1 or 0 in the corresponding bit position of the incoming data (which, of course, you must also think of as binary) "matches" the value you entered. In other words, it tells the program to ignore that bit.

On the other hand, an entry of 1 says that the incoming data bit "matches" *if and only if* it is the exact bit you entered.

For an obvious, but not trivial, example, consider mask values of >0[16] (0000) and >F (1111). With a mask of >0, *any* data or address value byte matches; with a mask of >F, *only* the exact data or address value does.

For another example, consider a mask of >3, which is 0011. Since the first two mask bits are 0's, any of 00, 01, 10, or 11 in the first two bits of the incoming data digit would qualify; but, since the last two mask bits are 1's, only the specific values in those bit positions of the byte to be masked qualify. For example, look at a mask of >3 working against an entered value of >F (1111). The program would consider any of the following incoming values as a match: >3 (0011), >8 (0111) >B (1011), or >F (1111).

---

[16] The > symbol indicates hexadecimal notation throughout this manual.

Masking works exactly the same for the remaining digits. Consider a mask of >3A (00111010) working against an entered value of >FF (11111111). The program would consider any of the following incoming values as a match: >3A (00111010) to >3F (00111111), >8A (01111010) to >8F (01111111), >BA (10111010) to >BF (10111111), or >FA (11111010) to >FF (11111111).

Again, masking is independent of the system numeric mode, but it is probably easier to keep up with in hexadecimal notation.

5.2 Command Editing

Command editing usually requires cursor movement to the point of error. You can edit command names and parameter values before or after entry.

5.2.1 Cursor Movement

If you have used the ICC (Initialize Cursor Control) command on Page 4-67, move the cursor left or right with the terminal arrow keys or with whatever control sequence that you entered at that time. On most terminals, you can also use Backspace. Arrow-key movement usually doesn't erase characters; backspacing usually does.

To move the cursor up or down for parameter-value editing, use the Up and Down arrow keys or whatever you selected to move the cursor up or down. If you move up, that is, to a previous parameter, the monitor program redisplays the prompt line with the value you entered or accepted, but with a minus sign in front of the line.

If you didn't use the ICC command, or if you cancel it with the RCC (Restore Cursor Control), use + to move up and - to move down (hexadecimal mode) or use CNTL/K to move up and CNTL/J to move down (decimal mode).

5.2.2 Before Entry

5.2.2.1 Command Names

1) If you mistyped a command name, or any syntax, the monitor program redisplays the command line with the cursor at the incorrect character when you press RETURN to execute the command. Retype and enter again.

2) If you type the wrong command name, move the cursor to the mistake with the left arrow, BACK SPACE, or right arrow key and retype it. Or press <CNTL/D> to delete the character at the cursor position and press <CNTL/I> to insert a blank space for a new character.

3) Or, type ? to redisplay the command line including any deleted character(s).

4) You can also press + or - (hexadecimal mode) or CNTL/J or CNTL/K (decimal mode) with the ? prompt displayed to call the alternate command buffer. The monitor program moves the other buffer to be the current buffer and positions the cursor at the first character position in that buffer, ready for editing.

   - If the buffer is empty, type the command correctly and execute by pressing RETURN.
   - If the buffer isn't empty, and you don't want to keep its content, just type over it, then execute by pressing RETURN. Don't forget to erase any extra characters by pressing <CNTL/D> or <SP> as many times as needed.
   - If the buffer isn't empty, and you do want to keep it, press + or - or CNTL/J or CNTL/K again with the ? prompt displayed to return the original buffer with the original error, then move the cursor to that point and correct it. Now, execute by pressing RETURN.

5.2.2.2 Parameter Values

If you notice an incorrect parameter value before pressing RETURN, you have two editing choices.

If you notice it before leaving the line, move the cursor left or right to the point of error and retype. If you notice it in a preceding line, press - (hexadecimal mode) or CNTL/K (decimal mode) as many times as necessary to return to that line, then move the cursor right or left and correct the error.

After correction, press + (hexadecimal mode) or CNTL/J (decimal mode) as many times as necessary to return to where you were working.

5.2.3 After Entry

If you entered the wrong command, press ESC to stop it (or any key to stop if the screen displays RUNNING), then enter the correct command.

If you entered the wrong parameters, enter the command name to bring up the parameters in the Prompt mode, then move the cursor to the incorrect value by pressing RETURN to accept the correct values, then retyping the incorrect value; or by pressing + as many times as necessary to move to the incorrect value.

5.2.4 Other XDS Functions Equivalent to Editing

You can use the Parenthesis mode (Section 5.1.3.3) with dots for the correct values, the correct value to replace the incorrect value, then Q to use all remaining values.

You can also change memory content with the

- IDM (Inspect Data Memory) command, Page 4-69
- IPM (Inspect Program Memory) command, Page 4-78.
- MDM (Modify Data Memory) command, Page 4-91.
- MPM (Modify Program Memory) command, Page 4-93.

5.3 The XDS System Monitor Program

The TMS320C2x Emulator monitor program lets you control TMS320C2x-device emulation with the commands described in Section 5.1. The monitor program also controls cursor position and screen display and detects errors as described in this Section.

Topics in this section include:

5.3.1 Commands

Table 5-6 shows the TMS320C2x Emulator commands in functional groups. Section 5.1 describes command entry in detail.

Table 5-6. Emulator Commands Grouped by Function

Initialization	
DEC	Decimal Format
HEX	Hexadecimal Format
ICC	Initialize Cursor Control
IHC	Initialize Host Control
IMD	Initialize Multiprocessing
INIT	Initialize Emulation
IPORT	Initialize Parameters
IPRM	Initialize Parameters
ITR	Initialize Target RAM
LOAD	Load Stored Parameters
MAG	Magnitude Format
MAP	Map Expansion Memory
RCC	Restore Cursor Controls
RESTART	Restart
SNAP	Freeze Display
Memory	
DDM	Display Data Memory
DPM	Display Program Memory
FILL	Fill Memory with Data
FIND	Find Data in Memory
IDM	Inspect Data Memory
IPM	Inspect Program Memory
MDM	Modify Data Memory
MPM	Modify Program Memory
Communications	
DIO	Display I/O
MIO	Modify I/O
Offload	
DL	Download
UL	Upload
Run	
CRUN	Continue Run
GHALT	Group Halt (MP Mode)
GRUN	Group Run (MP Mode)
RTR	Run on Target Reset
RUN	Continuous Execution
SS	Single-Step Execution
STOP	Stop Alternate-Run Mode
THALT	Total Halt (MP Mode)
TRUN	Total Run (MP Mode)

Mode	
ARM	Initiate Alternate-Run Mode
BGND	Initiate Background Mode
DISARM	End Alternate-Run Mode
HOST	Initiate Host Mode
IMP	Initiate Multiprocessing Mode
#n	Select Emulator #n
Hardware Breakpoint/Trace	
BTT	Set BTT Parameters
DBTT	Show BTT Parameters
DT	Display Trace
DTIME	Show Time Settings
FT	Find Trace
IBTT	Initialize BTT
IT	Inspect Trace
XTIME	Time Analysis
Software Breakpoint	
CASB	Clear Software Breakpoints
CSB	Clear one Software Breakpoint
DSB	Display all Breakpoints
SSB	Set one Breakpoint
Assembler	
XA	Execute Assembler
XRA	Reverse Assembler
Status	
DES	Display Emulator Status
DHS	Display Halt Status
DMAP	Display Memory Map
DPS	Display Processor Status
DTS	Display Trace Status
/n	Display Emulator MP Status
Miscellaneous	
DV	Display Values
HELP	Display Command Menu
ID	Display Firmware Revision
LOG	Print Commands and Response
SAVE	Save Parameters
Registers	
DR	Display Registers
IR	Inspect Registers
MR	Modify Registers

5.3.2 Cursor Controls

The XDS system monitor program can define cursor controls for both its controlling terminal and a host system by means of the ICC (Initialize Cursor Control) and RCC (Reset Cursor Control) commands.

5.3.2.1 The ICC Command

The ICC command lets you define cursor-control keys for general use and for particular use with the SNAP command (Page 4-100) for a fixed-format display with changing values.

If your terminal has arrow keys, define those as the control keys. If your terminal uses or allows Control sequences for cursor movement, you may want to use those.

> **Note:**
>
> Remember, after execution of the ICC command, *only* the keys selected by that command execution can move the cursor.
>
> Pressing a default cursor-control key causes display of that key on the screen, but no cursor movement. You do not get any other message.

If you use CNTL/D and CNTL/I for cursor-up or cursor-down movement, you can't then use the corresponding delete and insert functions in the Prompt mode; however, you can use them for any other kind of command entry. On the other hand, if you use CNTL/D or CNTL/I for cursor-left or cursor-right movement, then you can't use the delete or insert function in any entry mode.

An XDS system accepts only single ASCII characters as cursor-control characters; therefore, the multiple-character sequences generated by some terminals will not work. In that case, don't execute the ICC command, but use the XDS power-up values.

Also, don't use the ICC command if the terminal doesn't have cursor-control functions.

5.3.2.2 The RCC Command

The RCC command restores cursor control to default values summarized in the following table.

Table 5-7. Cursor Control and Insert/Delete Function Keys

KEY	ACTION
<	Moves cursor one position left without erasing characters. (CTRL/H is standard backspace character on all terminals).
>	Moves cursor one position right without erasing characters.
-	Moves cursor up one line in hexadecimal mode. Terminal displays previous line marked with - sign to indicate reprint. Used only in Prompt Mode.
+	Moves cursor down one line in hexadecimal mode. Used only in Prompt Mode.
^K	Moves cursor up one line in decimal mode. Terminal displays previous line marked with - sign to indicate reprint. Used only in Prompt Mode.
^J	Moves cursor down one line in decimal mode. Used only in Prompt Mode.
CNTL/D	Deletes character under cursor.
CNTL/I	Inserts one space before cursor position and moves text to the right.

5.3.3 Display Control

You can control the output display in several ways.

1) **Pause control**: Display of command output scrolls towards the top of the screen. To stop scrolling at any point, press any key except ESC. To restart scrolling, press any key except ESC.

   As usual, pressing ESC aborts command execution, ends the display, and returns the monitor to the Prompt mode.

2) **Fixing the Display**: You can also end scrolling by entering the SNAP command, then the command or commands that you wish to have a fixed display. During execution of the last command, the final 22 output lines become the fixed display. Any preceding lines scroll off the screen and do NOT reappear.

   Now, if you repeat the command string, for example by pressing <SP> with the ? displayed, the screen shows only the last 22 lines, with any changes that may result from execution.

To use the SNAP command, the terminal must have cursor control functions and you must have executed the ICC (Initialize Cursor Control) command.

Since the SNAP command does not clear the screen, execution may cause a confusing display if the procedure includes commands that use interactive parameter-definition (refer to Command Entry on Page 5-2) or commands that require user data entry.

5.3.4 Errors

5.3.4.1 Syntax Errors

The monitor program scans the command line for syntax errors before executing any command(s). If it finds an error, it displays an error message, enters the edit mode, and displays the command line with the cursor positioned over the first wrong character.

Any of the following is a syntax error:

- Misspelled or invalid command name
- Commands improperly separated
- Invalid command modifiers
- SNAP command as other than first in command line
- Misplaced asterisk (*) modifier

5.3.4.2 Parameter-Definition Errors

If you attempt to enter a parameter value outside the range of values defined for that parameter, the monitor program displays an error message and enters the edit mode.

5.3.4.3 Invalid Control Characters

In the prompt mode (? displayed), the monitor program generally ignores any characters other than valid modifiers.

5.3.4.4 Invalid Commands

The monitor program rejects invalid commands, for example, a multiprocessing command with the XDS system not operating in the multiprocessing mode.

> **Note:**
>
> Be particularly careful with command entry in the parenthesis mode. It's very easy to enter parameters in the wrong order, with wrong values, with wrong separators or modifiers, or with the wrong number of values. Don't forget the period to keep an existing value.

5.3.4.5 Register Command Errors

Be careful to use indexes only where applicable.

5.3.4.6 Miscellaneous Error Conditions

Be careful of the following:

Hardware breakpoint errors -- Breakpoint/Trace/Timing card not installed

Multiprocessing errors -- Units not properly linked, command errors

Hardware error -- Memory parity, using wrong clock, no clock.

5.4 Memory Considerations

An XDS/22 configured for TMS320C2x emulation lets you substitute the 64K of random-access memory on the Memory Expansion/Communications card for any data or program memory in your target system, except what would be physically located inside the TMS320C2x. This feature lets you design your system before all the parts have been purchased or lets you design your program in RAM, then use a PROM Programmer connected to the XDS to immediately transfer it to firmware.

The Emulator card also has up to 8K of single-cycle static RAM for evaluation, stand-alone operation, or speed-critical software operation. You can also substitute for this memory.

This Section describes monitor-program commands that control substitution. Refer to Section 3.1 for a description of the commands to display, inspect, or modify TMS320C2x and Memory Expansion/Communications-card memory.

> **Note:**
>
> Some of the XDS command prompts let you use memory that doesn't exist or isn't available in actual TMS320C2x device operation, as described in the relevant user's guides. Texas Instruments suggests that you not use those locations, or use them only during emulation to comment your program, then erase them before firmware preparation.

> **Note:**
>
> A TMS320C2x device powers up in the memory configuration described in the TMS32020 or TMS320C25 User's Guide as the configuration after a CFND instruction. To change it, enter a CFNP instruction with the XA command (Page 4-110).

5.4.1 Memory-Substitution Commands

Substitution means that the XDS monitor program checks each memory access from your program; and, if that address is in the substitution range, changes it to access the memory on the Memory Expansion/Communications card. You won't notice this substitution during execution of such XDS commands as FILL, IDM, or IPM; but, it does take a finite amount of time, which means you may notice it during execution of your application program.

Also, for example, patching a program in substitution memory won't change information in target-system memory unless the program changes the substitution memory.

This flexibility can sometimes cause synchronization problems, therefore, the INIT command lets you specify waiting for the target system's READY signal or not waiting for that signal before ending memory access.

5.4.1.1 The MAP Command

The MAP command lets you do the actual substitution. You specify

1) Data or program memory
2) Address at which you want substitution to start
3) Hexadecimal number of 1K-word blocks you want to substitute
4) What you want the substituted memory to be
5) How many additional wait states you want the substituted memory to have.

Item 1 is obvious. The address (item 2) must be a number evenly divisible by >400 (decimal 1024); otherwise, the command rounds it down to the first number that is. The number of blocks (item 3) must be entered in hexadecimal.

Item 4 lets you specify how the block is substituted. Unmapping removes the specified block from the memory map, but does not change its content. ROM prevents write-access to the block during program execution; except by means of the FILL, IDM, or IPM commands. It does not change the block content. RAM allows write-access to the block, but does not change the block content. Copy writes the content of target-system memory into the block, then designates it as ROM.

The last item allows use of a variety of memory devices in the target-system memory. The dynamic RAM in the expansion memory runs with one extra wait state (two cycles) if the system clock is less than 5 MHz, or with three extra wait states (four cycles) if the clock is greater than 5 MHz. If the devices on the card require more, enter that number here.

5.4.1.2 The DMAP Command

The DMAP command (Page 4-34) lets you see the memory map to determine what is substituted where. It also shows the onboard target RAM if that has been turned on, but not substituted for.

5.4.2 Onboard Target RAM

The TMS320C2x Emulator card comes with 8K words of onboard RAM that you can use for evaluation, stand-alone operation, or speed-critical software development. Memory location is >0 - >FFFF in the target-system program memory and >0400 through >13FF in the data memory.

You control this memory with separate data- and program-memory prompts in the ITR (Inspect Target RAM) command.

> **Caution:**
>
> **If enabled onboard RAM and external target memory have the same address, bus conflict can damage the emulator and/or target memory.**

Refer to Section 9.4 on Page 9-27 for further details.

5.5 TMS320C2x Emulator-Card Assembler

The TMS320C2x Emulator-card assembler supports the following features:

- Absolute addressing
- Line-by-line assembly
- Symbolic addressing
- Limited forward referencing
- Free-form buffered input
- Plus and minus (+/-) expression operators with hexadecimal and decimal data, text, and symbols
- Alphabetic symbol-table dump
- Responds to most Assembler Directives (Section 5.5.7, Page 5-31).

Topics in this section include:

5.5.1 Comparison of TMS320C2x Assembly Language and TMS320C2x Emulator Assembly Language

1) The TMS320C2x Emulator Assembler does not process multiplication or division operators in expressions.

2) Operands must be numerical (hexadecimal or decimal), text, or symbolic. The assembler evaluates from left to right with no parentheses allowed.

3) The assembler considers numbers to be decimal unless preceded by any of:

   - >
   - 0X
   - 0H

   or preceded by 0 and followed by H.

4) You can use symbol and label names that start with H or X.

5) The assembler considers numbers to be positive unless preceded by a - sign.

6) If an operation generates more than 16 digits, you get an overflow message. The monitor program keeps only the 16 least-significant digits.

7) The assembler does not support binary data.

8) The only limit on expression length is the 60-character buffer.

9) The only expression terminators are End-of-Line, blank, or comma.

10) Text must begin and end with an apostrophe. The assembler uses only the last two characters, and stores double apostrophes as a single apostrophe.

11) The only predefined symbol is $, which refers to the present location-counter value. Defined symbols can occur anywhere, undefined symbols (when allowed) must occur by themselves (no operands).

12) The TMS320C2x Emulator assembler does not process any of the following Directives:

CEND	DSEG
COPY	LOAD
CSEG	PEND
DEND	PSEG
DORG	SREF

13) The TMS320C2x Emulator assembler ignores the following directives if entered from the terminal keyboard, but honors them in a downloaded program:

IDT	TITL
LIST	UNL
RORG	

5.5.2 The XA (Execute Assembler) Command

5.5.2.1 General

The XA command lets you enter TMS320C2x assembly-language statements into the TMS320C2x Emulator-card assembler program; therefore, you *must* be familiar with TMS320C2x Assembly Language as detailed in the various device User's Guides.

You can also download source code from a host computer system by means of the DL command (Page 4-32).

5.5.2.2 Syntax

The XA command syntax is:

```
DISPLAY                                                    ENTER

?                                                          XA<CR>
XA
 START ADDRESS                              = 0000         <CR>
 USE OLD SYMBOL TABLE (0=NO, 1=YES) = NO                   <CR>
```

START ADDRESS: Sets location in program memory to contain first assembled command. Be sure the address is actually available and configured as RAM.

USE OLD SYMBOL TABLE: Response of 0=NO clears any existing values, which means you must redefine all symbols. Response of 1=YES keeps existing table values.

Any labels undefined after last assembler usage can be added to the old symbol table so that they can be defined.

5.5.3 TMS320C2x Assembly Language

5.5.3.1 Source-Code Format

After your response to the symbol-table prompt, the XA command displays the first program line, with a line number, the address you entered above, and a blank data field that will hold the object code resulting from assembler action on your source-code entries.

Before this line appears, you can end the XA command by pressing ESC as usual. After this line appears, you must enter an END directive or press CNTL and E together.

```
LINE   ADD    DATA   LABEL  MNEM      OPERANDS  COMMENT
0001   0000          []
```

Figure 5-1. TMS320C2x Assembler Entry Format

You enter statements in the source-code format suggested by the headings. They work just as described in the above-mentioned manual except:

Label Unchanged. If you don't use a label, press the Space bar several times or press CTRL and T together to move the cursor under the MNEM

heading. You *can* get by with just pressing the Space bar one time, but your display will be easier to read if you align it with the headings.

Command Unchanged. Remember to move the cursor to the next heading after the mnemonic or directive.

Operand

1) The operand field does not accept binary integer constants, but considers any integer to be decimal unless preceded by ">", "0X", "0H", or preceded by "0" and followed by "H".

2) Expressions must be 60 characters or less and the only permitted arithmetical operators are + and -.

3) Text strings used as expression operands must start and end with single quote. The assembler uses only the last two characters and ignores control characters.

4) The assembler considers only <CR>, comma, or space as the end of an expression.

Comments

You can include comments after any mnemonic that has operands. Skip at least one space or move the cursor under the COMMENT heading. End the comment in column 60 (cursor will not move further).

Comments cause no output, are not stored, and do not appear in reverse-assembly.

You can also turn any program line into a comment by typing an asterisk in the first LABEL character position. Such a comment must also end in column 60. The line number increments, but the address doesn't.

Note:

You can ignore any spurious overflow warning resulting from byte operations with negative operands.

5.5.3.2 Source-Code Entry

After entry of an assembly-language statement, the screen redisplays the line with the hexadecimal code for the entered mnemonic under the data heading, and also displays the next line with a new decimal line number, the appropriate address, and the cursor in the first LABEL character position.

5.5.3.3 Ending Assembly

To end an assembly, enter an END directive (Section 5.5.7, Page 5-31) or press CNTL and E together <CNTL/E>. The former displays a list of errors, warnings, and unresolved labels, whether or not you actually had any (count of 0); the latter only displays those items if they occur.

5.5.4 Editing

You can edit source data before entering it. The following key combinations move the cursor:

CNTL/R Moves cursor right one character/space
CNTL/H Moves cursor left one character/space
CNTL/I Inserts one blank space
CNTL/D Deletes character under cursor
CNTL/T Moves cursor right to next tab stop. The first character position under each heading is a tab stop.

Pressing CNTL/T also moves the cursor as follows:

Label Skip If cursor at end of line less than seven characters long, it moves to the seventh column (beginning of the command field).

Word Skip Tabs to first character following next blank, or to end of line.

Wrap Around If cursor at end of line longer than six characters, it moves to the beginning of the current line.

5.5.5 Error Correction

If you discover an error before entering a statement, correct it as described in the preceding topic.

If you try to enter a statement containing a syntax error, the monitor program redisplays the line along with an error message (Section 5.5.10) after you press RETURN. You can move the cursor as described in the preceding topic, or press ESC to redisplay the index number and address with the cursor positioned in the label field.

If you discover an error after entry, go ahead and complete program entry, then correct the program-memory location containing the error with the IPM (Inspect Program Memory, Page 4-77) or MPM (Modify Program Memory, Page 4-93) command.

After correction, set the program counter to your program's starting address with the PC (Program Counter, Page 4-120) command, then enter the RUN command.

5.5.5.1 Example of Assembly-Language Entry

```
                              BEFORE DATA ENTRY
LINE    ADD     DATA    LABEL   MNEM        OPERANDS    COMMENT
0001    0000            []

                                DATA ENTRY
LINE    ADD     DATA    LABEL   MNEM        OPERANDS    COMMENT
0001    0000                    LRLK        AR0,1       Press RETURN


                              AFTER DATA ENTRY
LINE    ADD     DATA    LABEL   MNEM        OPERANDS    COMMENT
0001    0000    D000            LRLK        AR0,1       Press RETURN
0001    0001
0002    0002            []
```

Figure 5-2. XDS System TMS320C2x Assembler Data-Entry Example

This is a line from the program entered in Section 3.2.1 on Page 3-8. Because the source statement requires two bytes, line 0001 displays two addresses. All line numbers are in decimal, addresses in hexadecimal.

5.5.6 Downloading Source Code from a Host Computer

With ? displayed, type DL, then press RETURN. When the screen displays

```
DL
 LOAD OFFSET  =0000
```

type a new value, then press RETURN, or just press RETURN to accept the current value. In a similar fashion, change or accept the remaining parameters:

```
DESTINATION [0=PROGRAM, 1=PROM, 2=DATA, 3=ASM] = PROGRAM     2<CR>
PROTOCOL [0=NONE, 1=TEK, 2=ASR, 3=VAX]         = NONE        <CR>
SOURCE [0=HOST, 1=USER]                        = HOST<CR>
```

As the emulator receives downloaded data, it lists the program on the screen.

The emulator handles download errors by inserting four NOP (no operation) codes into memory for later patching by direct entry from the keyboard. The emulator also prints an error message with errors chained (each error refers to line number of previous error). The message includes labels on lines containing errors and includes the location-counter value for referencing.

To exit the assembler, enter an END Directive or CNTL/E.

5.5.7 Assembler Directives

The assembler can execute only the directives shown in Table 5-8 and described on the following pages; however, it accepts the DEF and REF directives.

Table 5-8. TMS320C2x Emulator Assembler Directives

DIRECTIVE	DEFINITION
AORG	Absolute origin
BES	Block ending with symbol
BSS	Block starting with symbol
DATA	Initialize word
END	End program
EQU	Define assembly-time constant
LIST	Restart source listing
OPTION	Define output format
PAGE	Eject page
TEXT	Initialize text editor
UNL	No source list

AORG Defines locations as absolute, starting at an address that you supply. If you include a label, it also has that address. If you enter a decimal number, it displays as a hexadecimal number. You get an error message if you try to start at greater than >FFFF (4095).

For example, HEX AORG >1000 sets label HEX to >1000 and starts defining locations from that address.

BES Forms block of words starting at location-counter value at execution time, then advances location counter by decimal, hexadecimal, or well-defined expression value in operand. Assigns label, if used, to next location after block.

For example, BUFF2 BES >20 saves a block of 32 bytes, then assigns label BUFF2 to the next location. If executed at >200, BUFF2 is >220.

BSS Forms block of words starting at location-counter value at execution time, then advances location counter by decimal, hexadecimal, or well-defined expression value in operand. Assigns label, if used, to location counter value at execution time.

For example, DATA1 BSS >15 reserves 21 bytes and assigns DATA1 to the location counter value at execution time. If executed at >150, DATA1 is >150 and the block ends at >165 which is now the location-counter value.

DATA Places decimal, hexadecimal, label, or well-defined operand value directly into adjacent memory locations as 16-bit two's-complement word, most-significant byte first, starting at location-counter value at execution time. Enter several values, up to 60 characters including comma separators, if desired.

A label, if used, takes the value of the first memory location.

For example, KONSI DATA 3200,1+'AB',>F4A0,'A' defines four words starting at the location-counter value, which becomes the value of KONSI, and stores those words as >0C80, >4143, >F4A0, and >0041.

END Ends assembly, returns control to monitor program, then prints all unresolved labels, warnings, and errors at the end of the listing. Must be last source statement in program. Anything following considered part of next assembly. <CNTL/E> ends assembly, but doesn't display a listing unles you actually had an error.

Caution: Do NOT use END-directive label field.

EQU Assigns value in operand to label content. Any symbols in operand must already be defined.

For example, SUM EQU AR1 assigns AR1 to SUM. After execution, you can use SUM and AR1 interchangeably.

LIST Overrides UNL (No Source Listing) directive to restart source listing for programs downloaded from external source only.

A label included in the directive takes the current value of the location counter.

OPTION Selects assembler output-listing format as one of the following:

DUNLST Each DATA directive listing limited to one line

FUNLST Turns all list options off.

NOLIST Turns listing output off (overrides LIST directive)

SYMLST Symbol table included in object file

TUNLST Each TEXT directive listing limited to one line

XREF Symbol cross-reference table included in listing.

A label included in the directive takes the current value of the location counter.

PAGE Starts new page in source-code listing with next source statement. A label included in the directive takes the current value of the location counter.

TEXT Specifies character string enclosed in single quotes in operand field. Maximum total length of statement label and operand including spaces and quotes is 52 characters.

TEXT directive places each eight-bit ASCII representation in memory as one byte of a word, filling the last word by placing >20 (blank) in the last byte if necessary.

If a - minus sign precedes the string, the assembler negates the last string character.

A label included in the directive takes the value where the assembler places the first word.

For example, MSG TEXT 'EXAMPLE' produces >4558, >414D, >504C, and >4520 and assigns >4558 as the location of MSG.

UNL Stops assembly listing if included in program downloaded from external source. Requires LIST directive in that program to restart output. Does not affect error and warning messages.

A label included in the directive takes the current value of the location counter.

5.5.8 Running the Assembled Program

If you believe the assembled program is correct, set the Program Counter to its starting address with the IR (Inspect Register, Page 4-82) or MR (Modify Register, Page 4-94) or PC (Program Counter, Page &ppc.) command, then enter the RUN command (Page 4-98).

If you believe the assembled program is not correct, refer to Section 5.5.4 on Page 5-29 for editing information, then run the corrected program as described above.

5.5.9 The Assembler as a Pseudo-Linker

You can use the assembler as a pseudo-linker for modular code development. In HOST Mode, the DL (Download) command LOAD-OFFSET parameter sets the initial location-counter value. You can stack consecutive downloads by maintaining the symbol table and location counter, or change the location counter by means of an AORG directive.

At the end of the Host mode, the assembler initializes the symbol table again, and the location counter assumes the LOAD OFFSET value of the next download.

Since the Assembler does not support the DEF and REF directives, overlapping symbols may cause problems. For a *simple* solution, make all labels *unique.*

5.5.10 Error and Warning Messages

Errors found during source-statement keyboard entry do not alter target memory, except for TEXT and DATA Directives. This lets you abort a patch without destroying memory.

Table 5-9 shows error messages that may appear during program entry.

Table 5-9. Assembler Warning Messages

CODE	ERROR	DEFINITION
0001	TRAILING CHARACTERS	
0002	OPERAND TOO LARGE	Longest operand used to assemble mnemonic.
0003	OVERFLOW	

Table 5-10. Assembler Error Messages

CODE	ERROR	DEFINITION
1501	INVALID TERMINATOR	Characters following label or before mnemonic are not spaces, or character after mnemonic is not space or carriage return.
1502	INVALID MNEMONIC	
1503	INDIRECT ADDRESSING REQUIRED	This mnemonic requires indirect addressing.
1504	"+" or "-" REQUIRED	"0" or "BR" addressing mode requires "+" or "-".
1505	INVALID EXPRESSION	
1506	INVALID REGISTER NUMBER	Number must be 0-4 for TMS32020 or 0-7 for TMS320C25
1507	ALPHA REQUIRED	
1508	"'" REQUIRED	
1509	DUPLICATE LABEL	Label already exists.
1510	LABEL TABLE FULL	No additional symbols can be accepted.
1511	OPERAND REQUIRED	Mnemonic requires operand.
1512	INVALID SHIFT COUNT	Shift count must be 0,1, or 4 for TMS32020, or 0-7 for TMS320C25.
1513	DEFINED EXPRESSION REQUIRED	Label must be previously defined
1514	OPERAND TOO LARGE	
1515	ASSEMBLING INTO ROM	

5.6 The XRA (Reverse-Assembler) Command

The reverse-assembler command produces source code from an assembled object file; however, it cannot recreate any of the original program labels or symbols.

Command Parameters

```
DISPLAY                                          ENTER

?                                                XRA<CR>
XRA
 START ADDRESS    = 0000                         F000<CR>
 LINES OF OUTPUT = 0000                          6<CR>
   0000  CB04 LDPK >004
   0001  D001 LALK >0100,0
   0003  6800 SACH >00,0
   0004  5900 TBLW >00
   0005  D102 ADLK >8000,1
   0007  FF80 B
?
```

START ADDRESS: Sets first memory address of reverse-assembled data.

LINES OF OUTPUT: Sets number of lines of source code displayed, six, in the above example.

5.7 Communication with External Devices

An XDS/22 can communicate with a variety of external devices such as:

- Terminal
- Logging Device
- PROM Programmer
- Host computer
- Modem
- Local-Area Network
- Another XDS unit.

to:

- Download data files from an external device to emulator memory
- Download data to a PROM programmer or logging device
- Transmit data from emulator memory to PROM programmer or logging device
- Upload data files from emulator to external device
- Emulate multiprocessor applications.

Topics include:

5.7.1 XDS/22 Communications Description

An XDS/22 communicates with peripheral devices by means of the Memory Expansion/Communications card (Section 9.2) and the four EIA RS-232C connectors located on the unit back panel. Throughout the remainder of this Manual, these will be referred to as Ports A, B, C, and D.

Figure 5-3 shows a typical XDS/22 installation with peripheral devices connected to the communication ports. For stand-alone operation, the host computer would not be connected.[17] Figure 8-1 on Page 8-2 shows a typical multiprocessing system with peripheral devices.

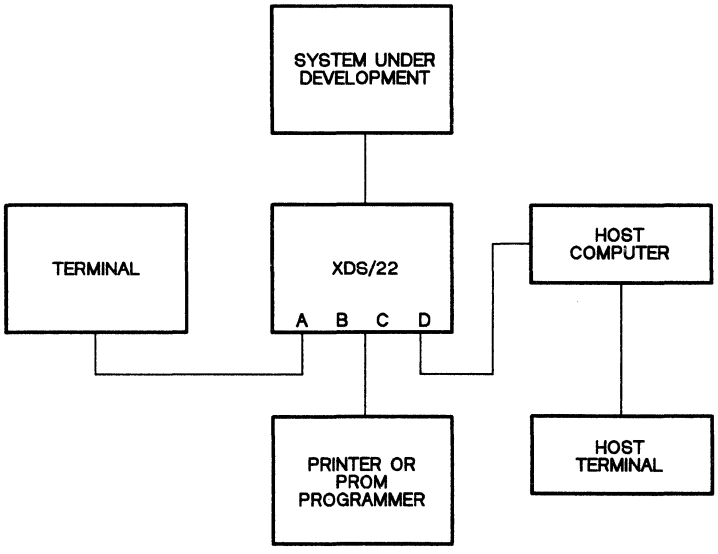


Figure 5-3. XDS/22 With Typical Peripheral Devices

Port A communicates with your terminal in single-emulator mode (as you have already seen in Section 2) or with the preceding XDS unit or your terminal in multiprocessor mode (Section 8).

Port B is reserved for XDS system expansion.

Port C communicates with an optional printer to log emulator operation, or with an optional PROM programmer to write your final program into a PROM.

Port D communicates with an optional host computer system for two-way file transfer which lets you use that computer for text editing, macroassembling, linking, and preliminary debugging. You then download the finished program into the XDS system for execution, final testing, and final debugging. The host system provides the communications hardware as well as the supporting software necessary for data transfer.

---

[17] In either case, the host terminal and associated cable are **not** part of the XDS system.

5.7.2 EIA Port Configuration

5.7.2.1 Port A

You configure Port A as to RS-232C signals by means of seven-position switches S3 and S4 on the Memory Expansion/Communications card (Section 9.2).

5.7.2.2 Port C

You select the RS-232C signals from Port C to a programmer or printer by cable wiring as shown in Section 9.2. The cable may be different for the two devices. After power-up, the Port C baud rate matches Port A; however, the baud rate can be changed with the IPORT command (Page 4-79).

The XDS monitor-program Autobaud feature sets baud rate (9600 preferred). The XDS monitor program also sets default parity and protocol - even parity, two stop bits, seven bits/character - for those terminals that can be programmed by the connected device. If your terminal cannot be remotely programmed, set it manually.

5.7.2.3 Port D

You configure Port D as to RS-232C signals by means of seven-position switches S1 and S2 on the Memory Expansion/Communications card (Section 9.2). You configure Port D as to baud rate, parity, number of stop bits, and number of bits/character by means of the IPORT (Initialize Port) command (Page 4-79) as shown in the following example which changes baud rate to 4800, parity to odd, and stop bits to 1:

```
DISPLAY                                                          ENTER
?                                                                IPORT
IPORT
 PORT [0=HOST("D"), 1=LOG/PROM("C")]                  = HOST     <CR>
 BAUD [19.2K, 9.6K, 4.8K, 1.2K, 600, 300, 110]       = 19.2     2<CR>
 PARITY [0=OFF, 1=ODD, 2=EVEN]                       = OFF      1<CR>
 STOP BITS [0=2, 1=1]                                = 2        1<CR>
 BITS/CHAR [0=7, 1=8]                                = 7        <CR>
```

5.7.3 Terminal Communications

5.7.3.1 Dumb Terminal

As described in Section 2.3, pressing RETURN twice at power-up or after pressing RESET operates the XDS/22 autobaud feature which sets baud rate; and the monitor program sets parity, number of stop bits, and bits/character if the latter items can be set with software. If not, set them manually before attempting to use the terminal.

Section 7.1 describes specific procedures for several specific terminals.

5.7.3.2 Intelligent Terminal or Personal Computer used as Terminal

You can use an intelligent terminal or personal computer as both a terminal and a host computer system. The autobaud sequence and monitor program defaults configure the device as a terminal. You must specify the USER value in the parameter selection for the DL and UL commands.

Section 2.4 describes a procedure for an IBM PC.[18] Section 7.2 describes a procedure for a Texas Instruments personal computer.

5.7.4 Logger/Programmer Communications

5.7.4.1 Logger

You control the logging device by means of the LOG command (Page 4-87). Power-up default for this command is OFF; therefore, to log all data during XDS/22 operation, execute this command and set the LOG DEVICE ON/OFF parameter to 1 (ON) before executing a RUN command.

If you want to stop logging at any time during the session, execute the LOG command and set the LOG DEVICE ON/OFF parameter to 0 (OFF).

Section 7.5 describes the procedure for connecting a specific printer.

5.7.4.2 Programmer

You control the Programmer by means of the DL (Download) or UL (Upload) command with the DESTINATION parameter set to 1 (PROM). Those commands (Pages 4-32 and 4-107) also let you set the starting point (usually 0, which is also the default) and protocol (check your programming device). For the DL command you also select the source: 0, default, for a Host computer; 1, User, if you are programming a ROM from an intelligent terminal or personal computer used as a terminal. For the UL command, you also select the object-code format.

The download data path if you select Host is from the Host into the XDS/22 through Port D, then out to the Programmer through Port C. The download data path if you select User is from the terminal or PC into the XDS/22 through Port A, then out to the Programmer through Port C. The upload data path is from XDS memory through XDS unit Port C.

> **Note:**
>
> If you select the NONE or TEK protocol, you must define certain control characters by means of the IHC (Initialize Host Controls) command on Page 4-70.

Section 7.4 describes the procedure for connecting and operating a specific programmer.

---

[18] IBM is a trademark of International Business Machines, Incorporated.

> **Note:**
>
> For uploading to a PROM programmer through Port C or to the user terminal through Port A, the UL command starts uploading when executed; and you do not need the HOST command.

5.7.5 Host Computer Communications

5.7.5.1 General

You can configure an XDS/22 for communications with a host computer as any type of I/O device as described in Section 7.6.

If treated as a terminal, the XDS/22 adjusts its baud rate to match that of the host.

5.7.5.2 Formats

An XDS/22 supports five data formats for upload, download, or both as follows:

1) Texas Instruments normal ASCII
2) Texas Instruments compressed
3) Intel Intellec 8/MDS
4) Motorola Exorcisor
5) Tektronix (TEK) hexadecimal

For details on these formats, refer to Section A.2.3.

5.7.5.3 Download Procedure with Host Computer Attached to XDS/22 Port D

1) Execute IPORT command (Page 4-79) to configure host port.
2) Execute IHC command (Page 4-70) to define control characters, if NONE or TEK download protocol selected.
3) Execute DL command (Page 4-32) to define download parameters.
4) Execute the HOST command (Page 4-62) to enter Terminal Mode and begin data transfer.
5) Log on to host system.
6) When prompted, send proper host command to start downloading specified file to Emulator.
7) Enter download-complete character from terminal keyboard, if host system does not automatically supply one.
8) When session complete, log-off of host system.
9) Press CNTL/E to exit HOST Mode and return control to the monitor program.

5.7.5.4 Download Procedure for Device Connected to XDS/22 Port A

1) Execute IHC command (Page 4-70) to define control characters, if NONE or TEK download protocol to be selected.

2) Execute DL command (Page 4-32) to define download parameters. You cannot select the ASR protocol. Select USER as the source. Be sure your file includes the proper ending character; however, it does not require a starting character. Transfer begins immediately after you press RETURN.

5.7.5.5 Upload Procedure to Host Computer Attached to XDS unit Port D

1) Execute IPORT command (Page 4-79) to configure host port.

2) Execute IHC command (Page 4-70) to define control characters, if NONE or TEK protocol selected.

3) Execute UL command (Page 4-107) to define upload parameters.

4) Execute HOST command (Page 4-62) to begin data transfer.

> **Note:**
>
> For uploading to a PROM programmer through Port C or to the user terminal through Port A, the UL command starts uploading when executed; and you do not need the HOST command.

5) Log on to host system.

6) When prompted, send commands to create file for uploaded object-code from Emulator.

7) If necessary, enter start-upload character from terminal keyboard.

8) Uploading begins immediately.

9) When session complete, log-off host system.

10) Press CNTL/E to exit HOST Mode and return control to monitor program.

5.7.5.6 Examples of Downloaded Data

The following is a TI Standard object-code format with tag characters underlined.

```
0000 8TASK     A0000B000AB0200 B0000BC0008F7EEF
 TASK      021/83 12:32:54              <-- EOM separator record
```

Each character in the printout/display represents four bits (half a byte) of memory.

The same data would appear in TI standard object-file format (first record only) in hexadecimal as follows.

```
3030  3030  3854  4153  4B20  2020  2041  3030
3030  4230  3030  4142  3032  3030  4230  3030
3042  4330  3030  3746  3745  4646
```

In hexadecimal, (first record only) the TI-compressed format would print as follows.

```
0100  0854  4153  4B20  2020  2041  0000  4200
0A42  0200  4200  0042  C000  37F7  EF46
```

The following is a typical set of data records transmitted in Tek format:

```
/00001E0F55005500550055005500550055005500550055005500550055005500550096
/001E1E1E55005500550055005500550055005500550055005500550055005500550096
/003C06155500550055005000001E
```

The following is a typical set of Intel data records.

```
:1E000000550055005500550055005500550055005500550055005500550055005500E7
:1E001E00550055005500550055005500550055005500550055005500550055005500C9
:06003C01550055005500
```

The following is a typical set of Motorola data records.

```
S11D000055005500550055005500550055005500550055005500550055005500550091
S11D001A55005500550055005500550055005500550055005500550055005500550077
S111003455005500550055005500550055005500550067
S9030000FC
```

5.7.6 Modem Communications

You can connect a modem to Port A or Port D for communications with a remote computer. For typical connections and procedure, refer to Section 7.8.

5.7.7 Multiprocessing Communications

You can connect from two to nine XDS/11 or XDS/22 units together to emulate multiprocessing applications. For typical connections and procedure, refer to Section 8.7.1.

5.8 Emulation Communications

Your target system can be set up to communicate with up to 16 input and 16 output devices, numbered 0 - 15, as described in the TMS320C25 User's Guide's INPUT/OUTPUT FUNCTIONS section. This setup requires certain logic elements as detailed in that section.

The emulator DIO command lets you display the 16-bit content of one selected port. The MIO command lets you change the content of one selected port without having to change your program.

5.9 Connecting an XDS/22 to a Target System.

1) Be sure system and XDS power turned off.

2) If necessary, remove device to be emulated from its socket (Section 5.9.1).

3) Unwrap packing material around XDS target connector, if you have not already done so; or remove connector from protective cover.

4) The target connector is a PLCC (plastic-leaded chip carrier) to match a device in an FN surface-mount package. If your target-system device uses that package, plug the target-system connector directly into that socket.

   Be sure to plug the cable in with correct orientation. Holding the target-cable connector so that the lettering is right side up, with the Texas Instruments logo in the upper left, the white mark above and to the right of TMS corresponds to the notched corner of an FN package and the white bar on the connector goes over pin 1 (Figure 5-4).

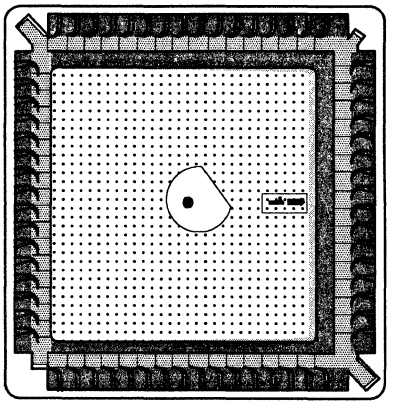


Figure 5-4. FN (PLCC) Socket

5) If your target system uses a device with pin-grid array (GB package), then you must first install the PLCC - PGA (pin-grid array) adapter (Figure 5-5) included in the XDS accessory package onto your target system.

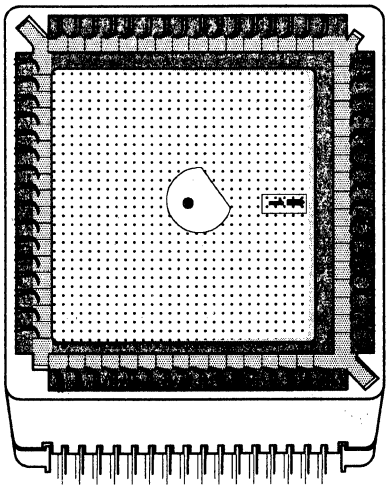


Figure 5-5. PLCC - PGA Socket Adapter

Be sure to install the adapter with the correct orientation. As shown in Figure 5-5, the corner with the small notch (immediately to the left of the adapter pin 1 engraving) goes over the A1 corner of the GB socket. The PLCC target-system connector then plugs into the adapter as previously-described, with the white triangular shape on the connector in the A1 corner and the white bar over the pin 1 engraving in the adapter.

Caution:

The target-connector pins bend easily!

6) If your XDS/22 has the extended-address option, follow this procedure:

   a) Connect extended-address cable ground clip to XDS/22 chassis ground clip.

   b) Plug cable into P3 on Breakpoint/Trace or Breakpoint/Trace/Timing card.

   c) Connect extended-address probe or probes that you plan to use to the appropriate points in your target system.

   d) Connect extended-address probe to connector on extended-address cable.

   For an example, refer to Section 6.9.2 on Page 6-24.

7) For multiprocessing applications, repeat the above steps for each unit. Also interconnect the units as described in Section 8.

5.9.1 Device or Target-Cable Connector Removal Procedure

5.9.1.1 Device

The recommended procedure is to remove the device with a special tool such as an PPS-068-1A1133-L. If that tool is not available, use this procedure:

- **PGA Device.** Carefully lift each corner, one at a time with a small flat-bladed screwdriver. Take your time.
- **PLCC Device.** Carefully pry the device out by inserting a pointed object in one of the long socket notches below or to the left of the small A1 notch and lifting, then repeating that action in the other long notch.

> **Caution:**
>
> **Do not pry on any adapter installed in your system to convert from one 68-pin package to another.**

5.9.1.2 Target Cable

The procedure for removing a target-cable connector is the same as for removing a device when the special tool is not available.

5.10 Ending Emulation

5.10.1 Temporary - Emulation to Continue

At the end of a day, or at any time that you must work on other projects before completing emulation, power-down your XDS/22 as follows:

1) Execute the SAVE command (Page 4-99) to save XDS parameters.

2) Turn XDS/22, terminal, and target system off in any order.

3) If you remove the target connector from the target system, be sure to install the plastic protective cover or press the target connector into conductive foam.

5.10.2 Restoring Operation

1) Replace the target connector if removed. Remember, the pins bend easily!

2) Turn power on in any order.

3) At terminal keyboard, press RETURN twice.

4) If screen does not display menu of commands, press XDS/22 RESET, then press RETURN twice. If menu still does not display, refer to Section 9.6.2.

5) Execute LOAD command to transfer saved parameters out from the emulator-card non-volatile memory.

6) Execute all configuration commands (INIT, ITR, MAP, etc.) and press RETURN to accept each displayed parameter value. Executing the LOAD command alone does NOT restore these values.

7) Enter application program, if required, either by downloading from a host computer or through the emulator-card assembler (XA command).

5.10.3 Permanent - Emulation Complete

Upon completion of emulation, follow the procedure for a temporary halt, except that it is your decision, based upon your next emulation project, whether to Save parameters or not. Be sure to protect the target cable connector when not in use.

Pagination

6. Operation - Program Debugging

An XDS/22 with a Breakpoint/Trace/Timing card offers tracing and breakpointing for fast, efficient program debugging.

- Tracing is the collection of selected program occurrences - for example, a certain type memory access of a selected address for a selected data value - for later review, with the option of stopping your program after collecting a selected number.
- Hardware breakpointing is the stopping of your program when a combination of selected events occurs, with the option of stopping after the first occurrence or after a selected number of occurrences.
- Software breakpointing - an emulator-card function - is the stopping of your program when it reaches a selected address, with the option of setting up to 10 such breakpoints.

Topics discussed in this Section include the following:

6.1 The Breakpoint/Trace/Timing Card

The XDS/22 implements breakpoint and tracing with the Breakpoint/Trace/Timing card described in this Section. A Breakpoint/Trace/Timing card also lets you analyze program timing, and has a convenient attachment for a logic analyzer.

6.1.1 General Breakpoint/Trace/Timing Card Operation

The Breakpoint/Trace/Timing card lets you select up to four independent operating states numbered 0 through 3. Each state lets you define one of 31 combinations of its resources:

- Hardware breakpoint
- Tracing operation
- Jump to another state
- Timing analysis of program operations

BTT operation always starts in State 0. For some applications, you remain in State 0; for others, you go through one or more of the remaining states sequentially; and, for still others, you jump to another state.

You can also repeat certain BTT operations from 1 to >FFFF times without having to reenter the command.

6.1.2 Breakpoint/Trace/Timing-Card Commands

6.1.2.1 Command Set

The Breakpoint/Trace/Timing card supports the following commands:

1) BTT (Breakpoint/Trace, Pages 4-16 and 6-4)
2) DBTT (Display BTT card Parameters, Pages 4-24 and 6-5)
3) DT (Display Trace, Page 4-42)
4) DTIME (Display Time Results, Page 4-45)
5) FT (Find Trace, Page 4-53)
6) IBTT (Initialize BTT card, Pages 4-63 and 6-3)
7) IT (Inspect Trace, Page 4-83)
8) XTIME (Execute Timing Analysis, Page 4-113).

6.1.2.2 Command Sequence

You begin any Breakpoint/Trace/Timing-card operation by using the IBTT (Initialize BTT) command to select one of the 31 combinations described above (called options) for each state that you plan to use. You then tell the BTT card what you want to do with these resources by means of the BTT command.

At any time thereafter, you can use the DBTT (Display BTT Parameters) command to see what is defined, the DTIME (Display Time) command to see the total time counted by two of the BTT timers, or the XTIME command to see how much time your program spent in what memory areas.

After program execution, you can use the DT (Display Trace), FT (Find Trace), and IT (Inspect Trace) commands to review the collected traces. You can re-execute the

IBTT command to display the traces in various formats, with or without time information.

6.1.3 BTT-Card Initialization (IBTT Command)

The first IBTT command prompt lets you select options for each state: with display of those options in a two-part table, (0=OPTIONS), or without display (1=BTT); or lets you select a trace-display format (2=TRACE).

```
DISPLAY                                                    ENTER
?                                                          IBTT<CR>
IBTT
 INITIALIZE [0=OPTION, 1=BTT, 2=TRACE] = OPTION            <CR>
```

```
        ADDRESS & DATA
OPTION:| 1  2  3  4  5  6  7  8 |
-------|--|--|--|--|--|--|--|---|
BP     | 2| 1| 1| 1|  |  |  |   |
TR/TRIX|  | 1|  |  | 2| 1|  |   |
JUMP   |  |  | 1|  |  |  |  |   |
R TIMER|  |  |  |  |  |  | 1|   |
P TIMER|  |  |  | 1|  | 1|  | 2 |
```

ADDRESS-ONLY MODE

```
OPTION:|9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F|
-------|-|-|-|-|-|-|-|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
BP     |4|3|3|3|2|2|2| 2| 2| 2| 2| 1| 1| 1| 1| 1| 1| 1| 1|  |  |  |
TR/TRIX|  |1|  |  |2|  |  |  | 1| 1|  | 3| 2| 2| 1| 1|  |  |  |  | 4| 2|  |
JUMP   |  |  |1|  |  |2|  |  | 1|  | 1|  | 1|  | 2|  | 2| 1| 1|  |  |  |  |
R TIMER|  |  |  |  |  |  |1|  |  |  |  |  |  |  |  | 1|  | 1|  | 1|  | 1| 2|
P TIMER|  |  |  |1|  |  |  | 2|  | 1| 1|  |  | 1|  |  | 1|  |  | 2| 1|  |  |  |
```

BP is Breakpoint (Section 6.2), TR/TRIX is Tracing (Section 6.3), and JUMP (Section 6.4) is moving from one state to a non-adjacent state, for example, from State 0 directly to State 2.

"R TIMER" and "P TIMER" stand for Range timer (Section 6.5.1.1) and Point timer (Section 6.5.1.2), respectively.

Options 1 through 8 let you qualify address and data values, options 9 through >1F let you qualify address values only, option 0 is OFF.

You can also select an option, then not use all its features; for example, if you only want one breakpoint, you could select option 2, then, during BTT-command parameter selection, simply not enable the trace feature of that option.

The rest of the IBTT display is:[19]

```
    SELECT AN OPTION FOR EACH STATE:
    STATE 0 OPTION (0=OFF--1F)    = 02                    <CR>
    STATE 1 OPTION (0=OFF--1F)    = 00                    <CR>
    STATE 2 OPTION (0=OFF--1F)    = 00                    <CR>
    STATE 3 OPTION (0=OFF--1F)    = 00                    <CR>
    END OF SEQUENCE STATE (0-3)   = 0                     <CR>
    ALLOW TRIX MODE [0=NO,1=YES] = NO                     <CR>
    EXTERNAL QUALS [0=NO,1=EXT]   = NO                    <CR>

?
```

As previously stated, you enter the desired option number for each state that you plan to use, or accept the default. Note that option 2, one breakpoint and one trace, is the default for State 0; option 0, OFF, is the default for States 1, 2, and 3.

[19] Of course, as with any XDS command, these lines appear one at a time

The END-OF-SEQUENCE parameter defines how many states you want to use. You should coordinate this parameter value with the number of states for which you select options.

The ALLOW . . . YES parameter refers to trace-mode operation (Section 6.3).

The EXTERNAL QUALS prompt enables the extended-address probes if you want to use extended addressing or data operation (Section 6.9).

The default entries select option 2 for State 0, no options, OFF, for States 1 through 3, State 3 as end-of-sequence, no TRIX mode, and no external qualifiers.

6.1.4 Definitions (BTT Command)

6.1.4.1 General

The BTT command selects parameter values for each option that you initialized with the IBTT command. Its first prompt is the same for any IBTT-command action as described in following text.

The next prompts let you select breakpoint, jump, trace, and timing prompts, in that order, for the states and options you selected. For example, if you select S0 (State 0) and one of address-only options 9 through >1F, you get only qualifier and address prompts for State 0.

Refer to the discussions in later text for the exact display. These topics first treat each BTT capability as independent, then discuss combining them.

6.1.4.2 BTT Command Convenience Features

For your convenience, the Breakpoint/Trace/Timing command lets you use the following keys as function keys:

1) ! to advance to the next Option within a state.
2) @ to advance to the next state.
3) # to save and end the BTT command.

On many terminals, entering ! requires pressing SHIFT and 1 together, entering @ requires pressing SHIFT and 2 together, and entering # requires pressing SHIFT and 3 together.

Repeatedly typing ! lets you scroll through the prompts *within the same state* to view or change them. Prompts always appear one after another in the same order.

Typing @ advances the command to the next state, then you can use ! within that state.

Typing # is a quick way to end the BTT command and save new parameters, or tab through all remaining prompts to the end.

You can also advance to the next option with a state by typing Q, then pressing RETURN, any time that the screen displays a current parameter value for you to accept or change.

6.1.4.3 First BTT-Command Prompt

The first BTT command prompt appears for any IBTT-command selection:

DISPLAY	ENTER
?	BTT<CR>

```
BTT
  SELECT [S0, S1, S2, S3, ALL, COUNT, TIME] = ALL
```

Select the default of ALL if you initialized more than one state with the IBTT command or select one of S0 through S4 if you initialized only that one state with the IBTT command.

Select COUNT if you want to use one of the BTT counters; select TIME if you want to set a time limit for program operation.

6.1.5 Definition Display (DBTT Command)

The DBTT (Display BTT) command (Section 6.7.1) lets you see the operating conditions.

6.2 Hardware Breakpoint Operation

You can select hardware breakpoint operation in 1, 2, 3, or all four BTT states, with an independent number of event counts and up to four breakpoint conditions for each state.

Since operation always starts in State 0, you must select an option for State 0 that includes a hardware breakpoint.

For options with more than one breakpoint condition per state (1, >9 - >13), the number of event counts before breakpoint actually happens is the sum of the event counts from each of those conditions.

For example, if you select an event count of 5 in a state with two breakpoint conditions, BTT operation could go to the next state or breakpoint could occur after any of:

- Four occurrences of the first condition and one of the second
- One occurrence of the first condition and four of the second
- Three occurrences of the first condition and two of the second

or any other combination that adds up to five.

6.2.1 Definitions

Hardware breakpoint event
Specified memory-access type on an address in a selected range that contains a specified data byte (as further refined by masking) on the emulator data bus or occurrence of a specified value (as further refined by masking) on the extended-address cable.

Hardware breakpoint
Final hardware state that returns control to the emulator-card monitor program.

6.2.2 Single-State Operation

If you select one breakpoint (for example, Option 2), then breakpoint occurs after the selected number of breakpoint events occur, or you can select collection of a certain number of traces before actual breakpoint occurs.

You must also leave State 0 as the end-of-sequence state.

6.2.3 Two-State Operation

If you also want breakpoint operation in State 1, then you must both define an option for State 1 that includes hardware breakpoint and define State 1 as the end-of-sequence state.

After the selected number of breakpoint events occurs in State 0, operation goes to State 1. Breakpoint finally occurs after the number of breakpoint events selected for State 1, plus any selected delay.

6.2.4 Three- or Four-State Operation

You would perform similar actions to extend breakpoint operation to States 2 and 3, and the BTT card would perform similar actions with breakpoint finally occurring after the last state, plus any selected delay after that state only. For three-state operation, define State 2 as end-of-sequence; for four-state operation, define State 3.

6.2.5 Sequence Repetition

You can also repeat the entire sequence from State 0 to your selected end-of-sequence state from 1 to 65,535 (>FFFF) times. Breakpoint occurs after the final repetition, plus any selected delay.

6.2.6 Hardware Breakpoint Example

6.2.6.1 Initialization

You call the IBTT command and get the display as previously described. For breakpoint operation, the available options are 1-4 and 9 - hexadecimal 1C. For purposes of discussion, assume that you entered Option 1: Two breakpoints.

6.2.6.2 Definition

After accepting a state or states, you get this display, one line at a time, for State 0, Option 1:

```
STATE 0: BREAKPOINT COUNTER
 EVENT COUNT (0-FFFF)   = 0001

STATE 0: BREAKPOINT 1 OF 2
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF
 ADDRESS #1                                          = 0000
 ADDRESS #2                                          = 0000
 ADDRESS MASK (ONES ENABLE)                          = FFFF
 ADDR TYPE [0=INCL, 1=EXCL, 2=SNGL]                  = INCL
 DATA #1                                             = 0000
 DATA #2                                             = 0000
 DATA MASK (ONES ENABLE)                             = 0000
 DATA TYPE [0=INCL, 1=EXCL, 2=SNGL]                  = INCL
```

Since Option 1 has two breakpoints, after accepting or selecting each line in the display above, you get a repeat of the display from the third line which now reads STATE 0: BREAKPOINT 2 OF 2.

Which means that you can set up another breakpoint event, but not another count. The BTT card decrements the total you entered for State 0 for occurrences of either breakpoint event.

For breakpoint, the next and final BTT prompts are:

```
GLOBAL COUNT VALUES
 SEQUENCE COUNT (0-FFFF)          = 0001
 DELAY COUNT (0-7FF)              = 000
 TRACE COUNT (1-7FF,0=INFINITE)   = 000
```

The sequence count is the number of times you want to perform the entire BTT sequence, 1 (once) default.

For the delay count to work, you must have selected an option that allows tracing within the defined end-of-sequence state. The delay count is the number of traces that you want to collect after satisfaction of the breakpoint-event count before the emulator actually halts.

The trace count applies to the XDS tracing function (Section 6.3). For breakpoint operation only, accept the default of 0.

6.3 Breakpoint/Trace/Timing Card Trace Function

As previously stated, tracing is the collection of selected program occurrences for later review, with the option of stopping your program after collecting a selected number.

You can also mark your trace samples with the time of collection, then display these traces with this information in several formats. The Breakpoint/Trace/Timing card stores emulator clock cycles to obtain the time information.

6.3.1 Initialization

As with any BTT operation, you first call the IBTT command. Its power-up defaults select one trace and one breakpoint in State 0 only. You can, of course, select more than one trace function per state and more than one active state, or you can select no tracing within a state. You must select tracing in an active state if you want the BTT card to store trace or event samples.

You also select TRIX[20] operation. This mode lets you set a trace on instruction-acquisition cycles within a specified memory range as well as on *any* memory address accessed by those instructions. The default is to trace only the specified memory range.

You can also select external probes for the trace function. If you enable that option in the IBTT command, the appropriate prompts display during the BTT-command state initialization. You can also select tracing on extended addresses in one state, and on memory accesses in another.

6.3.2 Trace Definitions

The BTT command defines trace qualifiers. The command displays the following, one at a time:

```
STATE 0: TRACE MODE
 TRACE MODE [0=TRACE,1=TRIX]                          = TRACE[21]
STATE 0: TRACE
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF
 ADDRESS #1                                           = 0000
 ADDRESS #2                                           = 0000
 ADDRESS MASK (ONES ENABLE)                           = FFFF
 ADDR TYPE [0=INCL, 1=EXCL, 2=SNGL]                   = INCL
 DATA #1                                              = 0000
 DATA #2                                              = 0000
 DATA MASK (ONES ENABLE)                              = 0000
 DATA TYPE [0=INCL, 1=EXCL, 2=SNGL]                   = INCL
 EXTERNAL PROBES DATA VALUE                           = 00
 EXTERNAL PROBES DATA MASK (ONES ENABLE)              = 00
```

Data prompts and external-probe prompts appear only if selected with the IBTT command. You select address and data as ranges (inclusive =0, exclusive =1) or single points. Remember that inclusive ranges contain their endpoints.

Address and data masks enable or disable individual bits within the qualifier (Section 5.1.12).

---

20 From a command associated with the Breakpoint/Trace card formerly available for XDS units.

21 This and the preceding line appear only if you selected TRIX with the IBTT command.

You can disable a trace function within a state by setting the Qualifier prompt to OFF with the BTT command. This lets you select options that include tracing, then turn them on or off without having to completely reinitialize the BTT card.

6.3.3 Trace-Buffer Display

Before or after execution of your program, you can reexecute the IBTT command to set the trace display into any of four display modes:

```
?                                                              IBTT<CR>
IBTT
 INITIALIZE [0=OPTION, 1=BTT, 2=TRACE]           = OPTION   2<CR>
 TRACE DISPLAY [0=NORM, 1=TIME, 2=DELTA, 3=MARK] = NORM
?
```

You can execute the IBTT command repeatedly to change the trace display to look at the same trace data in a variety of ways.

6.3.3.1 NORMal Mode

Normal mode gives you a tabular display of each requested trace with column headings of index, cycle type, external qualifiers, address, data, and reverse-assembly.

6.3.3.2 TIME Mode

In the Time mode, you also get a tabular display with headers of index, cycle type, time information, address, data, and reverse assembly. Time information is the absolute time from the start of run mode until the time that the trace sample occurred.

If the trace buffer has less than 2047 samples, that is, if it has not wrapped around, then the first sample has the time of the first instruction and all other samples have a higher value.

6.3.3.3 DELTA mode

In the Delta mode, the BTT card marks each trace sample with the time difference from the previous sample. This lets you determine how long execution of an instruction takes, or the time from module start to module end, which is sometimes more useful than an absolute time measurement.

For the first sample, time difference is marked as from zero.

6.3.3.4 MARK mode

The MARK mode lets you define one sample time as time zero, then display all other samples with time offsets from that sample, + for later samples, - for earlier samples.

The default when entering the Inspect-Trace command is for sample 1 to have a delta offset of zero. You can change the marked sample by entering M*xxx* where *xxx* is a valid trace sample index. If the index does not exist, the mark does not change.

The trace display updates immediately upon entry.

6.3.3.5 Example

Refer to Section 3.7 on Page 3-17.

6.4 Jump Operation

You may also want to jump from one state to another, for example, from State 0 to State 2, or from State 1 back to State 0.

After selecting an option with the IBTT command that includes jumping (3,>B,>E,>11,>13,>15>,>17,>19,>1A,>1B), you use the BTT command to set the jump condition(s) as follows (display for State 0, other state prompts similar):

```
STATE 0: JUMP CONDITION DESTINATION
 JUMP TO STATE (0-3)                                          = 0
STATE 0: JUMP CONDITION
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF
 ADDRESS #1                                                   = 0000
 ADDRESS #2                                                   = 0000
 ADDRESS MASK (ONES ENABLE)                                   = FFFF
 ADDR TYPE [0=INCL, 1=EXCL, 2=SNGL]                           = INCL
 DATA #1                                                      = 0000
 DATA #2                                                      = 0000
 DATA MASK (ONES ENABLE)                                      = 0000
 DATA TYPE [0=INCL, 1=EXCL, 2=SNGL]                           = INCL
```

With these prompts you can select the "jump-to" state and the jump qualifications. The data prompts appear only if you select Option 3.

You can also jump to the current state, for example, if you are in State 1, you can jump to State 1. Such a jump clears all BTT counters.

6.5 Breakpoint/Trace/Timing Counter Operations

The Breakpoint/Trace/Timing card has several counters and timers driven from the emulation clock for various convenience functions such as program analysis, hardware breakpoint event counting, sequence or repetition counting counter, and total run-time counting (watch-dog timer). Resolution of each timer is the period of the emulation clock.

As previously mentioned, trace time-stamping information comes from the emulator clock, not from a Breakpoint/Trace/Timing counter.

6.5.1 Program-Analysis Timing

Timers 1 and 2 can be configured as point or range timers for program performance analysis.

6.5.1.1 Point Timer

A Point timer toggles (starts or stops) when program execution reaches a selected memory address or either of two selected memory addresses within a state; or starts upon such a memory access in one state and stops on such a memory access in another state. For either timer, the start and stop addresses do not have to be the same.

```
STATE 0: POINT TIMER 1
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF
 START ADDRESS                                       = 0000
 STOP ADDRESS                                        = 0000
 ADDRESS MASK (ONES ENABLE)                          = FFFF
 ENABLE [0=BOTH, 1=START, 2=STOP]                    = BOTH
 DATA #1                                             = 0000
 DATA #2                                             = 0000
 DATA MASK (ONES ENABLE)                             = 0000
 DATA TYPE [0=INCL, 1=EXCL, 2=SNGL]                  = INCL
 EXTERNAL PROBES DATA VALUE                          = 00
 EXTERNAL PROBES DATA MASK (ONES ENABLE)             = 00
```

6.5.1.2 Range Timer

A Range timer starts or stops every time execution of your program reaches a selected memory address, either of two selected memory addresses, a range of memory addresses, or anywhere in memory *except* a range of memory addresses.

You must start and stop the Range timer in the same state, unless you want it to continue running; in which case, it stops when the command ends.

```
STATE 0: RANGE TIMER 1 START
 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF
 ADDRESS #1                                          = 0000
 ADDRESS #2                                          = 0000
 ADDRESS MASK (ONES ENABLE)                          = FFFF
 ADDR TYPE [0=INCL, 1=EXCL, 2=SNGL]                  = INCL
 DATA #1                                             = 0000
 DATA #2                                             = 0000
 DATA MASK (ONES ENABLE)                             = 0000
 DATA TYPE [0=INCL, 1=EXCL, 2=SNGL]                  = INCL
 EXTERNAL PROBES DATA VALUE                          = 00
 EXTERNAL PROBES DATA MASK (ONES ENABLE)             = 00
```

You get a similar set of prompts - except the first line says STOP to stop the timer. Start and Stop qualifications do *not* have to be the same.

For timer 1 configured as either a point or range timer, the BTT card also computes the average time spent at the address or in the range selected for timing. The formula is

Average time=(total time)/(# of timer starts).

Thus, if timer 1 were timing a subroutine, the average would indicate the average time spent in the subroutine during program execution.

The average time for timer 2 is not available.

6.5.2 Hardware-Breakpoint Timing

As previously mentioned, one timer counts the breakpoint events in each enabled state.

6.5.3 Sequence Counting

As previously mentioned, you can perform the entire BTT sequence of up to four states from 1 to >FFFF times. The count decrements from your selected starting value after each loop. Hardware breakpoint occurs after completion of the loop that starts *after* the count reaches 0.

You set this counter (called the sequence) by means of the Global-Count-Values part of the BTT command prompt as follows. Note that the default is 1, one performance only.

```
GLOBAL COUNT VALUES
 SEQUENCE COUNT (0-FFFF)          = 0001
 DELAY COUNT (0-7FF)              = 000
 TRACE COUNT (1-7FF,0=INFINITE) = 000
```

6.5.4 Delay and Trace-Sample Counting

You also select delay and trace counts from the Global Count Values shown above. The delay count occurs only in hardware breakpoint operation after the final sequence. Tracing occurs only in a state in which you have defined tracing. Refer to Section 6.3.

6.5.5 Trace Time-Stamping Counter

Trace-sample time stamping comes from the emulator card to avoid tying up the BTT resources for this function. Refer to Section 6.3.

6.5.6 Watch-Dog (Time-out) Timer

You can also end BTT and emulator operation after a selected time interval by using the BTT command time-out function. This function displays prompts for seconds, milliseconds, microseconds, and nanoseconds[22] as shown below. The emulator stops after running the time you select, unless some other condition, such as a hardware breakpoint or a trace-memory-full condition, stops it first.

```
?                                                          BTT<CR>
BTT
 SELECT [S0, S1, S2, S3, ALL, COUNT, TIME] = ALL           6<CR>
 SECONDS       IN DECIMAL (000-999)          = 000
 MILLISECONDS  IN DECIMAL (000-999)          = 000
 MICROSECONDS  IN DECIMAL (000-999)          = 000
 NANOSECONDS   IN DECIMAL (000-999)          = 000
```

If the timer halts the emulator, the following display appears:

```
?
MR;RUN
RUN
 RUNNING
 TIME
 PC= xxxx ST0= xxxx ST1 =xxxx
```

where TIME is the halt code, and PC, ST0, and ST1 are the hardware register values.

This time-out option is independent of the trace-memory-full (TMF) and hardware breakpoint (HBP) stop conditions.

> **Note:**
>
> You cannot use the time-out stop condition in alternate-run mode if the emulator is already running.

[22] The card logic rounds your entry, if necessary, to 2/F where F is the emulation clock frequency in MHz.

6.6 Combinations

As indicated by the the IBTT-command Option tables, you can also combine various BTT operations such as breakpoint and jump or breakpoint and timers or breakpoint and tracing.

6.6.1 General Procedure

Regardless of the combination, the procedure is:

1) Select an option with the IBTT command that contains the desired combination

2) Use the BTT command to set all the conditions

3) If selected, the breakpoint prompt(s) appear first in each state.

6.6.2 Breakpoint and Tracing

A very common combination is tracing and breakpointing in the same state. The usual settings are for the breakpoint to move operation to the next state. You can then examine the trace buffer later with the DT or IT commands.

If, however, the trace-halt condition occurs first, the emulator stops; that is, you cannot move from one Breakpoint/Trace/Timing card state to another by means of tracing.

6.6.3 Breakpoint and Jump

Setting both a breakpoint and a jump in the same state gives you tremendous flexibility in programming debugging. For example, you can set breakpoint on a desired condition; jump on an undesired condition. Then, if the desired condition occurs first, you go to the next state; if the undesired condition occurs first, you go elsewhere. If they occur simultaneously, jump wins.

For example, suppose you want a breakpoint on a write to variable I in location >F010 in subroutine X. You could define the following:

1) Begin in State 0.

2) Jump to State 1 if an instruction acquisition occurs at the beginning of subroutine X.

3) In State 1, do hardware breakpoint on memory write to location >F010, or jump back to State 0 on instruction acquisition outside range of subroutine X.

Or set breakpoint on a range of addresses, and set jump on one specific address in that range. You get breakpoint after the selected number of near misses, but jump if the program hits the one address.

6.7 Other BTT Card Commands

6.7.1 DBTT command

The DBTT command shows which parameters you have entered in the BTT sequence. Since this command is intended to show the sequence flow (start state, stop state, jump-to state) more than the particular enable conditions, the display does not include the address mask, data mask, and external qualifiers.

Each state displays with its corresponding data and all inactive states marked. The outer loop, delay, and trace counts display along with a message indicating if a hardware breakpoint or trace memory full condition is impossible. The timeout value also displays if the TIME option is enabled.

For example,

```
?
DBTT
XDS TIMING AND ANALYSIS VERSION 1.0
STATE 0: START
   BP TO STATE 1 IF  0010[IAQ & SNGL AD(F012-F012)]
STATE 1:
   BP TO STATE 2 IF  0001[IAQ & INCL AD(F010-F020)]
   JUMP TO STATE 0   IF  [IAQ & SNGL AD(F013-F013)]
STATE 2: STOP
   BP (END SEQ)  IF  0001[MA & SNGL AD(F009-F009)]
STATE 3:
   ****INACTIVE
SEQUENCE COUNT=0001      DELAY COUNT=0000      TRACE COUNT=0000
NOTE: TMF CANNOT OCCUR
STOP ON TIME (SEC .. NS) = 010 000 000 000

   ?
```

6.7.2 DTIME command

The DTIME command displays the total time counted by timer 1 and timer 2. If either counter is not enabled, its time appears as zeros.

The display for timer 1 includes its approximate average time defined as

time 1 avg = (timer 1 total)/(number of times timer 1 started)

The BTT card rounds the timer 1 average to the nearest multiple of 2/F, where F is the emulation clock frequency.

The timer 2 average is not available.

A typical display is:

DISPLAY

```
?
DTIME
                HRS MI SEC MS  US  NS
 TIMER 1:       ... .. .2 503 817 750
 TIMER 1 AVG:   ... .. .. ... .19 250
 TIMER 2:       ... .. .5 477 068 000
?
```

ENTER

```
DTIME
```

6.7.3 XTIME command

The XTIME command displays a sample of either memory use or program execution per address range as determined by the first prompt: RANGE QUAL [MA or IAQ]. This command repeatedly enters the Run mode, accumulates time in timers 1 and 2, then halts and displays the percentage of total time in each address range.

Execution of the XTIME command obviously destroys any existing values in timers 1 and 2.

The ADDRESS INCREMENT determines the number of steps the command uses to cover the selected range. For the default of 0, the command displays one line consisting of time for the entire range.

For an entry of 1, the command displays a line for the first and second addresses in the range, then displays a line for the third and fourth addresses, then for the fifth and sixth, etc.

For an entry of 2, the command displays a line for the first three addresses in the selected range, then a line for the next three, then a line for the next three, etc.

This pattern continues up to an entry number that matches the address size. For any larger entry, the display consists of a single line covering from the first address in the range up to the entry number. For example, with a range from 0 to FF, an entry of 3FF gives a single line for time from address 0 to address 3FF.

The SAMPLE TIME parameter tells the command how long to stay in the Run mode for each of the steps defined above. The larger the number, the longer each step runs. The exact time depends on the emulation clock frequency, the address increment, and the program, so it is not necessarily true that a setting of, for example, 500, runs half as long as a setting of 1000.

Run time may or may not have an effect on the program-execution time displayed, depending on that time, the address step size, and the program itself.

For this reason, you need to coordinate your entries with your program; for example, entering an address increment of 0 and a sample time of 050 for a very long program would not give a meaningful display.

For an immediate end to this command, press RESET on the XDS/22 Operator Panel. Pressing ESC also works, but the command does not end until after the current address increment runs.

For example:

```
DISPLAY                                                          ENTER
?                                                                XTIME
XTIME

 QUAL [OFF, ALL, P(R W IAQ):D(R W):IO(R W):I(A IAQ)] = OFF
 START ADDRESS                                       = 0000   000<CR>
 END ADDRESS                                         = 0000   013<CR>
 ADDRESS INCREMENT (0=ONLY 1 SAMPLE)                 = 0000   1<CR>
 SAMPLE TIME (0=NORM--7FFF)                          = 0000   100<CR>

 ADDR RANGE  AVG: MI SEC MS  US  NS  PERCENT
 000--001    ... .. .. ... ... ...   00.0%   .
 002--003    ... .. .. ... ... ...   00.0%   .
 004--005    ... .. .. ... ... ...   00.0%   .
 006--007    ... .. .. ... ... ...   00.0%   .
 008--009    ... .. .. ... ..2 500   00.8%   .
 00A--00B    ... .. .. ... ... ...   00.0%   .
 00C--00D    ... .. .. ... ..2 500   00.8%   .
 00E--00F    ... .. .. ... ..2 500   00.8%   .
 010--011    ... .. .. ... ..2 250   00.7%   .
 012--013    ... .. .. ... ..2 250   00.7%   .
 014--015    ... .. .. ... 301 500   96.1%   ******************
                                       TOTAL = 99.9%
?
```

The values, shown for example only, indicate that the program spent the most time in addresses >014 and >015. Note that rounding times to the nearest multiple of 2/F causes the total to not be 100%.

The string of asterisks forms a sort of bar graph. If the display had several points of activity, each would have a number of asterisks roughly equivalent to the point's percentage.

6.8 Logic-Analyzer Interface

6.8.1 General

A Texas Instruments LOGIC-SHOW (logic-analyzer) interface offers buffered access to the data and address busses, and also to the BTT card qualifiers. The event and trace-sample signals are also available to let you trigger an oscilloscope or logic analyzer each time your program reaches a specified address or range of addresses.

You can select different modes and options in each BTT card state; but no state-usage indication is available at the logic-pod interface.

Table 6-1. Logic-Analyzer Pinout

A31 -- A16	Q6	Q5	Q4	GRD
A15 -- A00	Q3	Q2	Q1	GRD
D31 -- D16	Q0	$\overline{\text{RUN}}$	CLK	GRD
D15 -- D00	$\overline{\text{EVT}}$	TRC	$\overline{\text{ZRO}}$	GRD

where,

Axx = Processor address lines
Dxx = Processor data lines
Q0 - Q6 = Qualification signals
$\overline{\text{RUN}}$ = Emu running flag
CLK = Sample clock
$\overline{\text{EVT}}$ = Breakpoint Event flag
TRC = Trace sample flag
$\overline{\text{ZRO}}$ = Breakpoint Halt pending flag (Event count decremented to zero)

6.8.2 Logic-Analyzer Pin Description

The logic-analyzer connection lets you see the complete address/data bus and several of the BTT qualifiers.

RUN Emu running flag. Logic 0, emulator running Logic 1, processor in command-entry mode. In Alternate-Run mode, emulator can still be running with $\overline{\text{RUN}}$ signal high.

EVT Event Breakpoint flag. Logic 0 indicates hardware breakpoint event condition satisfied.

TRC Trace sample flag. Logic 1 indicates taking of trace sample.

CLK Sample clock. High-low edge transition indicates valid address, data, and qualifier. Use to sample all signals on logic pod, except $\overline{\text{RUN}}$, $\overline{\text{EVT}}$, TRC, and $\overline{\text{ZRO}}$, which are already synchronized.

The $\overline{\text{EVT}}$, TRC, and $\overline{\text{ZRO}}$ signals follow the CLK signal by one cycle. If you latch these signals with CLK, they are results of BTT evaluation of the previous CLK signal due to BTT-card pipelining.

For example, CLK latching the following indicates that the cycles with addresses >00A000 and >00A006 were traced, but the cycle with address >00A004 wasn't.

CLK 1 Address = >00A000	TRC = don't care
CLK 2 Address = >00A004	TRC = 1
CLK 3 Address = >00A006	TRC = 0
CLK 4 Address = don't care	TRC = 1

$\overline{ZRO}$ Breakpoint Halt-Pending Flag. Logic 0 indicates all breakpoint events finished and event count decremented to zero. Breakpoint occurs immediately, or after any delay count.

Q0 $\overline{IACK}$ Interrupt Acknowledge. Processor received interrupt.

Q1 PIS. Program Memory I/O access. Refer to Table 6-2.

Q2 R/$\overline{W}$ Read/Write. Logic 1 indicates processor read operation; logic 0 indicates processor write operation.

Q3 IAQ. Instruction Acquisition. Target processor prefetched instruction from program memory.

Q4 VPA. Valid Program Address. Logic 1 indicates address-bus contains valid program-memory address.

Q5 DIS. Data Memory I/O Access. Refer to Table 6-2.

Q6 $\overline{BIO}$. Branch on I/O active. Signal comes directly from $\overline{BIO}$ pin of emulator TMS320C2x device.

Table 6-2. Memory-Access Type

Q1	Q5	Memory-Access Type
0	0	Internal, no off-chip memory access
0	1	Off-chip data-memory access
1	0	Off-chip program-memory access
1	1	Off-chip I/O access

6.9 Extended Addressing

6.9.1 General

In addition to the tracing and breakpoint capabilities previously discussed in this Section, an XDS/22 with Breakpoint/Trace/Timing card can trace and breakpoint on addresses and data outside the normal target-system intelligent device's address- and data-bus capacities. This lets you, for example, trace and breakpoint on systems that use memory paging, or lets you check for the presence or absence of enabling signals.

The hardware in an XDS/22 to implement these extended capabilities is the Breakpoint/Trace/Timing card, an extended-address cable (Figure 6-1), and an extended-address probe assembly (Figure 6-2).

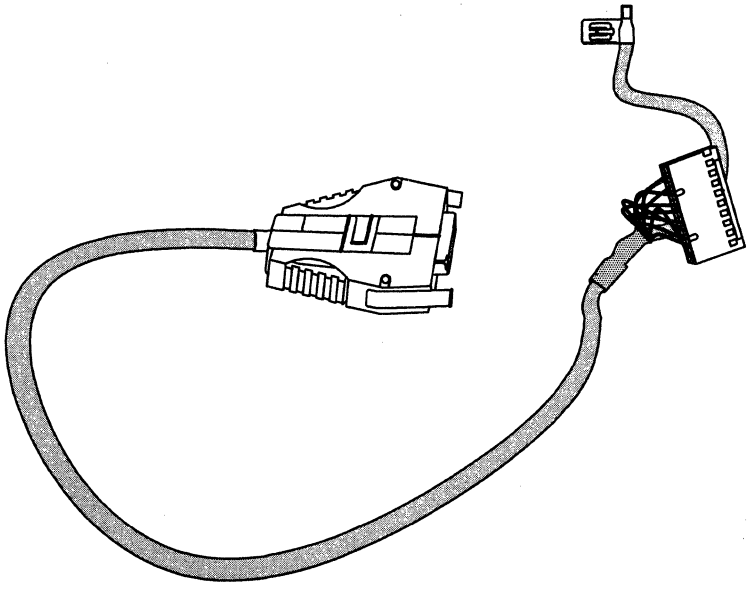


Figure 6-1. Extended-Address Cable

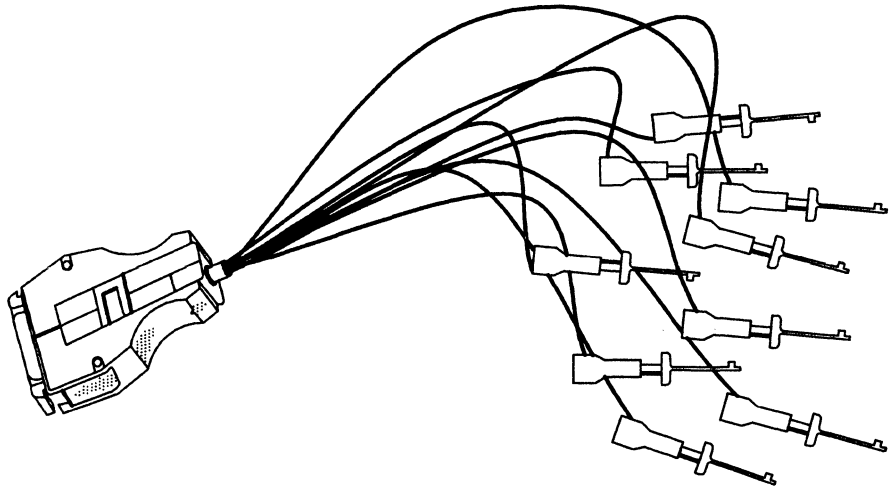


Figure 6-2. Extended-Address Probe Assembly

The small, rectangular connector on the extended-address cable plugs into P3 on the card (Section 9.4). The associated ground clip attaches to a lug on the XDS/22 chassis. The large connector (Figure 6-3) at the other end attaches to a similar connector on the probe assembly, and the color-coded probes connect to appropriate places in your target system.

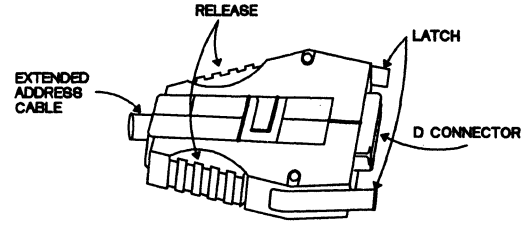


Figure 6-3. Extended-Address Cable Connector

The software part of the extended capabilities is prompts in the INIT command that define what the extended input means, and other commands as described in this Section that enable and qualify the extended address or data values.

At system power-up, the eight clip leads on the extended-probe assembly represent data bits numbered 0 (black lead) to 7 (violet lead). You can assign any number of these - starting at bit 0 and running consecutively - as address bits with the INIT command (Page 4-73). Bit 0 appends to the normal address as the next higher-order address bit. If you select more than one extended-address bit, they also append, with Bit 0 as the least-significant bit.

Any bit not assigned as an address bit with the INIT command remains as a data bit. If you use them, the lowest-numbered data bit becomes the least-significant bit of an extended-data byte that the system handles separately. Any higher-numbered data bits form more-significant parts of that byte.

If you use both extended-addressing and extended-data at the same time, probe bits 0 through A, {0<A≤7}, are address bits and probe bits D, {A+1=D≤8}, through 8 are data bits.

The probes are color-coded, using the standard Electronics Industries Association (EIA) color code (Table 6-3).

Table 6-3. Data Signal Values for Extended-Address Cable Lines

DATA-BIT NUMBER	LINE COLOR	DATA-BIT NUMBER	LINE COLOR
0 LSB	Black	4	Yellow
1	Brown	5	Green
2	Red	6	Blue
3	Orange	7	Violet
GROUND	White	--	--

Texas Instruments recommends that you not change the number of extended-address bits during an emulation session to avoid re-evaluation of associated commands and reset of many parameters.

6.9.2 Extended-Addressing Example

Assume a target system that addresses 128 kilobytes of RAM as two 64-kilobyte pages. In this example, you redefine one extended-data bit as an extended-address bit to switch pages. The remaining seven bits are available for data use, if desired.

Connect the probes and initialize the XDS as follows:

1) Connect the black probe to the target-system MSB address bit and the white probe (ground) to a nearby target-system ground point.

2) Connect remaining probes, if used, to appropriate data points; otherwise, make sure that they do not contact anything in the system.

3) Execute the INIT command to set one extended-address bit and seven extended-data bits by entering 1<CR> at the BP: Number of Extended-Address Bits prompt.

4) After execution of the INIT command, Bit #0 (Black - LSB) becomes the MSB for the extended address, and Bit #1 (Brown) becomes the LSB for the extended-data bits (Table 6-4).

Table 6-4. Modified Line Identification on Extended-Address Cable

COLOR	NUMBER	FUNCTION	COLOR	NUMBER	FUNCTION
Black	0 MSB	Address	Green	5	Data
Brown	1 LSB	Data	Blue	6	Data
Red	2	Data	Violet	7 MSB	Data
Orange	3	Data	White	--	GROUND
Yellow	4	Data	--	--	--

5) Now follow this procedure:

   a) Execute the IBTT command and respond with EXT to the External Quals prompt.

   b) Execute the BTT command. Each address and and address-mask prompt now accepts additional digits for as much extended-addressing as you selected with the INIT command.

6.9.3 Extended-Data Example

This example uses the XDS/22 extended-data feature as an impromptu logic analyzer to check for absence of a target-system device-enabling signal as the reason for unexpected target-system behavior. You could actually check for several signals, but the example uses only one.

You can also combine extended-addressing and extended-data operations.

1) Connect the lowest-numbered available probe to the device pin that generates the signal.

2) Execute the INIT command if you are also doing extended-addressing; otherwise, extended-data operation does not require this command because all eight probes are data by default.

3) Execute the IBTT command and respond with EXT to the External Quals prompt.

4) Execute the BTT command and respond to the prompts as appropriate with the qualifier set to the appropriate activity, the suspect device's I/O number for both breakpoint addresses; any data byte, and an external data byte that depends on the number of probes connected and whether the suspect signal is active-high or active-low. Breakpoint then occurs if the device generates or receives the proper signal.

6.9.4 Extended-Addressing and Extended-Data

You can easily combine the extended capabilities as follows:

1) Determine the number of extended-address bits to be used.

2) Connect probe 0 (black) to the point in your target system that represents the least-significant extended-address bit.

3) Connect any further extended-address bits in order: 1 (brown) through 6 (blue) to address bits in ascending significant order.

4) Connect the next available probe to the point in your target system that represents the least-significant extended-data bit.

5) Connect any further extended-data bits to circuit data bits in ascending significant order.

6) Assign address bits with the INIT command.

7) Follow the procedures in Section 6.9.2 and Section 6.9.3.

6.9.5 Extended-Address Probe Assembly Disconnection

To release the extended-address probe assembly from the extended-address cable, press on both release clips (Figure 6-3) to release the latches, then pull the probe assembly from the cable.

6.10 Software Breakpoint Function

The TMS320C2x Emulator card lets you halt execution of your program (unless you have the Alternate-Run mode in effect) whenever it reaches any of from one to 10 selected memory addresses.

The SSB command (Page 4-102) lets you set one software-breakpoint for each execution. You can repeat the command up to 10 times (with a different address each time) to set up to 10 breakpoints. You get an error message if you try to set more.

To restart the emulator, enter a RUN command (Page 4-98) or a CRUN command (Continue RUN, Page 4-22).

You can clear these breakpoints individually with the CSB (Clear Single Breakpoint) command on Page 4-23 or by means of the DELETE SOFTWARE BREAKPOINTS prompt in the INIT command (breakpoint clears after execution), or you can clear them all at once with the CASB (Clear All Software Breakpoints) command on Page 4-21.

6.11 Alternate-Run Mode

This section describes the Alternate-Run feature which lets TMS32020 emulation continue after any software breakpoint, hardware breakpoint, or trace-count fulfillment occurs, but returns it to the command-entry mode so that you can inspect breakpoints/traces and perform certain other functions.

TMS320C25 emulation continues running after a hardware breakpoint or trace-count fulfillment, but stops after a software breakpoint, regardless of the mode.

Topics include:

6.11.1 General

The Alternate-Run mode lets you see the results of breakpoints and traces while the emulator continues to execute your program.

Execution time in Alternate-Run mode is the same as in Normal mode, unless the monitor program must change or display a memory or register location. In that case, execution takes longer.

You may encounter other problems, too, as detailed in later text; therefore, Texas Instruments recommends the Alternate-Run mode only for analyzing realtime run activity with minimal program intervention. The normal mode is superior for program development.

6.11.2 ARM Command

Syntax

ARM<term>

MEMORY COMMANDS PERMITTED? [0=NO, 1=YES] = NO

The Memory-Commands-Permitted parameter lets you use certain memory-access commands after the emulator would have halted if you weren't in the Alternate-Run mode. Table 6-5 shows the commands affected.

A response of Yes may cause altered conditions for the program in progress. Other commands affected by enabling this parameter (See Command Description section, ARM command) also require that the emulator halt during each memory access, which affects execution time.

Table 6-5. Command Availability in Alternate-Run Mode

ALWAYS AVAILABLE			AVAILABLE WHEN MEMORY COMMANDS PERMITTED		NEVER AVAILABLE		
ARM	HELP	RUN	DDM	IPM	ACC‡	FSM¶	PC‡
BGND	IBTT	SAVE	DIO	MDM	AR(n)‡	GREG‡	PM¶
BTT	ICC	SNAP	DPM	MPM	ARB¶	HM¶	RTR
CRUN	ID	SOR	FILL	XA	ARP§	HOST	S(n)
DBTT	IHC	STOP	IDM	XRA	C¶	INIT	SS
DES	IMD	THALT			CASB	INTM§	SSB
DHS	IPORT	XTIME			CSB	IR	SXM¶
DISARM	IT				CNF¶	MAP	T‡
DPS†	LOAD				DL	MR	TC¶
FT	LOG				DP§	OV§	TXM¶
GHALT	RCC				DR	OVM§	UL
GRUN	RESTART				FO¶	P‡	XF¶

†No values displayed while running.
‡Register.
§Value in Status-Register 0 (ST0)
¶Value in Status-Register 1 (ST1)

6.11.3 DISARM Command

Syntax
DISARM<term>

The DISARM command changes the emulator from alternate to normal run mode; however, if the processor is running, it continues to run until meeting one of the halt conditions described in Table 6-6.

Table 6-6. Halt Conditions

CODE	DEFINITION
ABORT	<ESC> entered after entry of RUN command, but before execution started.
HBP	Hardware breakpoint interrupted program execution. (HBP display includes trace sample being processed when interrupt occurred.)
KEY	A keyboard entry interrupted program execution.
MULTI	Multiprocessing configuration resulted in interrupt.
PERR	Parity error halted program execution.
POR	System power up.
RES	RESET on XDS Operator Panel pressed.
RUNNING	Processor still executing instructions.
SBP	Software breakpoint interrupted program execution.
SS	Single-step command entered.
STOP	Program running in ARM received a STOP command.
TIME	BTT Time-out
TMF	Trace memory full.

6.11.4 STOP Command

Syntax
STOP<term>

The STOP command immediately halts emulator execution.

6.11.5 Alternate-Run Mode Characteristics

6.11.5.1 Command Restrictions

You can enter any command between execution of ARM and execution of any Run-type command; however, many commands are not available, or available only in a restricted form with the emulator running in the Alternate-Run mode as shown in Table 6-5.

6.11.5.2 Memory Commands

Be careful when using memory commands in the Alternate-Run mode. In addition to the change in processing time previously mentioned, changing memory with the emulator running can change the execution path.

6.11.5.3 Trace Commands

The number of trace samples displayed may differ slightly in the Alternate-Run mode, since tracing halts without waiting for processor acknowledgment.

6.11.6 Alternate-Run Mode States

Figure 6-4 shows the relationships between the Alternate-Run mode commands and the emulator state. The latter depends upon the following independent conditions:

Normal/Alternate run mode
Halted/Running mode
Monitor/Polling active

6.11.6.1 Normal/Alternate Mode

The ARM command places the emulator in the Alternate-Run mode. The DISARM command returns it to the Normal run mode.

6.11.6.2 Halted/Running Mode

In this mode, the emulator is either halted, that is, no program running; or running, that is, executing your application program.

6.11.6.3 Monitor/Polling

The emulator powers up in the command-entry mode with ? displayed. Entry of a CRUN (Continue Run), RUN, GRUN (Group Run), or TRUN (Total Run) command activates a polling mode wherein the emulator looks for breakpoints or keyboard entry. Upon detection of either, the monitor program returns to the command entry mode.

6.11.6.4 Combinations

1) Alternate/Halted/Monitor State

Similar to the Normal/Halted/Monitor state until the emulator receives a RUN-type command. At that time, the emulator goes into the Alternate/Running/Polling state. An SS command executed in this state activates the trace function, executes one instruction, then returns to this state through the Alternate/Running/Monitor state.

2) Alternate/Running/Monitor State

Emulator runs with status light #4 lit and with the trace buffer available for display. A RUN or CRUN command reactivates the breakpoint/trace functions. You cannot execute the SS command in this state. A STOP command returns the emulator to the Alternate/Halted/Monitor state. The DISARM command prepares the emulator to halt on the next breakpoint event.

3) Alternate/Running/Polling State

In this state, the emulator waits for a breakpoint condition or keyboard entry, but cannot execute any commands. For either condition, breakpoint/trace operations halt and the emulator enters the Alternate/Running/Monitor state. You can return the emulator to this state by executing a RUN-type command.

4) Normal/Halted/Monitor State

Power-up emulator condition. Pressing RESET on the XDS unit operator panel, hardware breakpoints, or keyboard entry while running also set emulator to this state. A RUN command activates the Breakpoint/Trace card and starts the emulator running.

5) Normal/Running/Monitor State

Holding condition before actually halting the emulator after an alternate-run session. From this state, you can define a breakpoint to halt the emulator or return it to the alternate-run mode at any time by executing the ARM command. Entry of a RUN-type command activates the breakpoint/trace card, and returns the XDS system to the Normal/Running/Polling mode.

6) Normal/Running/Polling State

The emulator enters this state from either the Normal/Halted/Monitor state or the Normal/Running/Monitor state by means of a RUN-type command and can only leave this state by means of a breakpoint condition. When a breakpoint condition occurs, the emulator stops executing the program and returns to the Normal/Halted/Monitor state with ? displayed and waits for the next command input. You can also inspect the trace buffer.

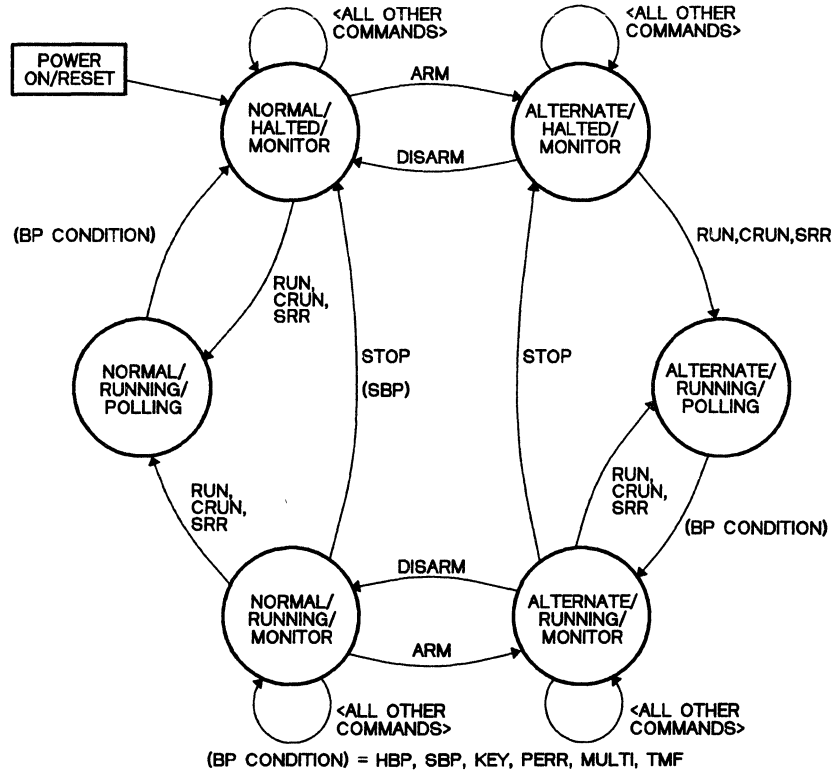


Figure 6-4. Alternate-Run Mode State Diagram

Pagination

7. XDS/22 Connections to External Devices

This section describes connection of a number of specific peripheral devices to a single XDS/22. For multiprocessing connections, refer to Section 8.

For more general communications information, refer to Section 5.7.

Unless otherwise stated in this Manual or in other manufacturer's documentation, no connecting cable should be longer than 25 feet (7.6 meters).

All XDS/22 ports have female DB-25 connectors.

To avoid possible XDS/22 damage from devices that place voltages on normally-unused pins, Texas Instruments recommends that you connect only the interface-cable pins indicated for a particular device, unless the text specifically says otherwise.

This Appendix includes the following trademarks:

- AT&T and DATAPHONE are trademarks of AT&T
- CROSSTALK is a trademark of Microstuf, Incorporated
- DEC, VAX, VMS, VT100, and VT125 are trademarks of Digital Equipment Corporation
- IBM and IBM PC are trademarks of International Business Machines Corporation
- UNIX is a trademark of Bell Laboratories
- Viewpoint is a trademark of ADDS, Inc.

Topics covered are:

7.1 Terminal Connections

Terminals connect to XDS/22 Port A as described in this Section. Section 7.2 describes connections for a Texas Instruments personal computer used as a terminal. Section 2 described similar connections for an IBM PC. Section 7.3 describes connections for a typical development system used as a terminal.

Many terminals, such as Digital Equipment Corporation models VT100 and VT102, use the connections described in this Section. For other models, refer to Section 9.2.2 on Page 9-18 and the terminal documentation.

Topics include:

7.1.1 Cabling

Connect a terminal to XDS/22 Port A by means of the cable shown in Figure 7-1. For multiprocessing, connect to XDS unit #1 only and be sure not to connect anything to XDS Port A pins 17 or 25.

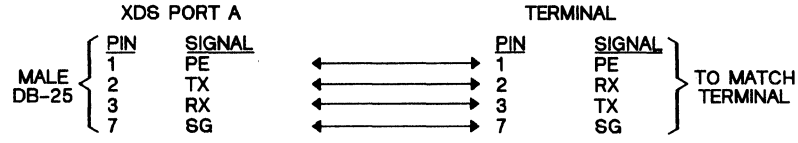


Figure 7-1. Cable, XDS/22 Port A to CRT Terminal

7.1.2 XDS/22 Setup

Table 7-1 shows the settings for S3 and S4 on the Memory Expansion/Communications card as viewed with the card *installed*, which is the usual viewpoint for checking the settings. Switch segments are ON to your right, OFF to your left.

On the other hand, Figure 7-2 shows switch settings as viewed with the card physically removed from the XDS and held with the three card-edge connectors up. Switch segments are ON to your left, OFF to your right.

The settings of S1 and S2 are important only if you are connected to a host computer or if you have a multiprocessing application. Refer to Section 7.6 or Section 8.

Table 7-1. Communications Switch Settings for XDS/22 Operation with many CRT Terminals

SWITCH POSITION	DIP SWITCH	
	S4	S3
1	ON	ON
2	ON	OFF
3	OFF	OFF
4	OFF	ON
5	ON	ON
6	ON	OFF
7	OFF	OFF

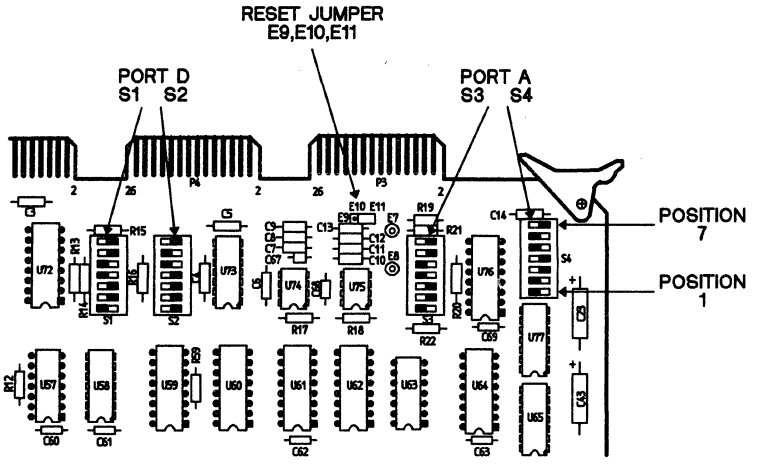


Figure 7-2. Communications Switch Settings for XDS/22 Operation with a General CRT Terminal

7.1.3 Terminal Setup

Set any of the terminal options below that apply to your terminal:

Option	*Comment*
Baud Rate	Set the XDS unit to the terminal baud rate by pressing RETURN twice after power-up. Texas Instruments recommends 9600 baud; however, an XDS/22 works with terminals from 300 to 19,200 baud.
Number of Data Bits	Seven.
Parity	Even.
Number of Stop Bits	Two
Duplex	Full
Auto Scroll	Enabled
Auto Line Feed	Disabled. XDS/22 sends line-feed character (CNTL/J) for each new line.

If you use a DEC VT100 series terminal, its settings must be as shown in Table 7-2:

Table 7-2. DEC VT100 Series Internal Settings

1101 0101 0000 0010

Use the terminal defaults, except for the following:

No Host sync	Advanced-Video	No BRDCSTMBX
No Wrap	No Escape	No REGLS
Scope	No TTSYNC	No Local Echo
BRODCAST	No Remote	No DMA
No Hangup	No Edit Mode	No Autobaud
Set-Speed	No Modem	No TYPE--HD

7.2 Texas Instruments Professional Computer Used as Terminal

You can use a Texas Instruments BusinessPro, Professional, or Portable Professional computer as a terminal for your XDS/22. The procedure is as described in Section 2 for an IBM PC, except that the cable connecting the computer and XDS/22 has DB-25 male connectors at both ends. Or you can use a cable suitable for the IBM PC and a gender adapter.

7.2.1 Downloading with CROSSTALK

You download a file from the computer with the XDS/22 DL (Download) and CROSS-TALK SEnd commands as follows:

```
        DISPLAY                                                    ENTER
?                                                                  DL<CR>
DL
 LOAD OFFSET                                          = 0000       †
 DESTINATION [0=PROGRAM,1=PROM,2=DATA,3=ASM]          = PROGRAM    †
 PROTOCOL [0=NONE,1=TEK,2=ASR,3=VAX]                  = NONE       <CR>
 SOURCE [0=HOST,1=USER]                               = HOST       1<CR>
```

†Your choice.

Now use the CROSSTALK SEnd command to transfer the file.

7.2.2 XDS/22 - CROSSTALK After Downloading

If the object file ends with a colon record followed by an ASCII string, such as in the following example, control automatically returns to the XDS/22 monitor program as indicated by return of the ? prompt.

LAST DOWNLOAD OBJECT-FILE LINES	COMMENT
90000B7F80B7F80B7F80B7F80BF900B000080000F	Last data line
:	Colon record
$string	ASCII String
?	XDS Monitor prompt appears on download completion.

If the object file does not contain a colon record, you must enter a download-end character after the file transfers. You can set this with the XDS/22 IHC (Initialize Host Control) command or use the power-up default of CNTL/W.

You can download file after file if none contains a colon record. Enter the download-end character after the last file.

7.2.3 Uploading Files From the XDS/22

Use the UL (Upload) command with the following parameter entries:

```
DISPLAY                                                    ENTER
  ?                                                                  UL
  UL
   START ADDRESS                                        = 0000       <CR>
   END ADDRESS                                          = 0000       <CR>
   OBJ FORM [0=TI,1=TICOM,2=TEK,3=INTEL,4=MR]           = TI         <CR>
   DATA FORMAT [0=WORD, 1=HI BYTE, 2=LO BYTE]           = WORD       <CR>
   SOURCE [0=PROGRAM, 1=DATA]                           = PROGRAM    <CR>
   DESTINATION [0=HOST, 1=PROM, 2=USER]                 = HOST       2<CR>
   PROTOCOL [0=NONE, 1=TEK, 2=ASR, 3=VAX]               = NONE       0†
```

†Before pressing RETURN, enable the CROSSTALK CApture command to write to a destination file.

Now press RETURN at the XDS/22 terminal to transfer the file.

For example,

A CApture file might look like the following:

```
  -------------------------------- Start of file                    blank
line
  9000B7F80B7F80B7F80B7F80B7F80B7F80B7F80B7F80BF900B00007F49CF
  :
  ?
```

You must delete the blank line at the beginning of the file before trying to download this file to the XDS/22.

For object files that do not contain the colon record and source-file downloads, enter the previously-defined "Download End" character after file transfer. The XDS/22 remains in the download mode until it receives an ending character, which lets you send more than one file per session.

7.3 Hewlett-Packard 64000 Development System as a Terminal

This section assumes familiarity with a Hewlett-Packard 64100A or 64110A Logic Development System. Use the standard Hewlett-Packard software package.

Topics include

7.3.1 Cabling

Connect the **TO MODEM** port on a a Hewlett-Packard HP 6400 system to XDS/22 Port A by means of the cable shown in Figure 7-3. It is also permissable to use a 25-conductor ribbon cable wired straight through.

For multiprocessing, connect to the *last* XDS unit in the chain.

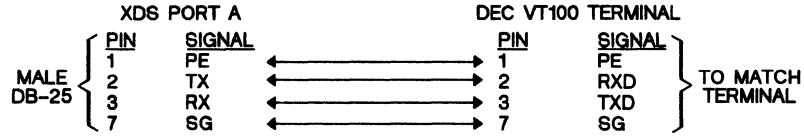


Figure 7-3. Cable, XDS/22 Port D to HP64000 Development System Cable

7.3.2 64100A I/O Setup

Refer to the Hewlett-Packard Service Manual for card location. Orient it with ejector tabs on top, then set the following switches:

S1 - Not Used
S2 - INT Clock - Set A and B (or 2 & 2) to right position
S3 - Select RS-232 operation (down position)
S4 - Mode-Select switches (left = 1 = ON):

SWITCH POSITION	SWITCH SETTING	PARAMETER
1	ON	Two stop bits
2	ON	----
3	ON	Even parity
4	ON	Parity enable
5	ON	7 Data bits
6	OFF	----
7	ON	Baud rate X16
8	OFF	Terminal mode disabled

S5 - Baud rate can be set from 300 to 19200 baud. For the latter, set all switches to the left.

7.3.3 64110A RS-232 Flex Driver Card Setup

7.3.3.1 Switch Settings

S1 - Baud rate can be set from 300 to 19200 baud.
S2 - Mode-Select switches (left = 1 = ON):

SWITCH POSITION	SWITCH SETTING	PARAMETER
1	ON	Two stop bits
2	ON	----
3	ON	Even parity
4	ON	Parity enable
5	ON	7 Data bits
6	OFF	----
7	ON	Baud rate X16
8	OFF	Terminal mode disabled

7.3.3.2 Jumpers

Install 64110A CPU/IO card jumper in J10 TERM position.

7.3.4 XDS/22 Setup

Table 7-3 shows the switch settings on the Memory Expansion/Communications card as viewed with the card *installed*, which is the usual viewpoint for checking the settings. Switch segments are ON to your right, OFF to your left.

On the other hand, Figure 7-4 shows the switch settings as viewed with the card *not* installed, that is, physically removed from the XDS/22 cabinet and positioned with the three edge connectors at the top. Switch segments are ON to your left, OFF to your right.

Set S1 and S2 to match a host computer, if one attached; or the next unit in a multiprocessing application.

Table 7-3. Communications Switch Settings for XDS/22 Operation with an HP64000 Development System

SWITCH POSITION	DIP SWITCH	
	S4	S3
1	ON	ON
2	ON	OFF
3	OFF	OFF
4	OFF	ON
5	ON	ON
6	ON	OFF
7	OFF	OFF

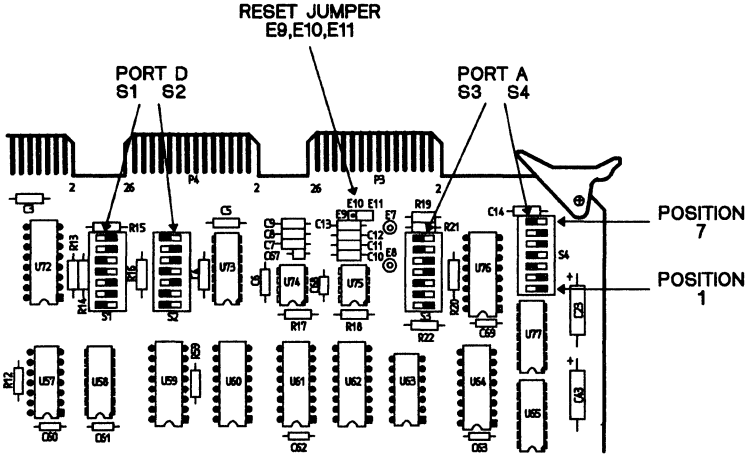


Figure 7-4. Communications Switch Settings for XDS/22 Operation with an HP64000 Development System

7.3.4.1 Additional XDS/22 Setup Information

1) Upload and Download parameters.

UPLOAD	Use 0=NONE Protocol.
DOWNLOAD	Use 0=NONE Protocol.

2) Use <CNTL/W> for upload-end control character.

7.3.5 Operation

7.3.5.1 HP 64000 Booting and Setup

After booting the HP 64000 device (refer to its Installation and Configuration Manual if necessary), press ---ETC--- Softkey until you find the ---DATACOM--- Softkey, then press the ---DATACOM--- Softkey, then press the "terminal" softkey, then enter data in response to prompts.

7.3.5.2 Example for 8-Bit Processor

```
TERMINAL SET UP:
  AUTO LINE FEED?                                          NO
  LOCAL ECHO?                                              NO
  WAIT FOR ECHO DURING UPLOAD?                             NO
  DOWNLOAD START SEQUENCE (0-6 CHAR)?                      "1f"
  SOURCE END OF FILE CHARACTER?                            17H
  FORMAT USED FOR ABSOLUTE FILE TRANSFERS?                 M-HEX
  PROCESSOR DATA BUS WIDTH (#BITS)?                        8
  SMALLEST ADDRESSABLE ENTRY (#BITS)?                      8
  TYPE OF PROTOCOL?                                        XON-XOFF
  UPLOAD PROMPT CHARACTER (0=NONE)                         00H
  XON CHARACTER?                                           11H
  XOFF CHARACTER?                                          13H
  DELAY TIME (ns) AFTER SENDING XOFF DURING DOWNLOAD?      0
```

7.3.6 XDS/22 Setup

1) Power-up XDS unit.

2) At HP 64000 Terminal, press RETURN twice.

3) When screen displays XDS system banner, menu, and ? prompt, type IHC, then press RETURN.

```
DISPLAY                                                    ENTER
?                                                          IHC
IHC
 DEPRESS KEY FOR DOWNLOAD START                =           <CR>
 DEPRESS KEY FOR DOWNLOAD END                  =           <CR>
 DEPRESS KEY FOR UPLOAD START                  =           <CR>
 DEPRESS KEY FOR UPLOAD END                    =           <CNTL/W><CR>
 DEPRESS KEY FOR PASS-THROUGH CHARACTER        =           <CR>

?
```

7.3.7 Downloading Files to the XDS/22

File transfer from an HP64000 to an XDS/22 requires a download from the XDS/22, then an upload to the HP64000.

7.3.7.1 XDS/22 Download

DISPLAY		ENTER
?		DL
<CR>		
DL		
LOAD OFFSET	= 0000	†
DESTINATION [0=PROGRAM, 1=PROM, 2=DATA, 3=ASM]	= PROGRAM	†
PROTOCOL [0=NONE, 1=TEK, 2=ASR, 3=VAX]	= NONE	<CR>
SOURCE [0=HOST, 1=USER]	= HOST	1<CR>
?		

†Your choice

7.3.7.2 HP64000 Upload

The XDS/22 awaits file transfer from the HP64000 as follows:

1) At the HP 64000 terminal, press the "upload" Softkey.

2) Enter the name of the file from which you want to transfer data.

3) The command line might then read something like this:

   COMMAND upload TEST:1:source <CR>

4) The STATUS line might read something like this:

   STATUS: Uploading:TEST: :1:source record#1

   where the record# counts the number of records sent.

5) Upon completion, the XDS/22 terminal screen displays the ? prompt.

7.3.8 Uploading Files From the XDS/22

File transfer from an XDS/22 to an HP 64000 requires an upload from the XDS/22, then a download from the HP 64000.

7.3.8.1 XDS/22 Upload

DISPLAY		ENTER
?		UL <CR>
UL		
START ADDRESS	= 0000	<CR>
END ADDRESS	= 0000	8<CR>
OBJ FORM [0=TI,1=TICOM,2=TEK,3=INTEL,4=MR]	= TI	<CR>
DATA FORMAT [0=WORD, 1=HI BYTE, 2=LO BYTE]	= WORD	<CR>
SOURCE [0=PROGRAM, 1=DATA]	= PROGRAM	<CR>
DESTINATION [0=HOST, 1=PROM, 2=USER]	= HOST	1<CR>
PROTOCOL [0=NONE, 1=TEK, 2=ASR, 3=VAX]	= NONE	0†

†Before pressing RETURN the final time, execute a download from the HP 64000 as follows.

7.3.8.2 HP 64000 Download

1) Press "download" Softkey.

2) Enter name of file to which you want to transfer data, then press RETURN twice. For example,

COMMAND download TEST:1:source<CR><CR>

3) The HP 64000 uses the first RETURN and sends the second one to the XDS/22.

4) The STATUS line displays:

STATUS: Downloading:TEST: :1:source record# 1

The record# counts the number of records received.

Upon completion, the XDS/22 terminal screen displays:

?

7.4 DATA I/O Model 19 or 29A PROM Programmer Connection

7.4.1 Cabling

Connect a DATA I/O Model 19 or 29A PROM Programmer or a serial printer to XDS/22 Port C by means of the cable shown in Figure 7-5.

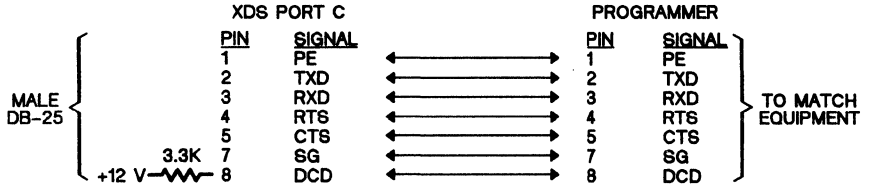


Figure 7-5. Cable, XDS/22 Port C to DATA I/O Programmer

7.4.2 DATA I/O Setup

7.4.2.1 Communication Parameters.

Refer to the manual for switch locations, then set the following:

Baud Rate 19200
Parity Even
Stop bits 2

7.4.2.2 Object-Code Format

To select the object code, enter the following on the DATA I/O:

DISPLAY	ENTER
SELF-TEST OK	SELECT
SELECT CODE	B3 START
FORMAT 50 ^	

Now enter a two-digit format code from the following table:

XDS FORMAT	DATA I/O CODE	REMARK
TI†	90	TI SDSMAC
TEK	86	Tektronix Hexadecimal
INTEL	88	Intel MCS-86 Hexadecimal
MR	87	Motorola Exorcisor

†In this format, the DATA I/O expects '0' as the first tag; however, this tag is NOT present when uploading from the XDS/22, and may not be present when transferring a file from the host computer.

The screen displays:

DISPLAY	ENTER
FORMAT NO ^00xx	START
FORMAT xx	START
FORMAT NO 0xx **	*none*

where xx is the format code that you entered. The first START displays the code for review, the second enters it.

7.4.3 XDS/22 Setup

Enter the IPORT command to configure Port C as follows:

```
DISPLAY                                                          ENTER
?                                                                IPORT<CR>
IPORT
 PORT [0=HOST("D"), 1=LOG/PROM("C")]            = HOST   1<CR>
 BAUD [19.2K,9.6K,4.8K,2.4K,1.2K,600,300,110] = 19.2K <CR>
 PARITY [0=OFF, 1=ODD, 2=EVEN]                  = OFF    2<CR>
 STOP BITS [0=2, 1=1]                           = 2      <CR>
 BITS/CHAR [0=7, 1=8]                           = 7      <CR>

?
```

7.4.4 XDS/22 Data Transfer

Send data from the XDS/22 to the Programmer as follows:

```
DISPLAY                                                          ENTER
FORMAT NO 0xx **                                                 COPY
COPY DATA FROM                                                   PORT
POR^ADDR/SIZE TO                                                 RAM
CO POR) RAM^ADDR                                                 START
INPUT PORT 0                                                     none
```

Now send data from the XDS unit as follows:

```
?                                                                UL<CR>
UL
 START ADDRESS                                  = 0000           †<CR>
 END ADDRESS                                    = 0000           †<CR>
 OBJ FORM [0=TI,1=TICOM,2=TEK,3=INTEL,4=MR]     = TI             <CR>
 DATA FORMAT [0=WORD, 1=HI BYTE, 2=LO BYTE]     = WORD           <CR>
 SOURCE [0=PROGRAM, 1=DATA]                     = PROGRAM        <CR>
 DESTINATION [0=HOST, 1=PROM, 2=USER]           = HOST           1<CR>
 PROTOCOL [0=NONE, 1=TEK, 2=ASR, 3=VAX]         = NONE           0†

?
```

†As required.
‡2, 3, or 4. Selections 0 and 1 will not work.

The UL command format must match the format selected earlier. When the XDS/22 finishes processing the command, it redisplays the ? prompt.

7.4.5 Data Transfer from Host Computer
7.4.5.1 Object File

If a TI SDSMAC object file doesn't begin with an 'O' tag, either:

1) Insert one on the host computer, then download the file through the XDS/22, or

2) Download the file into the XDS/22 in the TI format, then upload to the DATA I/O Programmer with the TEK, Intel, or MR format.

If the first item, and your file begins with a 'K' tag, replace that tag with an /O/ tag, then change the '7' tag at end-of-record to an '8' tag.

If the file begins with a '9' tag, add a dummy header as the first line of your code as follows:

0hhhhDUMMYbbb8hhhhF

where, h means a hexadecimal digit, b means a blank, 8 means ignore checksum, and F (must be upper case) means end of line.

7.4.5.2 Downloading

First use the XDS/22 DL command, then the HOST command.

```
   DISPLAY
ENTER
   ?
DL<CR>
   DL
    LOAD OFFSET                                        = 0000       <CR>
    DESTINATION [0=PROGRAM,1=PROM,2=DATA,3=ASM]        = PROGRAM    1<CR>
    PROTOCOL [0=NONE, 1=TEK, 2=ASR, 3=VAX]             = NONE       0<CR>
    SOURCE [0=HOST, 1-USER]                            = HOST       <CR>

   ?
        ? HOST<CR>

        HOST
          EXIT          DOWNLOAD            UPLOAD             PASS
        <CNTL/E>   <CNTL/V><CNTL/W>  <CNTL/A><CNTL/Z>    <CNTL/P>
```

The XDS/22 now waits for a file transfer. Enable the DATA I/O device as follows:

DISPLAY	ENTER
FORMAT NO 0xx	COPY
COPY DATA FROM	START
POR^ADDR/SIZE TO	RAM
CO POR) RAM^ADDRO	START
INPUT PORT	0

Now send data from the host computer. The DATA I/O has a timeout, so begin data transfer as soon as possible. The XDS unit receives the data at Port D, then sends it the DATA I/O on Port C.

7.5 Texas Instruments 810 Printer to Port C

Connect a Texas Instruments 810 Printer to Port C on an XDS/22 with the cable shown in Figure 7-6. For multiprocessing, connect to XDS unit #1 only.

None of the switches on the Memory Expansion/Communications card affect Port C operation.

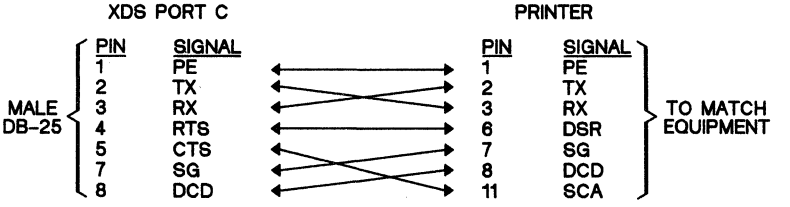


Figure 7-6. XDS/22 Port C to Texas Instruments 810 Printer Connecting Cable

7.6 Computer Connections

Computers connect to XDS/22 Port D.

> **Note:**
>
> Tables in this Section show Memory Expansion/Communications card switch settings as viewed with the card *installed*, which is the usual viewpoint for checking the settings. Switch segments are ON to your right, OFF to your left.
>
> On the other hand, figures in this Section show the switch settings as viewed with the card *not* installed, that is, physically removed from the XDS/22 cabinet and positioned with the three edge connectors at the top. Switch segments are ON to your left, OFF to your right.

Applications in this Section include:

7.6.1 Digital Equipment Corp. VAX 11/780 Computer to Port D
7.6.1.1 Cabling - Interface Adapter

DZ11 Asynchronous Serial Line with A, B, and C, expansion modules SDT.

7.6.1.2 Cabling - Wire List

Connect a VAX 11/780 computer DZ11 ASL to XDS/22 Port D by means of the cable shown in Figure 7-7. For multiprocessing, the computer connects to the *last* unit in the chain.

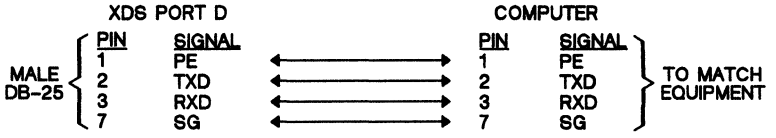


Figure 7-7. Cable, XDS/22 to VAX 11/780 Computer

7.6.1.3 XDS/22 Setup

1) Switch settings

Table 7-4 and Figure 7-8 show the settings for S1 and S2 on the XDS/22 Memory Expansion/Communications card. Set S3 and S4 to match your terminal.

Table 7-4. Communications Switch Settings for XDS/22 Operation with a VAX 11/780 Computer

SWITCH POSITION	DIP SWITCH	
	S2	S1
1	OFF	ON
2	OFF	OFF
3	OFF	OFF
4	OFF	ON
5	ON	ON
6	ON	OFF
7	OFF	OFF

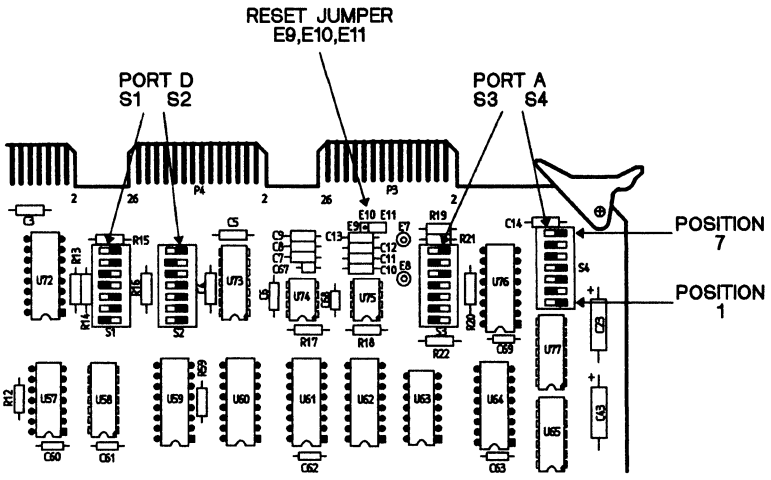


Figure 7-8. Communications Switch Settings for XDS/22 Operation with a VAX 11/780 Computer

2) Additional XDS/22 Setup Information

1) IPORT Parameters

```
DISPLAY                                                    ENTER
?                                                          IPORT<CR>
IPORT
 PORT [0=HOST("D"), 1=LOG/PROM("C")]            = HOST     <CR>
 BAUD [19.2K,9.6K,4.8K,2.4K,1.2K,600,300,110] = 19.2K    <CR>
 PARITY [0=OFF, 1=ODD, 2=EVEN]                  = OFF      <CR>
 STOP BITS [0=2, 1=1]                           = 2        <CR>
 BITS/CHAR [0=7, 1=8]                           = 7        <CR>
```

2) Use 3=VAX for both Upload and Download protocol parameter.

7.6.1.4 Additional Device Information

1) Requires VAX/VMS Operating System, version 3.1 or later.
2) Configure VAX port for VT100 terminal.

7.6.1.5 Downloading Object Files from a VAX Computer

Enter the DL command, then the HOST command. Actual DL-command parameters depend on your specific needs.

DISPLAY		ENTER
?		DL<CR>
DL		
LOAD OFFSET	= 0000	<CR>
DESTINATION [0=MEMORY, 1=PROM, 2=ASM]	= MEMORY	<CR>
PROTOCOL [0=NONE, 1=TEK, 2=ASR, 3=VAX]	= NONE	3<CR>
SOURCE [0=HOST, 1=USER]	= HOST	<CR>

DISPLAY	ENTER
?	HOST<CR>

HOST			
EXIT	DOWNLOAD	UPLOAD	PASS
<^/E>	<^/V><^/W>	<^/A><^/Z>	<^/P>

Now logon, using standard VAX procedure.

Now send a file using the VAX TYPE command, but end the command with <CNTL/V> rather than RETURN.

DISPLAY	ENTER
$	TYPE <filename><^/V>
TYPE <filename>	
$	

The XDS intercepts the <^V> and sends <CR> to the computer to start the download. Your terminal doesn't display the downloaded data; however, it does display the $ prompt upon completion.

$

You then enter the VAX logoff sequence and <^E> to end the XDS Host mode and redisplay the ? prompt.

7.6.1.6 Uploading Object File to a VAX Computer

Define upload parameters (actual values depend on your specific needs):

DISPLAY		ENTER
?		UL<CR>
UL		
START ADDRESS	= 0000	<CR>
END ADDRESS	= 0000	<CR>
OBJ FORM [0=TI,1=TICOM,2=TEK,3=INTEL,4=MR]	= TI	<CR>
DATA FORMAT [0=WORD, 1=HI BYTE, 2=LO BYTE]	= WORD	<CR>
SOURCE [0=PROGRAM, 1=DATA]	= PROGRAM	<CR>
DESTINATION [0=HOST, 1=PROM, 2=USER]	= HOST	1<CR>
PROTOCOL [0=NONE, 1=TEK, 2=ASR, 3=VAX]	= NONE	0†

Initialize terminal (HOST) mode.

DISPLAY	ENTER
?	HOST<CR>

HOST			
EXIT	DOWNLOAD	UPLOAD	PASS
<^E>	<^V> <^W>	<^A> <^Z>	<^P>

```
$
```

Now logon using the standard VAX procedure and create a file using the CREATE command and <^A>:

```
$                                                              TYPE
CREATE <filename><^A>
  CREATE<filename>
  UPLOAD COMPLETE
  $
```

The second message and the prompt appear after upload completion. Now logoff and enter <^ E> to end the XDS Host mode and redisplay the ? prompt.

7.6.2 Tektronix C78500-8540 Computer to Port D

7.6.2.1 Cabling - Interface Adapter

Connects to DCE slot.

7.6.2.2 Cabling - Wire List

Connect a Tektronix Computer to XDS/22 Port D by means of the cable shown in Figure 7-1. For multiprocessing, connect it to the *last* unit in the chain.

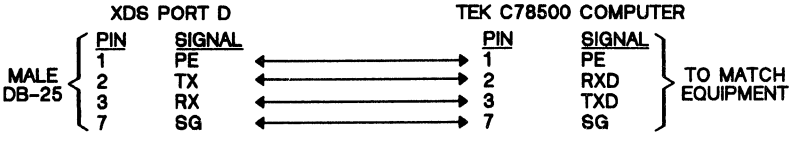


Figure 7-1. Cable, XDS/22 Port D to Tektronix Computer

7.6.2.3 XDS/22 Setup

Table 7-5 and Figure 7-10 show the settings for S1 and S2 on the XDS/22 Memory Expansion/Communications card. Set S3 and S4 to match your terminal or the preceding XDS unit in a multiprocessing application.

Table 7-5. Communications Switch Settings for XDS/22 Operation with a Tektronix Computer

SWITCH POSITION	DIP SWITCH	
	S2	S1
1	ON	ON
2	ON	OFF
3	OFF	OFF
4	ON	ON
5	OFF	ON
6	OFF	OFF
7	ON	OFF

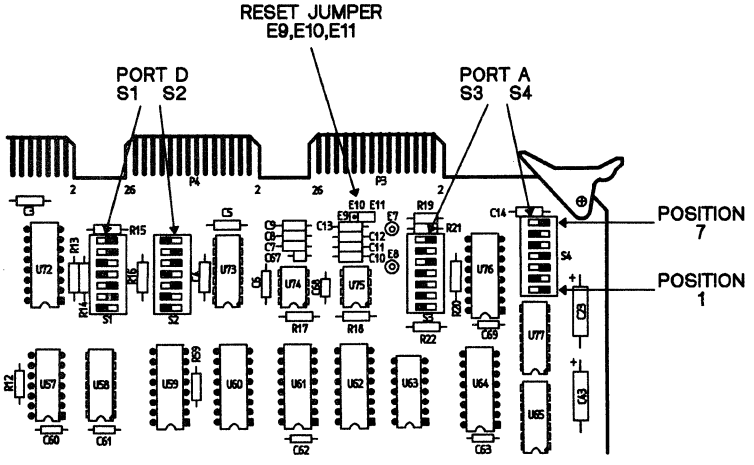


Figure 7-10. Communications Switch Settings for XDS/22 Operation with a Tektronix Computer

7.6.2.4 Additional XDS/22 Setup Information

1) IPORT Parameters

```
?                                                              IPORT<CR>
IPORT
 PORT [0=HOST("D"), 1=LOG/PROM("C")]                 = HOST    <CR>
 BAUD [19.2K,9.6K,4.8K,2.4K,1.2K,600,300,110]        = 19.2K   <CR>
 PARITY [0=OFF, 1=ODD, 2=EVEN]                       = OFF     <CR>
 STOP BITS [0=2, 1=1]                                = 2       <CR>
 BITS/CHAR [0=7, 1=8]                                = 7       <CR>
```

2) Upload and Download parameters.

UPLOAD	Use 1=TEK Protocol.
DOWNLOAD	Use 1=TEK Protocol.

3) Control Characters initialized to default IHC parameters.

7.6.3 Texas Instruments BS300 Computer to Port D

7.6.3.1 Cabling - Interface Adapter

AUX-2 Port on the Two-Channel Communications kit. Reference the Two-Channel I/O Guide: PN 2533313-9701.

7.6.3.2 Cabling - Wire List

Connect a Texas Instruments BS300 computer to XDS/22 Port D by means of the cable shown in Figure 7-11. For multiprocessing, connect it to the *last* unit in the chain.

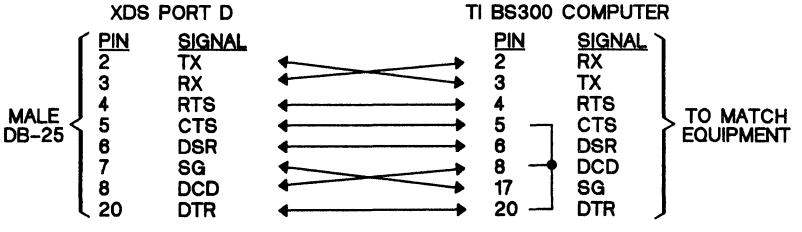


Figure 7-11. Cable, XDS/22 Port D to Texas Instruments BS300

7.6.3.3 XDS/22 Setup

Table 7-6 and Figure 7-12 show the settings for S1 and S2 on the XDS/22 Memory Expansion/Communications card. Set S3 and S4 to match your terminal or the preceding XDS unit in a multiprocessing application.

Table 7-6. Communications Switch Settings for XDS/22 Operation with a Texas Instruments BS300 Computer

SWITCH POSITION	DIP SWITCH	
	S2	S1
1	OFF	OFF
2	OFF	OFF
3	ON	OFF
4	ON	OFF
5	OFF	ON
6	OFF	OFF
7	ON	ON

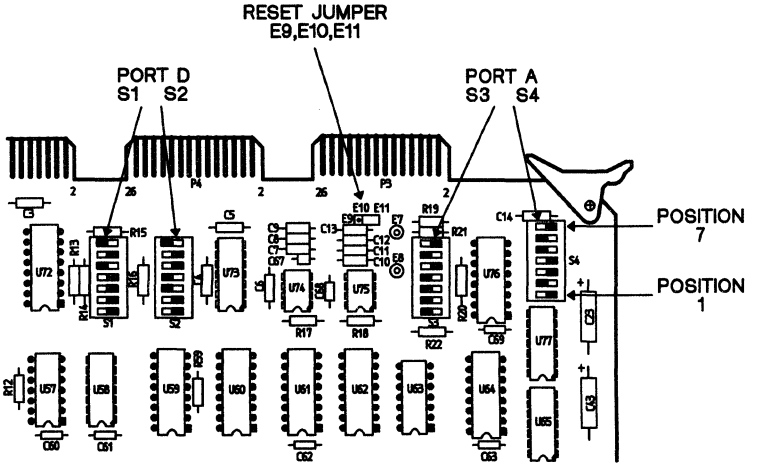


Figure 7-12. Communications Switch Settings for XDS/22 Operation with a Texas Instruments BS300 Computer

7.6.3.4 Additional XDS/22 Setup Information

1) IPORT Parameters

```
?                                                              IPORT<CR>
IPORT
 PORT [0=HOST("D"), 1=LOG/PROM("C")]                  = HOST   <CR>
 BAUD [19.2K,9.6K,4.8K,2.4K,1.2K,600,300,110]         = 19.2K  <CR>
 PARITY [0=OFF, 1=ODD, 2=EVEN]                        = OFF    <CR>
 STOP BITS [0=2, 1=1]                                 = 2      <CR>
 BITS/CHAR [0=7, 1=8]                                 = 7      <CR>
```

2) Upload and Download parameters.

UPLOAD	Use 0=NONE Protocol.
DOWNLOAD	Use 0=NONE Protocol.

3) Initialize Control characters using the IHC command.

```
DEPRESS KEY FOR DOWNLOAD START          = <CNTL/R>
DEPRESS KEY FOR DOWNLOAD END            = <CNTL/A>
DEPRESS KEY FOR UPLOAD START            = <CNTL/Q>
DEPRESS KEY FOR UPLOAD END              = <CNTL/$>
DEPRESS KEY FOR PASS THROUGH CHAR       = <CNTL/S>
```

7.6.3.5 System Generation Information

```
DEVICE TYPE =                   KSR
DSR TYPE (TPD/KSR/K820) =       TPD
TERMINAL TYPE =                 820
INTERFACE TYPE =                9902
SWITCHED LINE =                 NO
BAUD RATE =                     4800
ACU PRESENT =                   NO
FULL DUPLEX MODEM =             YES
ECHO =                          YES
CRU ADDRESS =                   >0B80
ACCESS TYPE =                   RECORD
TIME OUT =                      0
CHARACTER QUEUE SIZE =          20
INTERRUPT =                     4
```

Reserved system memory can become a critical factor. Ensure following parameters set as shown.

```
TABLE =                         8000
I/O BUFFERS =                   1000
```

7.6.4 Texas Instruments 990 Computer to Port D

7.6.4.1 Cabling - Interface Adapter

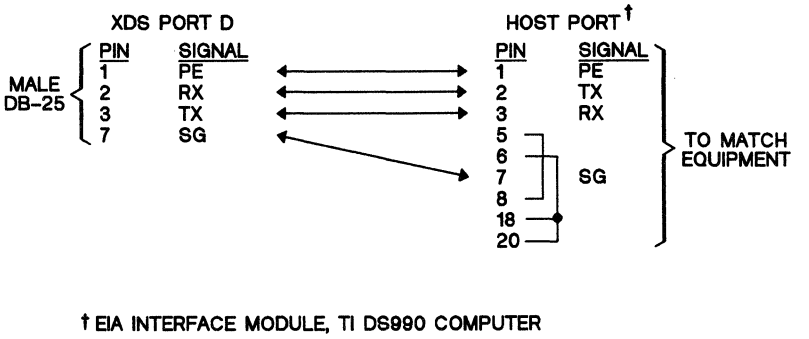
TTY/EIA Terminal Interface, PN 945075-00

Module Jumper Wires/Switch Settings

E1A - E7
E9 - E8
E12 - E11
E17 - E18
E20 - E21
E26 - E25

7.6.4.2 Cabling - Wire List

Connect a Texas Instruments 990 computer to XDS/22 Port D by means of the cable shown in Figure 7-13. For multiprocessing, connect it to the *last* unit in the chain.



† EIA INTERFACE MODULE, TI DS990 COMPUTER

Figure 7-13. Cable, XDS/22 Port D to Texas Instruments 990 Computer

7.6.4.3 XDS/22 Setup

Table 7-7 and Figure 7-14 show the settings for S1 and S2 on the XDS/22 Memory Expansion/Communications card. Set S3 and S4 to match your terminal or the preceding XDS unit in a multiprocessing application.

Table 7-7. Communications Switch Settings for XDS/22 Operation with a Texas Instruments 990 Computer

SWITCH POSITION	DIP SWITCH	
	S2	S1
1	ON	ON
2	ON	OFF
3	OFF	OFF
4	ON	ON
5	OFF	ON
6	OFF	OFF
7	ON	OFF

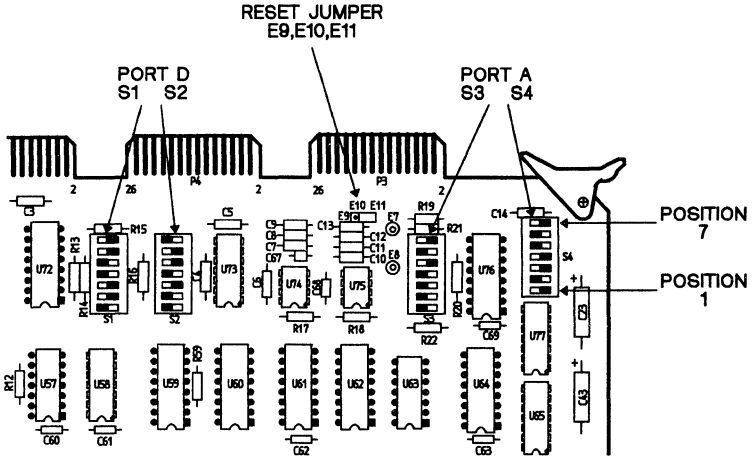


Figure 7-14. Communications Switch Settings for XDS/22 Operation with a Texas Instruments 990 Computer

7.6.4.4 Additional Information

1) IPORT Parameters

```
?                                                          IPORT<CR>
IPORT
 PORT [0=HOST("D"), 1=LOG/PROM("C")]              = HOST   <CR>
 BAUD [19.2K,9.6K,4.8K,2.4K,1.2K,600,300,110] = 19.2K  <CR>
 PARITY [0=OFF, 1=ODD, 2=EVEN]                    = OFF    <CR>
 STOP BITS [0=2, 1=1]                             = 2      <CR>
 BITS/CHAR [0=7, 1=8]                             = 7      <CR>
```

2) Upload and Download parameters.

UPLOAD	Use 0=TI or 1=TICOM Format.
DOWNLOAD	Use 2=ASR Protocol.

7.6.4.5 Device System-Generation Information

DEVICE TYPE =	ASR
CRU =	*Address of 990 chassis slot with TTY/EIA Interface card*
ACCESS TYPE =	RECORD
TIME OUT =	*Select time period in seconds*
CASSETTE ACCESS TYPE =	FILE
CHARACTER QUEUE SIZE =	20
INTERRUPT =	*Interrupt level of 990 chassis slot with card*
CHASSIS =	*Depends on expansion chassis*
POSITION =	*Depends on TTY/EIA card position in 990 chassis*

7.7 Sytek Inc. Local Area Network Connection

7.7.1 Cabling

7.7.1.1 Interface Adapter

Sytek "T-BOX"

7.7.1.2 Wire List

Connect a Sytek Local Area Network to XDS/22 Port D by means of the cable shown in Figure 7-15. For multiprocessing, connect it to the *last* unit in the chain.

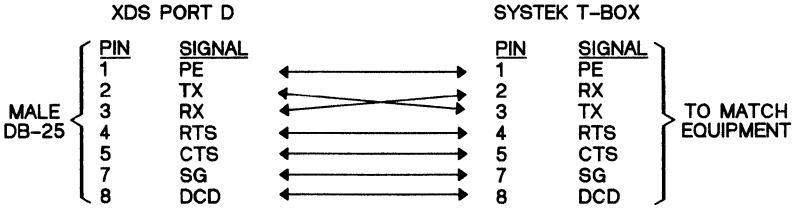


Figure 7-15. Cable, XDS/22 Port D to Sytek Local Area Network

7.7.2 XDS/22 Setup

Table 7-8 shows the settings for S1 and S2 on the XDS/22 Memory Expansion/Communications card with the card *installed*. Switch segments are ON to your right, OFF to your left.

Figure 7-16 shows the switch settings with the card on the bench, with the three tab connectors up. Switch segments are ON to your left, OFF to your right.

Set S3 and S4 to match your terminal or the preceding unit in a multiprocessing application.

Table 7-8. Communications Switch Settings for XDS/22 Operation with a Sytek Local Area Network

SWITCH POSITION	DIP SWITCH	
	S2	S1
1	OFF	ON
2	OFF	OFF
3	OFF	OFF
4	OFF	ON
5	ON	ON
6	ON	OFF
7	OFF	OFF

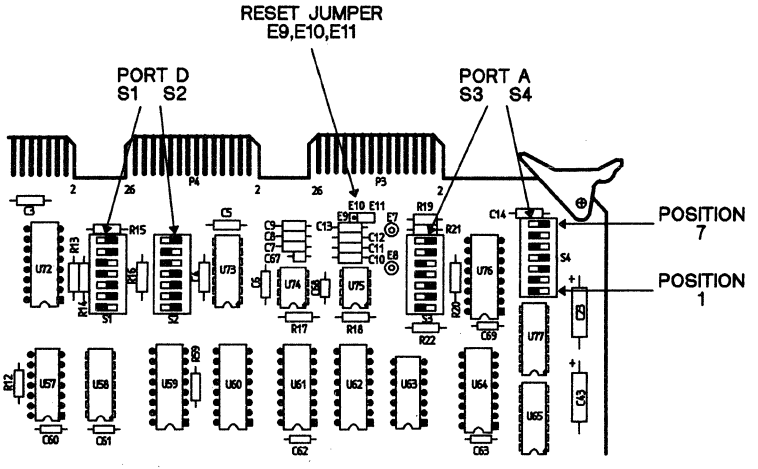


Figure 7-16. Communications Switch Settings for XDS/22 Operation with a Sytek Local Area Network

7.7.3 Additional XDS/22 Setup Information

1) IPORT Parameters

```
?                                                              IPORT<CR>
IPORT
 PORT [0=HOST("D"), 1=LOG/PROM("C")]                 = HOST    <CR>
 BAUD [19.2K,9.6K,4.8K,2.4K,1.2K,600,300,110]        = 19.2K   <CR>
 PARITY [0=OFF, 1=ODD, 2=EVEN]                       = OFF     <CR>
 STOP BITS [0=2, 1=1]                                = 2       <CR>
 BITS/CHAR [0=7, 1=8]                                = 7       <CR>
```

2) Upload and Download parameters.

UPLOAD	Use 0=NONE Protocol.
DOWNLOAD	Use 0=NONE Protocol.

3) Initialize Control characters using the IHC command as follows.

```
DEPRESS KEY FOR DOWNLOAD START          = <CNTL/V>
DEPRESS KEY FOR DOWNLOAD END            = <CNTL/W>
DEPRESS KEY FOR UPLOAD START            = <CNTL/A>
DEPRESS KEY FOR UPLOAD END              = <CNTL/B>
DEPRESS KEY FOR PASS THROUGH CHAR       = <CNTL/P>
```

7.7.4 Additional Device Information

This application requires a UNIX Operating System, version 5.0 or later.

You must use the STTY command to inform the VAX computer of the communications status with the following command sequence.

```
stty susp <CNTL/Y><CR>
stty erase <CNTL/H> crt; biff y<CR>
stty cr0 n10 tabs erase;tabs; TERM-vt100;<CR>
```

7.8 Stand-Alone Modem Connection

7.8.1 Cabling

Connect a stand-alone modem such as an AT&T DATAPHONE II modem or a Racal-Vadic 3450 Series Modem to XDS/22 Port A or Port D by means of the cable shown in Figure 7-17. If the modem does not place voltages on other pins, you can also use a 25-conductor cable wired straight through.

For multiprocessing, Texas Instruments recommends operation from Port D only. Connect to the *last* unit in the chain.

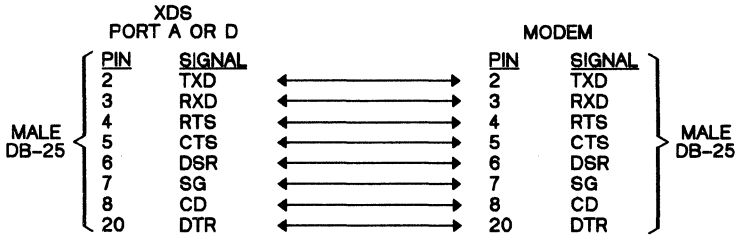


Figure 7-17. Cable, XDS/22 Port to Typical Stand-Alone Modem

7.8.2 XDS/22 Setup

Table 7-9 shows the settings for switches on the XDS/22 Memory Expansion/Communications card with the card *installed*. Switch segments are ON to your right, OFF to your left. For a modem connected to Port A, set S3 and S4. If connected to Port D, set S1 and S2.

Figure 7-18 shows the switch settings for the card on the bench, with the three edge-connectors up. Switch segments are ON to your left, OFF to your right. For a modem connected to Port A, set S3 and S4. If connected to Port D, set S1 and S2.

Table 7-9. Communications Switch Settings for XDS/22 Operation with a Typical Modem

SWITCH POSITION	DIP SWITCH	
	S4,S2	S3,S1
1	ON	OFF
2	OFF	ON
3	ON	OFF
4	ON	OFF
5	OFF	ON
6	OFF	OFF
7	ON	ON

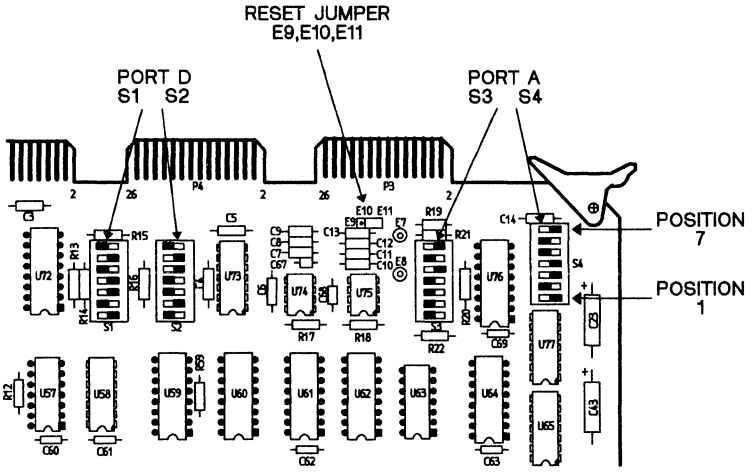


Figure 7-18. Communications Switch Settings for XDS/22 Operation with a Typical Modem

If modem connected to Port A, set S3 and S4 the same as S1 and S2 in Figure 7-18. In that case, set S1 and S2 according to what is connected to Port D.

7.8.3 Additional Information

Communications usually requires the following signals. Pin numbers refer to the RS-232C connector.

TXD (Transmitted Data, Pin 2). Emulator data to modem.

RXD (Received Data, Pin 3). Modem data to emulator.

RTS (Request to Send, Pin 4). Emulator sets to logic 1 to send data from modem to XDS.

CTS (Clear to Send, Pin 5). Modem sets to logic 1 to send data from emulator to modem.

DSR (Data Set Ready, Pin 6). Modem sets to logic 1 at beginning of session to show telephone connection made and enable emulator to modem transfer.

DCD (Data Carrier Detected, Pin 8). Modem sets to logic 1 when data carrier detected.

DTR (Data Terminal Ready, Pin 20). Modem or emulator sets to logic 1 at beginning of session.

Many modems place +12 V on connector pin 9 and -12 V on connector pin 10.

You must place or answer the call manually as described in the modem operations manual before XDS data transfer can begin. With most modems, you can monitor call status by means of front-panel indicators.

Pagination

8. Operation - Multiple Emulators

This section covers operation of from two to nine XDS units connected together in series (daisy-chained) and controlled by a single user.

Topics include:

8.1 General

You can operate up to nine XDS units together to emulate a system using up to nine Texas Instruments microprocessors.[23] You connect the emulators in series (sometimes called daisy-chaining) with the controlling terminal and optional logging device connected to the first unit, and a host system, if used, connected to the last unit.

Figure 8-1 shows a three-emulator chain.

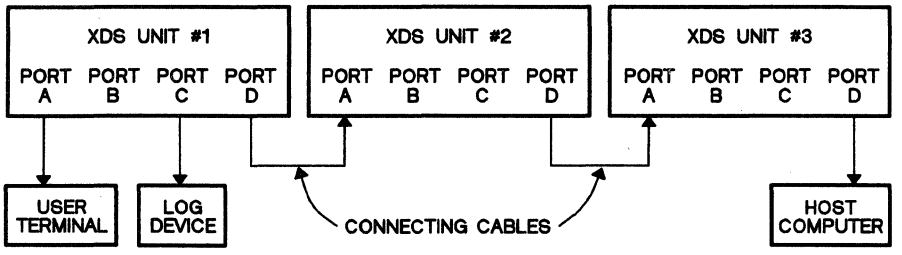


Figure 8-1. Three XDS Units in a Multiprocessing Chain

At system initialization, unit #1, the unit connected to the terminal, is in the foreground and all other units are in the background. The foreground unit can receive XDS commands, background units can execute commands received when in the foreground.

To bring any other unit into the foreground, type #*n*, *n* = 1-9, with the ? prompt displayed on the terminal screen, then press RETURN. To place a unit in the background without bringing any other unit into the foreground, type BGND with the ? prompt displayed.

You can also configure any number of emulators in the chain to work together in what is called a group (Section 8.3). Emulators in a group can start and stop together.

[23] These can be any Texas Instruments microprocessors with the appropriate emulator card in the XDS/22.

8.2 Procedure

1) Physically connect emulators as shown in Figure 8-1 and detailed in Section 8.7.1.

2) Configure Memory Expansion/Communications card switches in each XDS unit as shown in Section 8.7.1.

3) Turn on power at each XDS unit, and wait at least 3 seconds.

4) At terminal keyboard, press RETURN twice.

5) XDS unit #1 performs its autobaud sequence which automatically synchronizes its baud rate to the terminal's baud rate, 9600 bps recommended.

6) The screen displays the following:

```
                    TEXAS INSTRUMENTS
                TMS320C2x XDS VERSION x.x.x

COMMANDS:
   INIT      IPM     IR      RUN       BTT     IT      HOST    IMP
   IPORT     IDM     DR      CRUN      IBTT    FT      IHC     IMD
   IPRM      DPM     MR      SS        DBTT    DT      UL      ID
   ICC       DDM     DIO     RTR                       DL      BGND
   RCC       MPM     MIO
   RESTART   MDM

   LOAD      FILL    DPS     ARM       SSB     DTIME   LOG     GRUN
   SAVE      FIND    DES     DISARM    DSB     XTIME   SNAP    TRUN
   HEX       ITR     DHS     STOP      CSB             HELP    GHALT
   DEC       MAP     DTS               CASB    XA      DV      THALT
   MAG               DMAP              XRA

VARIABLES:
   PC        AR      ACC     ARP       INTM    ARB     XF      C
   ST        S       T       OV        DP      CNF     FO      HM
             GREG    P       OVM               TC      TXM     FSM
                                               SXM     PM

   ?
```

7) Type IMP, then press RETURN

8) The screen displays:

```
AUTOPOLL [0=NO, 1=YES] = NO
```

If you set this parameter to 1, the first emulator in a group (Section 8.3) to stop displays its status whether or not it is in the foreground at that time. If you leave it at 0, that display occurs only if the unit happens to be in the foreground.

9) Executing the IMP command causes XDS unit #1 to send a predefined character sequence to the next physical XDS unit. If this unit exists and returns the proper character sequence, the monitor program in unit #1 assigns it as XDS unit #2.

10) This process repeats down the physical chain until the last unit (#3 in this case) doesn't get a response from the "next" unit or the host, if so-connected. At this time, the last unit sends its ID # to unit #1.

11) The screen displays the identification number of the last emulator in the chain followed by the monitor-program logo for the first unit and a ? prompt.

```
LAST EMULATOR=3
TMS320C2x XDS VERSION x.x.x
?
```

12) If the display doesn't match what you think is connected, check the cabling and internal switches, then reset each emulator and re-execute the IMP command.

13) Unit #1 remains in the foreground until you execute a BGND command (Page 4-15) or set some other unit to that status.

14) Configure unit #1 as one of independent, master, or slave with the start-synchronous parameter of the IMD (Initialize Multiprocessor Mode) command. Units configured as masters and slaves form a group (Section 8.3).

DISPLAY		ENTER
?		IMD<CR>
IMD		
PROCESSOR MODE [0=IND, 1=MASTER, 2=SLAVE]	= IND	<CR>
START SYNCHRONOUS [0=NO, 1=YES]	= NO	<CR>

15) Bring unit #2 into the foreground by typing #2<CR>, then configure it as to mode and start-synchronosity with the IMD command.

16) Repeat last step for each remaining emulator.

17) Initialize Port C on Unit #1 and Port D on the last unit (regardless of what unit is in the foreground) by means of the IPORT command.

DISPLAY		ENTER
?		IPORT<CR>
IPORT		
[0=HOST("D"), 1=LOG/PROM("C")]	= HOST<CR>	
BAUD [19.2K,9.6K,4.8K,2.4K,1.2K,600,300,110]	= 19.2	†<CR>
PARITY [0=OFF, 1=ODD, 2=EVEN]	= OFF	†<CR>
STOP BITS [0=2, 1=1]	= 2	†<CR>
BITS/CHAR [0=7, 1=8]	= 7	†<CR>
?		

†As required.

18) Call each unit into the foreground one at a time and enter the desired program.

19) Conclude the program with a RUN command or send a GRUN (Group Run) command to start all emulators in a group together or send a TRUN (Total Run) command to start all emulators in the chain (group starts together, others start immediately).

20) Halt operation as follows:

    a) If you started an emulator with a RUN command, halt it by calling it into the foreground.

    b) Stop all emulators in a group with a GHALT (Group Halt) command from the foreground emulator.

    c) Stop all emulators in the chain with a THALT (Total Halt) command.

8.3 Group Operations

8.3.1 Definition

A group consists of all XDS units with the start-synchronous parameter set to 1; however, a master unit with that parameter set to 0 performs somewhat like a group unit.

Assigning a unit as a slave automatically sets that parameter to 1; independent and master units must be set individually.

8.3.2 Run/Halt Line

An emulator card contains an open-collector gate with its output tied to the card pin that connects to pin 17 in XDS/22 Ports A and D. After formation of the multiprocessing chain, the line interconnecting pin 17's becomes the Run/Halt (R/H) line. A resistor to V_{CC} on the last emulator in the chain completes the circuit.

Configuring a unit as master or slave with the IMD command enables the gate on that emulator which means that the gate can, under the right conditions, drive the R/H line low. Configuring a unit as independent disables the gate on that emulator which means that the emulator can never control the R/H line.

Configuring an independent or master unit to start synchronously enables other logic on the card to monitor the Run/Halt line. Configuring it as a slave automatically enables that logic.

An independent unit, then, is one that can monitor the Run/Halt line but cannot control it. A master or slave unit can both monitor and control the line.

Configuring a unit as a slave also enables logic on that emulator card which causes it to halt – not only on specific halt commands and conditions such as trace-memory-full or keyboard – but if any other unit in the chain halts. On the other hand, the halt logic on a unit configured as a master responds only to specific halt commands and conditions.

The remaining topics in this subsection further pinpoint the differences between the multiprocessing modes.

8.3.3 Running

For a master or slave unit, a GRUN (Group Run), RUN, or TRUN (Total Run) command causes the gate output to go high (releases the line) which means the logic level on the R/H line now depends on the remaining units connected to that line. When the last unit in the group releases the line, that is, after all units in the group receive a RUN command or you send a GRUN or TRUN command, the R/H line goes high and all units in the group start together.

An independent unit that is configured to start synchronously also starts only when the R/H line goes high, as described above for a master or slave unit. An independent unit that is not configured to start synchronously starts immediately upon receiving a RUN or TRUN command, but does not respond at all to a GRUN command.

A master unit that is not configured to start synchronously starts immediately upon receiving a GRUN, RUN, or TRUN command, regardless of the R/H line status.

8.3.4 Halting

Emulators in a multiprocessing chain can halt for breakpoints, errors, or in response to certain commands, etc. just as in normal emulation. When a master or slave unit in a group halts for any reason, it drives the R/H line low. When that line goes low, all slave units finish the command in progress, then halt.

If the halt occurred because of an error or breakpoint condition in one emulator, the remaining master and independent units continue to run. If the halt occurred because of a GHALT (Group Halt) command, master units in the group also halt after finishing the command in progress, but independent units continue to run. If the halt occurred because of a THALT (Total Halt) command, all units stop.

When a stand-alone emulator halts, it displays its status, that is, the program-counter number, register content, and a halt code. When an emulator in a chain halts, it does the same thing; however, you get no display unless:

1) The emulator that halts is the foreground unit, or

2) You turned Autopolling on when you executed the IMP command.

In the latter case, the XDS system monitor program automatically switches the first emulator that halts into the foreground for a status display. Logic on each emulator card lets it detect that it stopped with the R/H line high, that is, it was the first to stop.

8.4 Background Operation

If you send a RUN, GRUN, or TRUN command, the affected emulators go into the Background mode of operation, that is, offline; however, they may still be running programs.

With all units in the background, the terminal displays the following message:

```
BACKGROUND      or      AUTOPOLLING
   ??                        ???
```

The actual display depends on the AUTOPOLL parameter, defined at execution of the IMP command.

With all units running in background, the only acceptable input from the keyboard is the following:

#n<CR> Brings Unit #n into foreground to accept further keyboard input.

/n<CR> Unit n displays its current status as shown in Table 8-1

<ESC> Last unit to go into background mode returns to foreground operation with display of banner.

Table 8-1. Status Messages

DISPLAY	COMMENT
READY FOR INPUT	? prompt displayed. Monitor program waiting for command entry.
WAITING TO RUN	Emulator in background waiting to begin running. This could occur if BGND command sent before RUN command executed, or if unit in group waiting for others in that group to begin their runs.
RUNNING	Emulator currently executing program.
WAITING TO OUTPUT	Emulator in background waiting to link with an I/O port to output data.
RUN MODE COMPLETE	Emulator in group halted after GHALT or THALT command.

8.5 Multiprocessing Emulator Reset

An XDS unit in the multiprocessing mode has Global and Local reset capabilities. Jumpers at E9 through E11 on the Communications card (Section 9.2, page 9-17) determine reset conditions.

- **Global RESET**: Jumper E9 to E10.Pressing RESET on any unit configured as Global resets all units through the common reset line on the communication card and also removes all units (global and local) from multiprocessing mode.

  To reenter MP Mode after global reset, execute the IMP command, however, you don't need to reconfigure the communication ports or re-execute the IMD command.

- **Local RESET**: Jumper E10 to E11.Pressing RESET on any emulator configured as local resets only that unit. After local reset, you must re-execute the IMP command if the unit is between the foreground unit and the host computer.

> **Note:**
>
> Global and Local as defined above also applies to power-on reset; that is, the reset signal generated each time that you turn an XDS/22 on.

8.6 Flowcharts of Processing Modes

The flowcharts in Figure 8-2, Figure 8-3, and Figure 8-4 summarize the relationships among MASTER, SLAVE, and INDEPENDENT emulators. Figure 8-5 shows how the emulator responds to a halt condition.

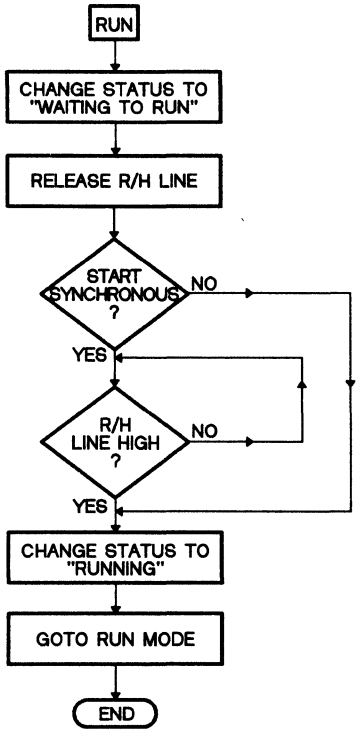


Figure 8-2. Master Unit after Receiving RUN/TRUN/GRUN Command

1) Execution of the IMD command causes a unit designated as a master to drive the R/H line low.

2) Upon receiving a RUN command, a master emulator releases the line and goes into the WAITING TO RUN mode.

3) If configured as START SYNCHRONOUS, a master emulator waits for the line to go high before RUNNING. If not so-configured, it starts immediately.

4) If the R/H line goes low, a master emulator continues to run.

5) To halt a master emulator, enter #n, where n is its ID number.

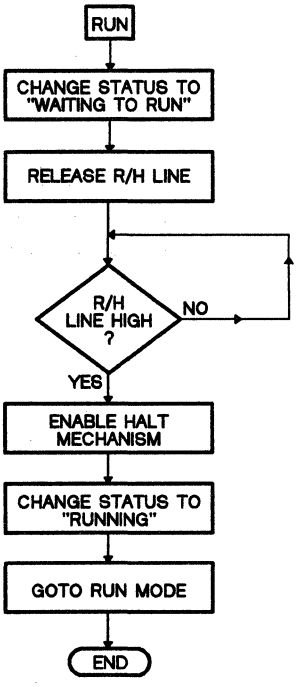


Figure 8-3. Slave Unit after Receiving a RUN, TRUN, OR GRUN Command

1) Execution of the IMD command causes a unit designated as a slave to drive the R/H line low.

2) Upon receiving a RUN command, a slave emulator releases the line and goes into the WAITING TO RUN mode.

3) When the R/H line goes high, a slave emulator begins to RUN.

4) SLAVE units enable the Halt mechanism at this point so that they can force the line to go LOW when execution halts.

5) If the R/H line goes low, all slave emulators halt.

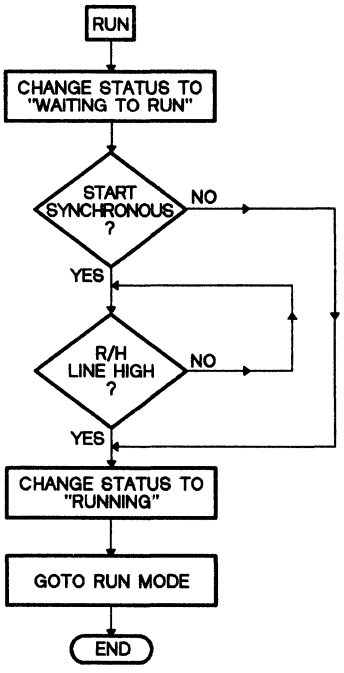


Figure 8-4. Independent Unit after Receiving RUN/TRUN/GRUN Command

1) Execution of the IMD command causes a unit designated as an independent to have no control over the R/H line; however, it does monitor its logic level.

2) Upon receiving a RUN command, an independent emulator that is not configured as start-synchronous begins running immediately.

3) If configured as start-synchronous, an independent emulator goes into the WAITING TO RUN mode.

4) If the emulator is configured as START SYNCHRONOUS, it waits for the Run/Halt line to go high, then goes into the RUNNING mode.

5) An independent emulator does not halt if the Run/Halt line goes low. It can be halted by entering the #n command, where n is the emulator's ID number.

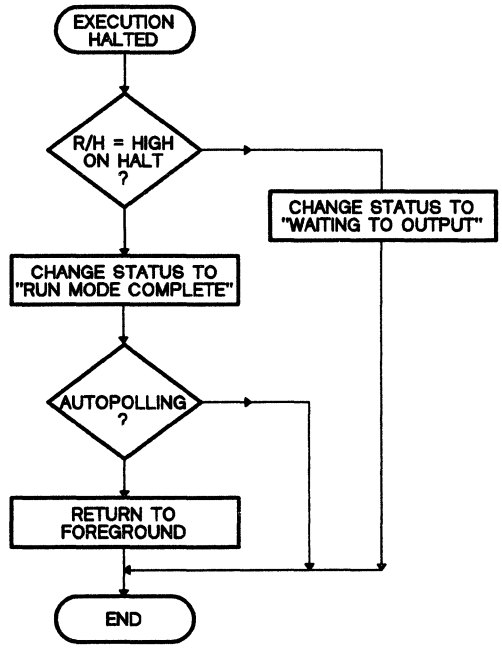


Figure 8-5. Emulator Status after Execution Halts

1) When an emulator stops for any reason, slaves and masters always drive the Run/Halt line low.

2) The slave or master emulator that stops first (R/H line still high) changes its operating status to RUN MODE COMPLETE and checks for Autopolling.

3) If Autopolling enabled, that emulator goes into the foreground and displays its status. If disabled, the emulator displays its status when and if called into the foreground.

4) Emulators that stop after the line goes low change their status to WAITING TO OUTPUT and display their output when they return to the foreground.

5) When such an emulator re-enters the foreground, it displays status information and the READY FOR INPUT question-mark prompt.

8.7 Multiprocessing Example

This example demonstrates a multiprocessing application involving three XDS units.

8.7.1 Multiprocessing Connections
8.7.1.1 Cabling

The cable shown in Figure 8-6 connects from Port D of the first unit to Port A of the second, from Port D of the second unit to Port A of the third, etc. and from Port D of the next-to-last unit to Port A of the last unit.

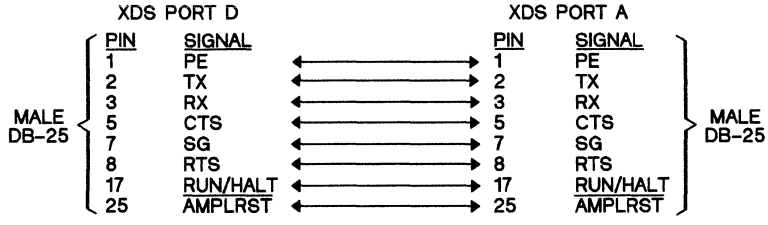


Figure 8-6. Multiprocessing Connecting Cable

Figure 8-7 shows three units connected together.

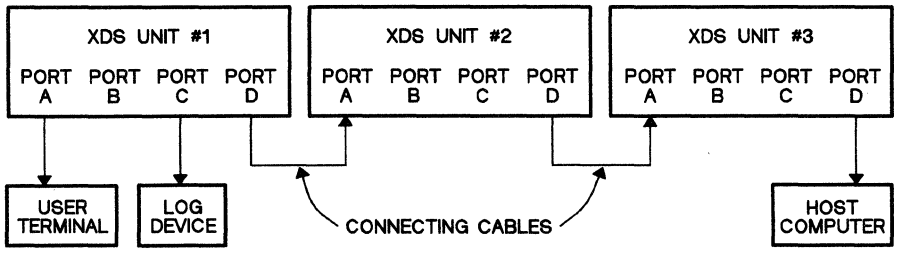


Figure 8-7. XDS Multiprocessing Connections

The terminal and optional logging device connect only to the first unit. The host computer, if used, connects only to the last unit. Units are numbered from 1 to n in the order of physical connection; that is, the unit connected to the terminal is unit #1, the unit connected to Port D of that unit is unit #2, etc.

8.7.1.2 XDS/22 Setup, First Unit in Chain

For the first unit in the multiprocessing chain, set switches S3 and S4 on the Memory Expansion/Communications card according to what terminal you have, and set switches S1 and S2 as shown in Table 8-2 and Figure 8-8. For Global reset (resetting any emulator in the chain resets them all, Section 8.5), connect jumper from point E9 to point E10 (Figure 9-12). For Local reset (resetting any emulator affects *only* that emulator), connect from point E10 to point E11.

Table 8-2. Multiprocessing Switch Settings, First Unit in Chain

SWITCH POSITION	DIP SWITCH			
	S4	S3	S2	S1
1	†	†	ON	OFF
2	†	†	OFF	ON
3	†	†	OFF	OFF
4	†	†	ON	ON
5	†	†	OFF	ON
6	†	†	OFF	ON
7	†	†	ON	OFF.

†Depends on system terminal. Refer to Section 7.1 for typical connection.

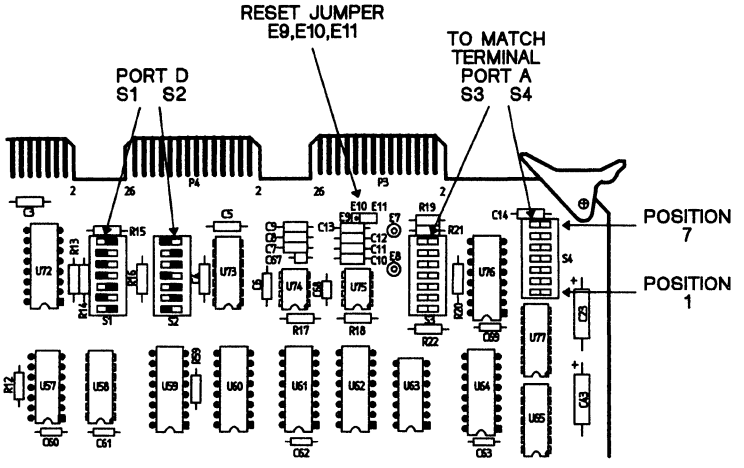


Figure 8-8. Communications Switch Settings, First XDS/22 Unit in Multiprocessing Configuration

8.7.1.3 XDS/22 Setup, Last Unit in Chain

For the last unit in the multiprocessing chain, set switches S1 and S2 on the Memory Expansion/Communications card according to what computer or other device you plan to connect to Port D, and set switches S3 and S4 as shown in Table 8-3 and Figure 8-9. In this case, the setting of S1-8 does not matter.

Table 8-3. Multiprocessing Switch Settings, Last Unit in Chain

SWITCH	DIP SWITCH			
POSITION	S4	S3	S2	S1
1	OFF	OFF	†	†
2	ON	OFF	†	†
3	OFF	OFF	†	†
4	OFF	ON	†	†
5	ON	ON	†	†
6	ON	OFF	†	†
7	OFF	ON	†	†

†Depends on what you have attached to Port D. Refer to various computer and modem descriptions in Section 7 for examples.

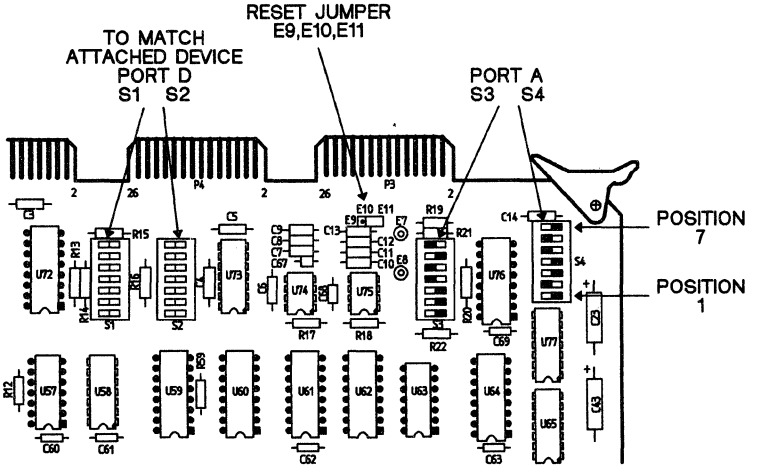


Figure 8-9. Communications Switch Settings, Last XDS/22 Unit in Multiprocessing Configuration

8.7.1.4 XDS/22 Setup, Remaining Units in Chain

For the remaining unit in the multiprocessing chain, set switches S1 through S4 according to Table 8-4 and Figure 8-10. For Global reset, refer to Section 8.5.

Table 8-4. Multiprocessing Switch Settings, Remaining Unit(s) in Chain

SWITCH	DIP SWITCH			
POSITION	S4	S3	S2	S1
1	OFF	OFF	ON	OFF
2	ON	OFF	OFF	ON
3	OFF	OFF	OFF	OFF
4	OFF	ON	ON	ON
5	ON	ON	OFF	ON
6	ON	OFF	OFF	ON
7	OFF	ON	ON	OFF

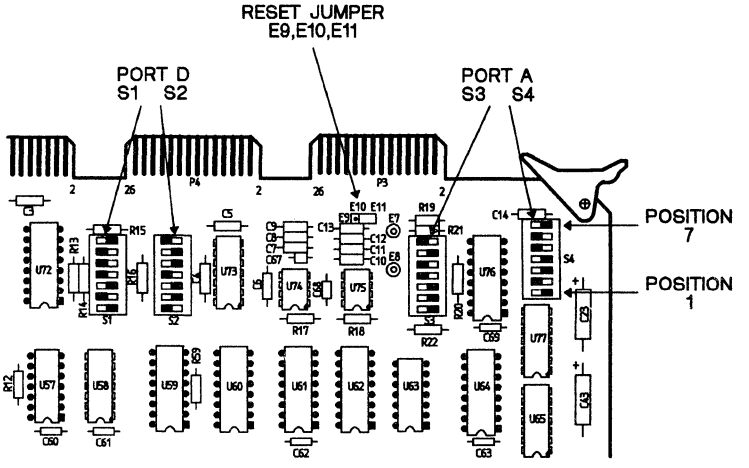


Figure 8-10. Communications Switch Settings, Remaining XDS/22 Units in Multiprocessing Configuration

8.7.2 Running in MP Mode

This description assumes a chain of three emulators.

Begin by initializing the Multiprocessing mode:

```
DISPLAY                                                    ENTER
?                                                          IMP<CR>
IMP<CR>
 AUTOPOLL [0=NO, 1=YES] = NO                               <CR>
 LAST EMU = 3
     TMS320C2x     XDS     VERSION x.x.x

?                                                          IMD<CR>
IMD
 PROCESSOR MODE [0=IND, 1=MASTER, 2=SLAVE] = IND           1<CR>
 START SYNCHRONOUS [0=NO, 1=YES]           = NO            1<CR>

?                                                          ID<CR>
#1
 TMS320C2x      XDS      VERSION x.x.x

?                                                          #2<CR>

?                                                          IMD<CR>
IMD<CR>
 PROCESSOR MODE [0=IND, 1=MASTER, 2=SLAVE] = IND           <CR>
 START SYNCHRONOUS [0=NO, 1=YES]           = NO            <CR>

?                                                          #3<CR>

?                                                          IMD<CR>
IMD<CR>
 PROCESSOR MODE [0=IND, 1=MASTER 2,SLAVE] = IND            2<CR>
 START SYNCHRONOUS [0=NO, 1=YES]          = YES            <CR>

?                                                          RUN<CR>
```

You first defined emulator #1 as a master with synchronous start, #2 as an independent without synchronous start, and #3 as a slave, then you ordered #3 to run. To see the current status of each unit, do the following:

```
DISPLAY                                                    ENTER
BACKGROUND

??                                                         /1<CR>
/1
READY FOR INPUT
BACKGROUND:

??                                                         /2<CR>
/2
READY FOR INPUT
BACKGROUND

??                                                         /3<CR>
/3
WAITING TO RUN
BACKGROUND
```

Remember that the RUN command to unit #3 didn't start it because it is a slave unit, but the command did place it in the background.

Now call unit #2 to the foreground and give it a RUN command.

DISPLAY	ENTER
??	#2<CR>
#2	
READY FOR INPUT	RUN <CR>
?	
RUN	
?BACKGROUND	

Now check the status again.

DISPLAY	ENTER ?? /1<CR>
/1	
READY FOR INPUT	
BACKGROUND	
??	/2<CR>
/2	
EMULATOR RUNNING	
BACKGROUND	
??	/3<CR>
/3	
WAITING TO RUN	
BACKGROUND	
??	#1<CR>
?	RUN<CR>
RUN	
BACKGROUND	
??	/1<CR>
/1	
RUNNING	
BACKGROUND	
??	/2<CR>
/2	
RUNNING	
BACKGROUND	
??	/3<CR>
/3	
RUNNING	
BACKGROUND	
??	#2<CR>

Unit #2 program execution halts after the pressing of any key on the terminal keyboard. Since it is in the foreground, the terminal displays the program-counter, status-register, and stack-pointer values. Since unit #2 is independent, slave unit #3 continues to run.

DISPLAY	ENTER
KEY	
PC=0000 ST0=xxxx ST1=xxxx	
?	BGND<CR>
?	
??	/1<CR>
/1	
RUNNING	
BACKGROUND	

Now place unit #2 in the background with the BGND command, check that unit #3 is still running, then place unit #3 in the foreground and stop it by pressing any key.

DISPLAY	ENTER
?? /2 READY FOR INPUT BACKGROUND	/2<CR>
?? /3 RUNNING BACKGROUND	/3<CR>
?? KEY PC=0000 ST0=3EFC ST1=xxxx	#3<CR>

Now place unit #3 back in the background and check status again.

DISPLAY	ENTER
? BACKGROUND	BGND<CR>
?? /1 RUNNING BACKGROUND	/1<CR>
?? /2 READY FOR INPUT BACKGROUND	/2<CR>
??/3 /3 READY FOR INPUT BACKGROUND	/3<CR>
?? KEY PC=0000 ST0=xxxx ST1=xxxx	#1<CR>

Now enter a GRUN (Group Run) command and notice that it starts unit #1, the master, and unit #3, the slave, since both are set for synchronous start. It does not start unit #2 since it is independent and not set for synchronous start.

DISPLAY	ENTER
? BACKGROUND	GRUN<CR>
?? /1 RUNNING BACKGROUND	/1<CR>
?? /2 READY FOR INPUT BACKGROUND	/2<CR>
?? /3 RUNNING BACKGROUND	/3<CR>

Units #1 and #3 started, Unit #2 didn't. Now stop units #1 and #3 by calling them individually into the foreground.

DISPLAY	ENTER
?? KEY PC=0000 ST0=xxxx ST1=xxxx	#1<CR>
? KEY PC=0000 ST0=xxxx ST1=xxxx	#3<CR>

Now start all three units together with a TRUN (Total Run) command.

DISPLAY	ENTER
? BACKGROUND	TRUN<CR>
?? /1 RUNNING BACKGROUND	/1<CR>
?? /2 RUNNING BACKGROUND	/2<CR>
?? /3 RUNNING BACKGROUND	/3<CR>
??	

Now halt units #1 and #3 with a GHALT (Group Halt) command.

DISPLAY	ENTER
? BACKGROUND	GHALT<CR>
?? /1 RUN MODE COMPLETE BACKGROUND	/1<CR>
?? /2 RUNNING BACKGROUND	/2<CR>
?? /3 RUN MODE COMPLETE BACKGROUND	/3<CR>
??	

The GHALT command stops the group emulators but doesn't affect the independent emulator.

Units #1 and #3 started, Unit #2 didn't. Now stop units #1 and #3 by calling them individually into the foreground.

DISPLAY	ENTER
??	#1<CR>
KEY	
PC=0000 ST=00 SP=00	
?	#3<CR>
KEY	
PC=0000 ST=00 SP=00	

Now start all three units together with a TRUN (Total Run) command.

DISPLAY	ENTER
?	TRUN<CR>
BACKGROUND	
??	/1<CR>
/1	
RUNNING	
BACKGROUND	
??	/2<CR>
/2	
RUNNING	
BACKGROUND	
??	/3<CR>
/3	
RUNNING	
BACKGROUND	
??	

Now halt units #1 and #3 with a GHALT (Group Halt) command.

DISPLAY	ENTER
?	GHALT<CR>
BACKGROUND	
??	/1<CR>
/1	
RUN MODE COMPLETE	
BACKGROUND	
??	/2<CR>
/2	
RUNNING	
BACKGROUND	
??	/3<CR>
/3	
RUN MODE COMPLETE	
BACKGROUND	
??	

The GHALT command stops the group emulators but doesn't affect the independent emulator.

9. Hardware Description

This section describes XDS/22 hardware and XDS plug-in cards.

Topics in this Section include:

Warning:

Since many of the procedures described in this section require a working knowledge of computer and electronic equipment, Texas Instruments recommends that all installations and procedures be made by a trained technician.

Note:

An XDS/22 contains no user-serviceable parts except the line fuse and service on that item is limited (Section 9.6.6). Also notice (Page 9-49) that unauthorized repair *may* void your warranty.

9.1 XDS/22 Unit

This section describes the XDS/22 unit. Topics include:

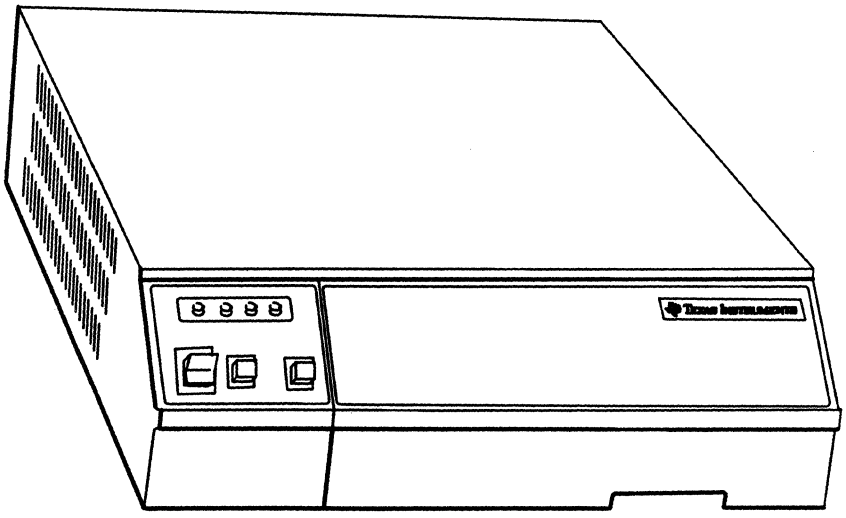


Figure 9-1. XDS/22 Cabinet

9.1.1 Physical Description

An XDS/22 (Figure 9-1) cabinet mounts an operator panel containing switches and indicator lights; a rear panel containing port connectors, line-cord connector and fuse holder, and an exhaust fan; a top cover; and a front card-cage cover.

The cabinet contains a power supply, seven-slot card cage with backplane connector, and chassis cable-mounting bracket.

9.1.1.1 Operator Panel

Figure 9-2 shows the operator-panel controls and indicators.

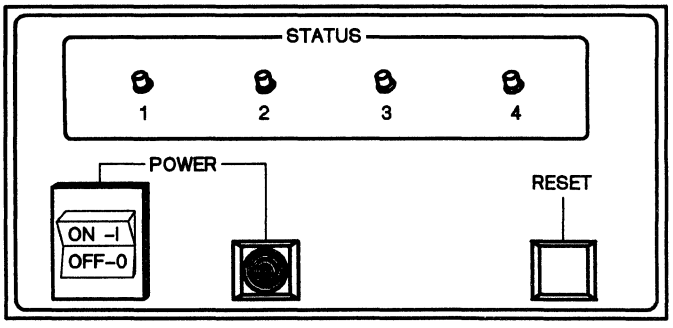


Figure 9-2. XDS/22 Operator Panel

Pressing the ON-1 segment of the POWER switch applies line voltage to the XDS/22. With line voltage applied, the POWER indicator lights and the internal fan runs (it takes a few seconds to work up to speed). With an emulator card installed, Status Light #4 should also light if the target-system cable isn't connected.

Pressing the OFF-0 segment removes line voltage from the XDS/22.

Pressing RESET reinitializes the emulator hardware, but not the monitor program. Expect a slight delay while the unit reinitializes.

> **Note:**
>
> Pressing RESET does *not* clear the emulator memory. You *must* use the RESTART command (Page 4-96)

The remaining Status lights show emulator-card operating conditions. Refer to Section 9.4 for a description of exactly what each indicates.

9.1.1.2 Rear Panel

The rear panel (Figure 9-3) houses the four female DB-25 EIA port connectors, line cord/fuse connector, and exhaust fan.

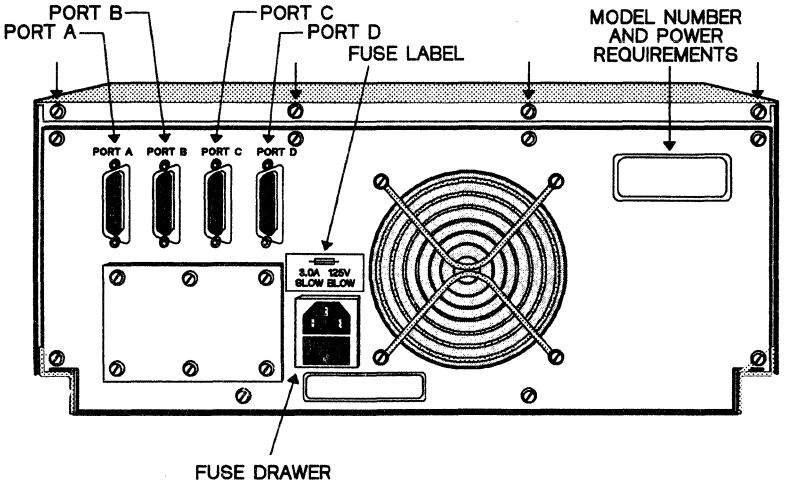


Figure 9-3. Rear of XDS/22 Unit

Table 9-6 on Page 9-21 shows the port connector pinout.

The line cord/fuse holder requires that you pull the line cord before replacing the fuse. The fuse holder accepts 3AG fuses or 5mm x 20mm fuses. For International applications (220V, 50 Hz), the holder also includes a special line filter (TI Part No. 22221618-0001) to meet VDE[24] requirements.

The exhaust fan pulls cooling air through the ventilation slots and air filters in the top cover.

It should not be necessary to remove the rear panel for normal maintenance and repair.

9.1.1.3 Top Cover

The top cover protects interior components, provides structural integrity, and mounts the unit air filters.

---

[24] VDE (Vereinigung Deutscher Electrizitaetswerke) sets electrical standards for equipment sold in the International market, just as Underwriters Laboratory (UL) sets standards for the United States.

9.1.1.4 Card-Cage Cover

The card-cage cover protects the plug-in cards mechanically, keeps externally-generated electromagnetic radiation (EMI) out and internally-generated EMI in, and directs cooling air over the cards.

> **Caution:**
>
> **Never operate an XDS/22 without the card-cage cover in place.**

9.1.1.5 Power Supply

The vendor-supplied power supply is a high-efficiency switching-type component that generates ±12 V and +5 V with sufficient capacity to handle any emulator configuration.

9.1.1.6 Card Cage

The various emulator cards plug into the card-cage backplane connector in the order shown in Table 9-1. An XDS/22 must include a Communications card and at least one emulator card.

Table 9-1. XDS/22 Card-Cage Slot Assignments

SLOT	CIRCUIT CARD
7	Not used
6	Not used
5	Not used
4	Memory Expansion/Communications card
3	Not used
2	TMS320C2x Emulator card
1	Breakpoint/Trace/Timing card

9.1.1.7 Backplane connector

The backplane connector (Figure 9-4) routes signals between cards; mounts connectors for power input (P1) and status indicators and switches (P2); and mounts logic to drive the status indicators (U1,U2), terminating resistors (U3), and reset logic (U4). U4 and associated components generate signal $\overline{\text{SPOR}}$ (System Power-On Reset) when you turn the XDS/22 on, or when you press RESET on the Operator Panel.

The Breakpoint/Trace/Timing card plugs into J1 and J2, the emulator card plugs into J3 and J4, and the Memory Expansion/Communications card plugs into J7 and J8.

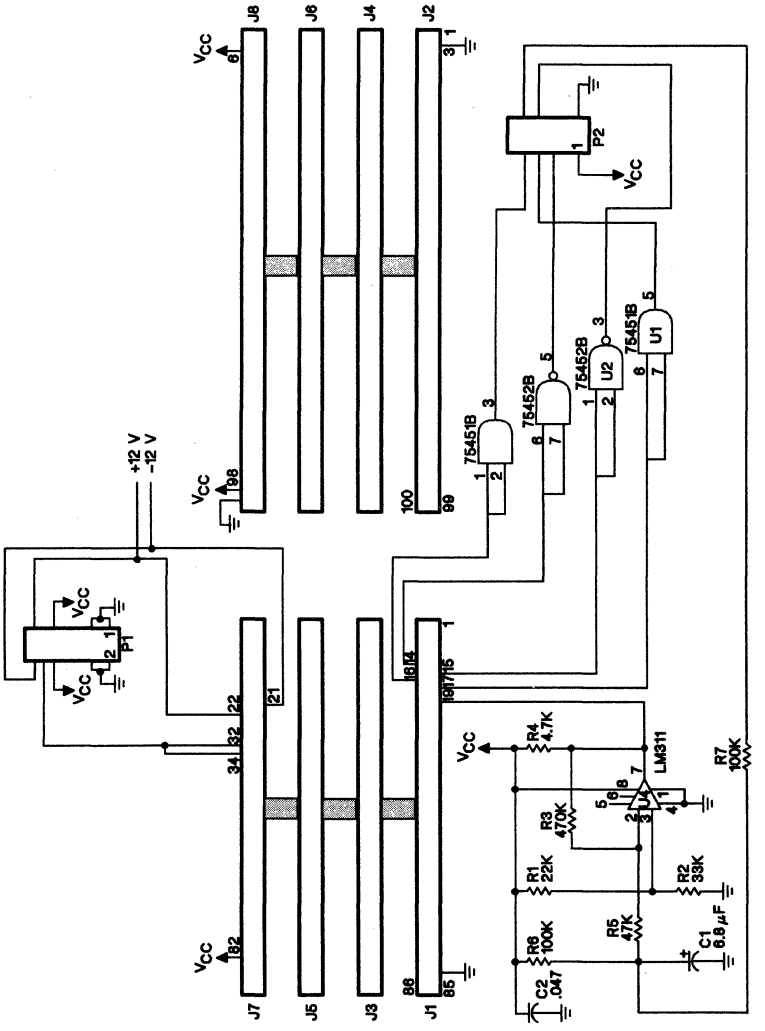


Figure 9-4. XDS/22 Card-Cage Backplane Connector Diagram

9.1.1.8 Chassis Cable Mounting

The chassis cable mounting (Figure 9-5) anchors the various strain-relief brackets used by XDS system cables. The bracket has two sets of mounting holes: one horizontal set located over the chassis edge and one set located at 45° on the extended tabs.

This application uses the horizontal set with the adapter for the Breakpoint/Trace card extended-address cable or with the strain-relief attached to the Breakpoint/Trace/Timing logic-analyzer cable.[25]

This application does not use the 45° set.

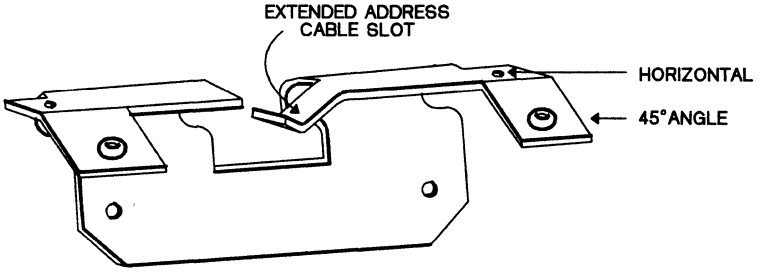


Figure 9-5. Chassis Cable Mounting

9.1.1.9 XDS/22 Chassis Block Diagram

[25] Firmware revision 1.2.0 or higher

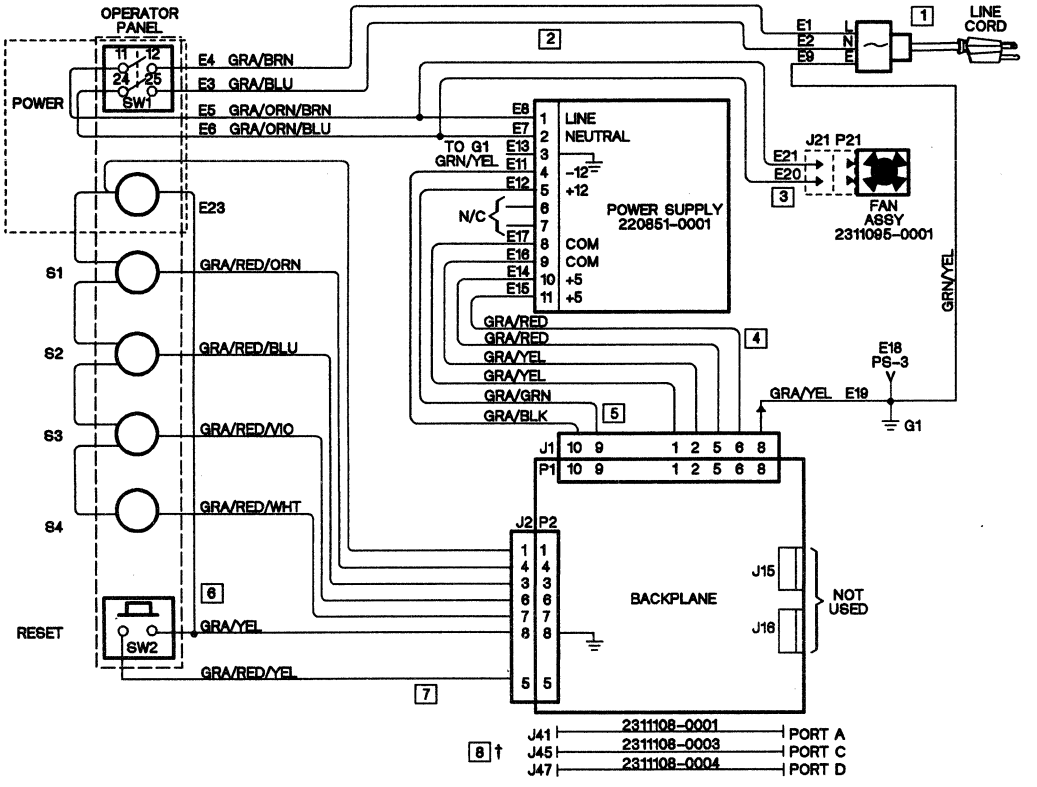


Figure 9-6. XDS/22 Chassis Block Diagram

9.1.2 Environmental Requirements

An XDS/22 is intended for use in an air-conditioned laboratory environment (Table 9-2). Do not expose the system to temperature changes greater than 12°F/hour (6.6°C/hour). Low relative humidity promotes static buildup which may damage the XDS/22 unit.

Table 9-2. XDS/22 Temperature and Humidity Ranges

OPERATION		STORAGE	
TEMPERATURE	HUMIDITY	TEMPERATURE	HUMIDITY
60 - 90°F 16 - 32°C	30 - 80%	-40 - +185°F -40 - +85°C	5 - 90%

Keep the XDS/22 unit at least 5 inches (125 millimeters) from other equipment or walls to allow sufficient airflow.

Never operate the XDS/22 unit without the card-cage cover installed. Such operation may cause it to overheat and may also cause electromagnetic interference with target system.

9.1.3 AC Input Requirements

Connect the XDS/22 unit only to a power source that matches the voltage and frequency stamped on the model plate on the rear panel (Figure 9-7). Table 9-3 shows power details.

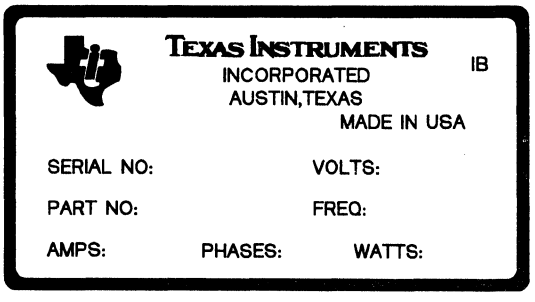


Figure 9-7. XDS/22 Model Plate

For best results, connect both XDS and terminal to a branch circuit that is free of copiers, fluorescent lights, pencil sharpeners, power tools, typewriters, etc., since voltage spikes from such appliances could affect operation.

If possible, connect terminal and XDS/22 to the same branch circuit.

Table 9-3. XDS/22 Voltage Requirements

RATED VOLTAGE	RATED FREQUENCY	RATED CURRENT	NEMA WALL RECEPTACLE
100 V_{ac}	50/60	3 Amps	5-15R
115 V_{ac}	60	3 Amps	5-15R
220 V_{ac}	50	2 Amps	Schuko 1050
240 V_{ac}	50	2 Amps	BSI 1363

9.1.4 XDS/22 DC Output

Table 9-4 shows the maximum power-supply output. Refer to individual card descriptions for their power requirements.

Table 9-4. XDS/22 DC Power Output

VOLTAGE	AMPS
5	20.0
12	4.0
-12	2.0

9.1.5 Component Removal

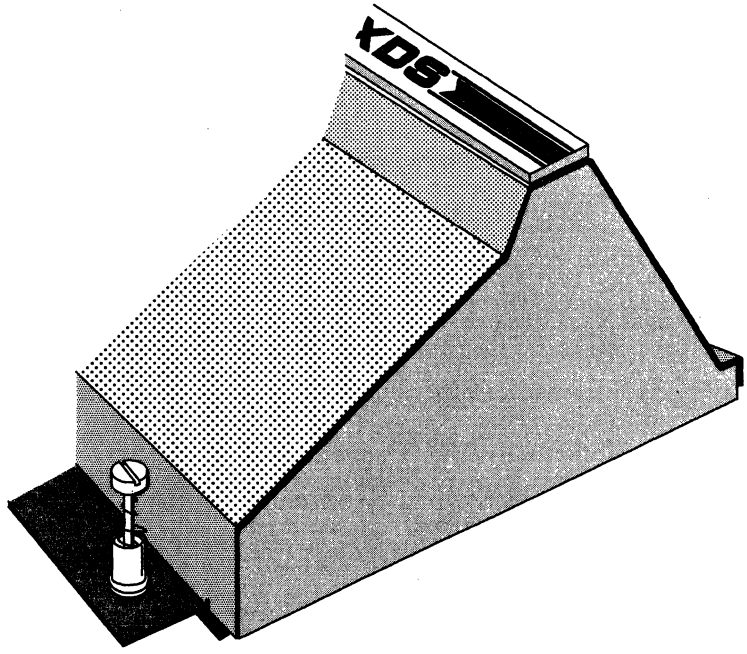


Figure 9-8. Card-Cage Cover, Right Side

9.1.5.1 Card-Cage Cover Removal

To remove the card-cage cover, follow this procedure:

1) Turn power OFF.
2) Loosen two captive thumb screws on XDS card-cage cover (Figure 9-8).
3) Apply slight upward pressure to cover. Pull bottom of cover up and away from chassis until bottom clears target-cable clamp hardware.
4) Let cover drop enough to clear top lip of chassis.
5) Be careful not to damage target-system cable when removing cover entirely.

9.1.5.2 Card Removal

1) General

1) Turn XDS and target system power OFF.
2) Remove card-cage cover (Section 9.1.1.4).
3) Note configuration list inside cover. Card numbers from 1 to 7 (bottom to top) appear on right side of inner chassis.
4) It is best to remove the emulator card before removing the Breakpoint/Trace/Timing card.[26]
5) Place card in protective bag and store in protected area.
6) If removal permanent, erase card name from list on card-cage cover.

2) Memory Expansion/Communications Card

1) Remove J41, J45, and J47 from card connectors.
2) Pull out on inside edge of ejector tabs to unseat card.
3) Carefully remove card from card cage.

3) Emulator Card

1) Disconnect target cable from target system
2) Install protective cover or non-conductive foam on target-cable connector
3) Loosen thumbscrews holding target-system cable strain-relief to adapter, Breakpoint/Trace/Timing-card cable bracket, or chassis cable bracket.
4) Pull out on inside edge of Emulator-card ejector tabs to unseat card.
5) Keep target cable free of obstructions while carefully pulling card from card cage.

4) Breakpoint/Trace/Timing Card

1) Remove Emulator card as described in Section 9.1.5.2.
2) If logic-analyzer cable installed, loosen thumbscrews holding its cable clamp to chassis cable bracket.
3) If applicable, unplug extended-address cable and its ground connector.

---

[26] Footnote deleted.

4) Pull out on inside edge of Breakpoint/Trace/Timing-card ejector tabs to unseat card.

5) Carefully remove card from card cage.

6) As soon as possible, reinstall card or install BTT card or install adapter and reconnect Emulator target-cable strain relief.

9.1.5.3 Fuse Replacement

Because fuse replacement should also involve finding out why the fuse failed, refer to Section 9.6.6 for the procedure.

9.1.6 Component Installation

9.1.6.1 Card-Cage Cover

To replace the card-cage cover, follow this procedure:

1) Place top of cover under lip at top of XDS/22 unit.
2) Push cover bottom toward unit, ensuring that the two thumb screws start properly.
3) Tighten screws.

9.1.6.2 Card Installation

1) General

1) Install each card in its appropriate slot, component side up. Press firmly on card edge or on ejector tabs to seat card in backplane connector. You may have to move a card up and down in its slot to mate it with the backplane connector.

2) For cards installed in slots 1 or 2, follow this procedure.

   a) Move card-ejector tabs parallel to side of card.

   b) Place card in slot and push it into card cage until it reaches backplane card sockets, then back it out slightly until you can rotate outside edge of each ejector tab into small square hole on side of card cage (Figure 9-9).

   c) Press card into backplane card connectors by pressing on card-ejector tabs.

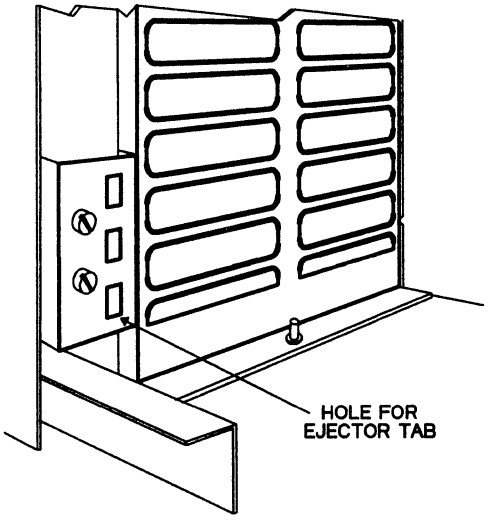


Figure 9-9. XDS/22 Card Cage Ejector Tab Holes

3) Check for proper security of the target-cable strain relief assembly and Breakpoint/Trace/Timing Card extended-address cable.

4) Replace cabinet cover as described on Page 9-12.

2) Memory Expansion/Communications Card Installation

1) Configure the card (Section 9.2.2) before installing it in chassis slot #4, component side up. Press firmly on card edge or card-ejector tabs.

2) Connect J41, J45, and J47 (Figure 9-10). Note that colored stripe indicates pin 1.

> **Note:**
>
> Some XDS units may have a fourth cable, labeled J43. Do not connect this cable.

3) List card on chassis-configuration label inside card-cage cover.

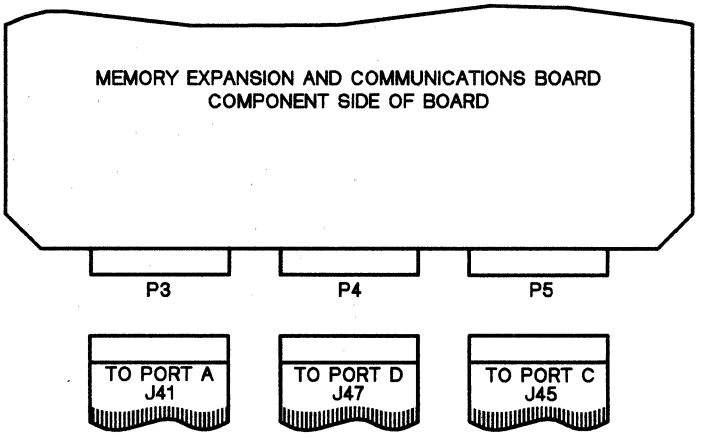


Figure 9-10. Communications Cable Connections

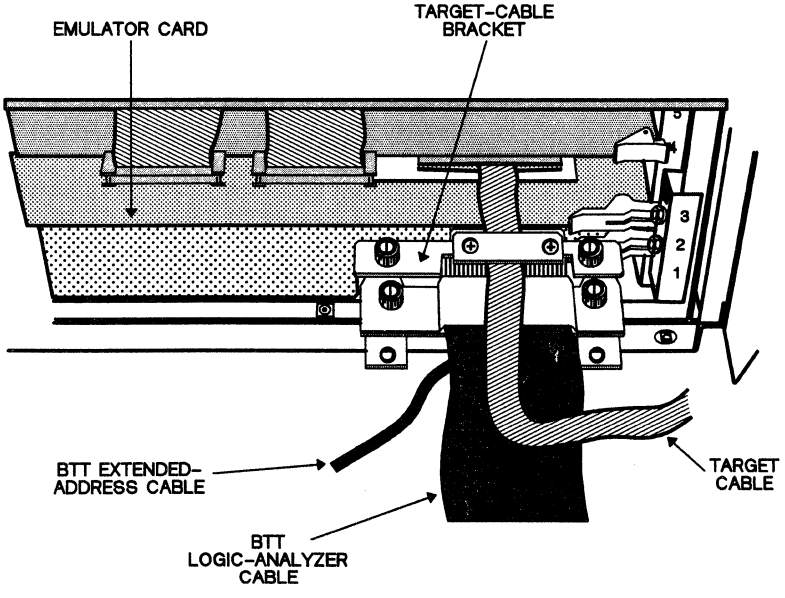


Figure 9-11. Emulator and Breakpoint/Trace/Timing Card Installation

3) Breakpoint/Trace/Timing Card Installation

1) If using a logic analyzer, connect cable J4 (stamped LOGIC POD) to BTT P4 and J5 to BTT P5, before installing the card. Be sure to align J4 with P4 and J5 with P5 and press connectors down firmly!

2) If using extended-address cable, plug its ground push-on connector into the XDS chassis clip.

3) Move card-ejector tabs on Breakpoint/Trace/Timing card parallel to side of card.

4) Place Breakpoint/Trace/Timing card in slot #1 and push it into card cage until it reaches backplane card sockets, then back it out slightly until you can rotate outside edge of each ejector tab into small square hole on side of card cage.

5) Press card into backplane card connectors by pressing on card-ejector tabs.

6) Plug extended-address cable into Breakpoint/Trace/Timing-card connector P3 after installation. If logic-analyzer also used, run extended-address cable underneath logic-analyzer cable.

7) Secure logic-analyzer cable clamp on chassis cable mount bracket with thumbscrews. If logic-analyzer cable not used, mount strain-relief adapter on chassis bracket mount by tightening the two thumb screws.

4) Emulator Card Installation

1) Install target cable on emulator card as described in Section 9.4.5.

2) Install card in XDS chassis slot #2, component side up. Be sure ejector tabs parallel to card sides.

3) Place card in slot and push it into card cage until it reaches backplane card sockets, then back it out slightly until you can rotate outside edge of each ejector tab into small square hole on side of card cage (Figure 9-9). Be sure card back does not scrape Breakpoint/Trace/Timing logic-analyzer cable.

4) Press card into backplane card connectors by pressing on card-ejector tabs.

5) With emulator target cable on top of logic-analyzer cable, attach target-cable strain-relief bracket to strain-relief adapter or logic-analyzer cable strain-relief bracket with two thumbscrews into the hexagonal standoffs (Figure 9-11).

6) List Breakpoint/Trace/Timing and emulator cards on the chassis-configuration label located on the card-cage cover.

Pagination

9.2 Memory Expansion/Communications Card

This section contains the following topics:

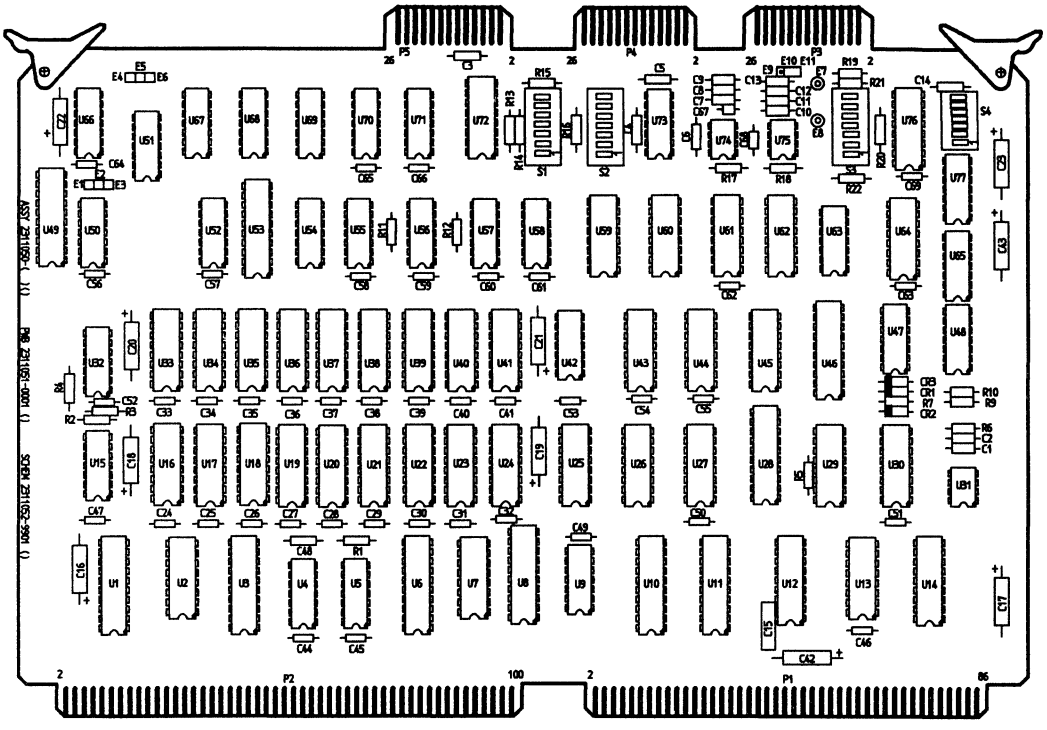


Figure 9-12. Memory Expansion/Communications Card

9.2.1 Card Description

The Memory Expansion/Communications card (Figure 9-12) contains communications logic for the XDS EIA ports. The -0001 configuration used in this application also contains additional memory.

9.2.2 Preparation for Installation

1) Before installing card, set S3 and S4 to configure Port A (terminal, Section 7.1) and, if necessary, set S1 and S2 to configure Port D (Section 7.6) for your specific application.

2) For multiprocessing mode, configure Port A and Port D depending on the XDS unit's position in the chain (Section 8.7.1). Also set Communications card jumper from E9 - E10 for Global reset (factory setting) or from E10 - E11 for Local reset.

3) The cable (Section 5.7.2.2) configures Port C.

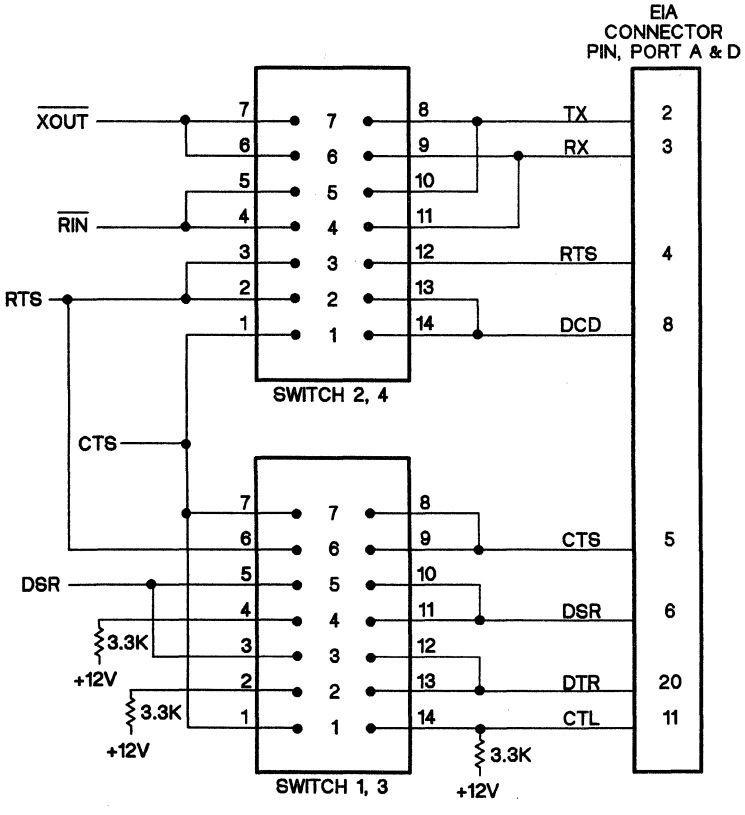


Figure 9-13. Switch Options for EIA Port A

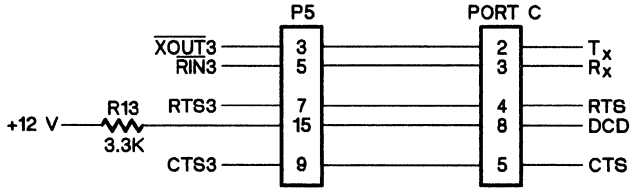


Figure 9-14. Port C Connection

Table 9-5 shows the cable connections between the Memory Expansion/Communications card and Ports A, C, and D on the XDS/22. Cable J41 connects from P3 on the Memory Expansion/Communications card to Port A. Cable J45 connects from P5[27] to Port C. Cable J47 connects from P4 to Port D.

You can select the signal source for XOUT, etc. by means of Memory Expansion/Communications card switches S3 and S4 for Port A or S1 and S2 for Port D (Section 9.2.3). You can also use the IPORT command.

You can only configure Port C by changing its cable connections.

[27] Only PG, TX, RX, RTS, CTS, and DCD are connected on P5.

Table 9-5. Cable, Memory Expansion/Communications Card Connector to EIA Port

SIGNAL NAME	CARD PIN	EIA CONNECTOR PIN
PG	1	1
TX	3	2
RX	5	3
RTS	7	4
CTS	9	5
DSR	11	6
SG	13	7
SG	13	7
DCD	15	8
†	17	9
†	19	10
‡	21	11
SCF	23	12
SCB	25	13
SBA	2	14
DB	4	15
SBB	6	16
DD	8	17
‡	10	18
SCA	12	19
DTR†	14	20
CG	16	21
CE	18	22
CH	20	23
CI	22	24
DA	24	25

†Reserved
‡Unassigned.

Table 9-6 shows the connections for Ports A - D. Remember that not all pins are connected on Port C.

Table 9-6. EIA Connector Configuration

PIN	SIGNAL	RS-232C NAME	DESCRIPTION
1	PE	AA	Protective Ground. Chassis ground between XDS and external device.
2	TX †	BA	Transmit Data. Data to be transmitted from DTE device to DCE device.
3	RX †	BB	Receive Data. Data received from DTE device by DCE device.
4	RTS †	CA	Request to Send. DTE sends to force DCE device to be ready to receive data.
5	CTS †	CB	Clear to Send. Acknowledges RTS recognized by external device that is ready to accept data from XDS.
6	DSR ‡	CC	Data Set Ready. Affirms port connection to functional DCE device, such as modem.
7	SG	AB	Signal Ground. Signal common between XDS and external device.
8	DCD †	CF	Data Carrier Detect. Affirms presence of carrier and verifies existence of correct operating conditions.
9-10	N/C		
11	CTL ‡	Unassigned	General Control. For use by any device needing to suspend XDS transmission until it can accept more data.
12-16	N/C		
17	RUN/HALT	Unassigned	Active during multiprocessing only.
18-19	N/C		
20	DTR ‡	CD	Data Terminal Ready. Indicates transmitting device in operating condition and ready state.
21-24	N/C		
25	$\overline{\text{AMPLRST}}$	Unassigned	Active during multiprocessing only.

† Signature for Ports A and D depends on switch settings.
‡ Not available for Port C.

9.2.3 XDS Memory Expansion/Communications Card Block Diagram

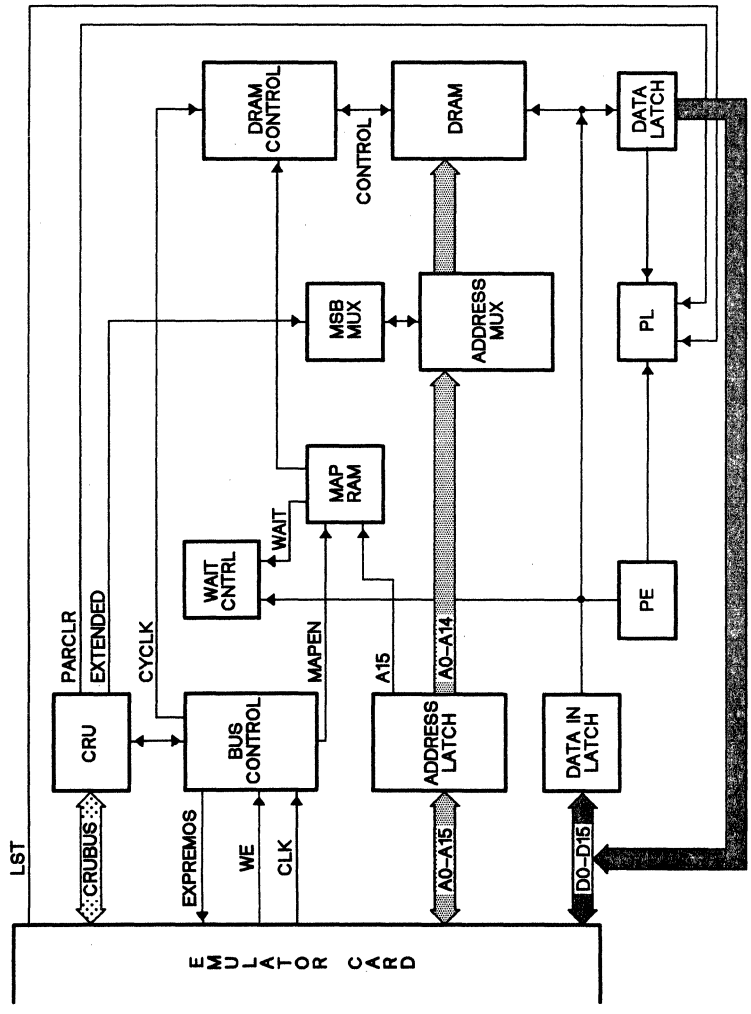


Figure 9-15. XDS Memory Expansion/Communications Card Block Diagram

9.3 Breakpoint/Trace/Timing Card

The Breakpoint/Trace/Timing (BTT) Card, Texas Instruments part number 2243660-0002, gives you advanced breakpoint and tracing operation on up to 32 bits and also adds sequencing and trace time-stamping functions.

Address, data, and qualifier latches on the card capture information from the XDS backplane; this data then passes through a series of lookup RAMs to form 32-bit address/data comparisons and processor-cycle qualifications. Outputs from the compare RAMs control event and delay counters, trace-sample enable, halt logic, sequence control, and timing logic.

The Breakpoint/Trace/Timing card also offers direct connection to a logic-analyzer.

Topics in this section include:

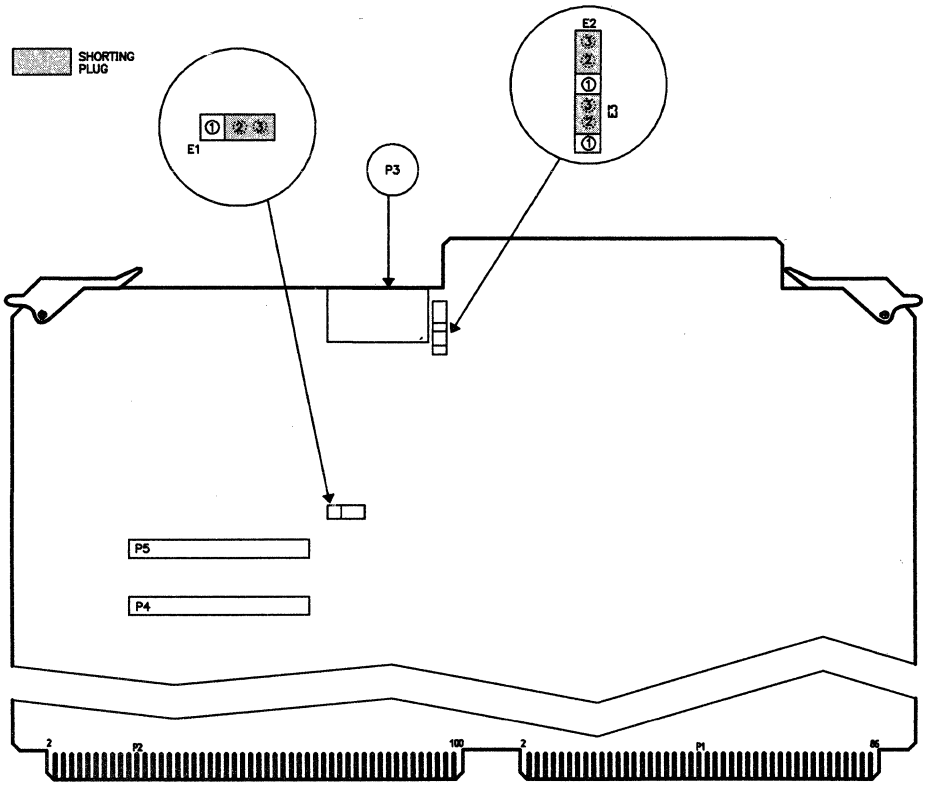


Figure 9-16. Breakpoint/Trace/Timing Card

9.3.1 BTT Card Jumper Connections

For this application, connect jumpers as follows (Figure 9-16):

E1	2-3
E2	2-3
E3	2-3

For E1 (Figure 9-16), pin 1 is to your left. For locations E2 and E3, pin 1 is at the bottom.

9.3.2 Cabling

A Breakpoint/Trace/Timing card can have both a logic-analyzer cable that plugs into P4 and P5 and an extended-address cable that plugs into P3. The logic-analyzer cable is flat, 2 inches (50mm) wide, and has a strain-relief bracket that fastens to the XDS chassis cable mount inside the XDS chassis (Section 9.1.6.2). The extended-address cable is round, about 5/16 of an inch (8mm) in diameter, and fastens under the logic-analyzer strain-relief bracket or adapter bracket in the semicircular groove in the chassis cable mount.

9.3.3 BTT Card Installation

Refer to Section 9.1.6.2.

9.3.4 Extended-Address Probes

Refer to Section 6.9 on Page 6-22.

9.3.5 Breakpoint/Trace/Timing Card Block Diagram

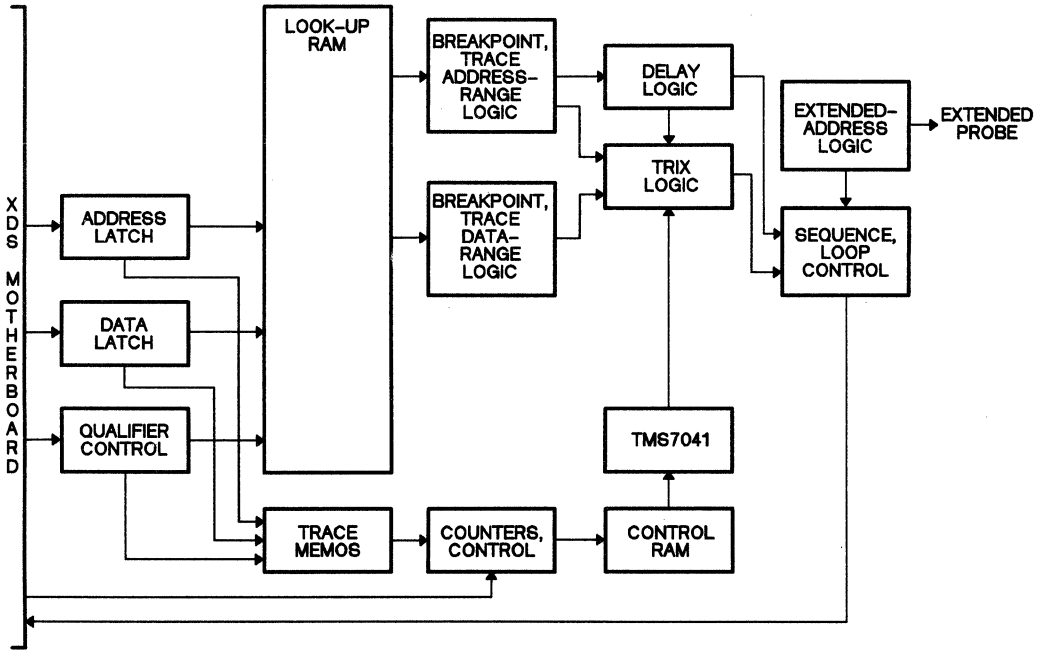


Figure 9-17. Breakpoint/Trace/Timing Card Block Diagram

9.4 TMS320C2x Emulator Card

The TMS320C2x Emulator card (Figure 9-18) provides real-time in-circuit emulation with breakpoints, logic-state trace, and target-system debugging capabilities of both hardware and software systems for the TMS320C2x digital signal processor.

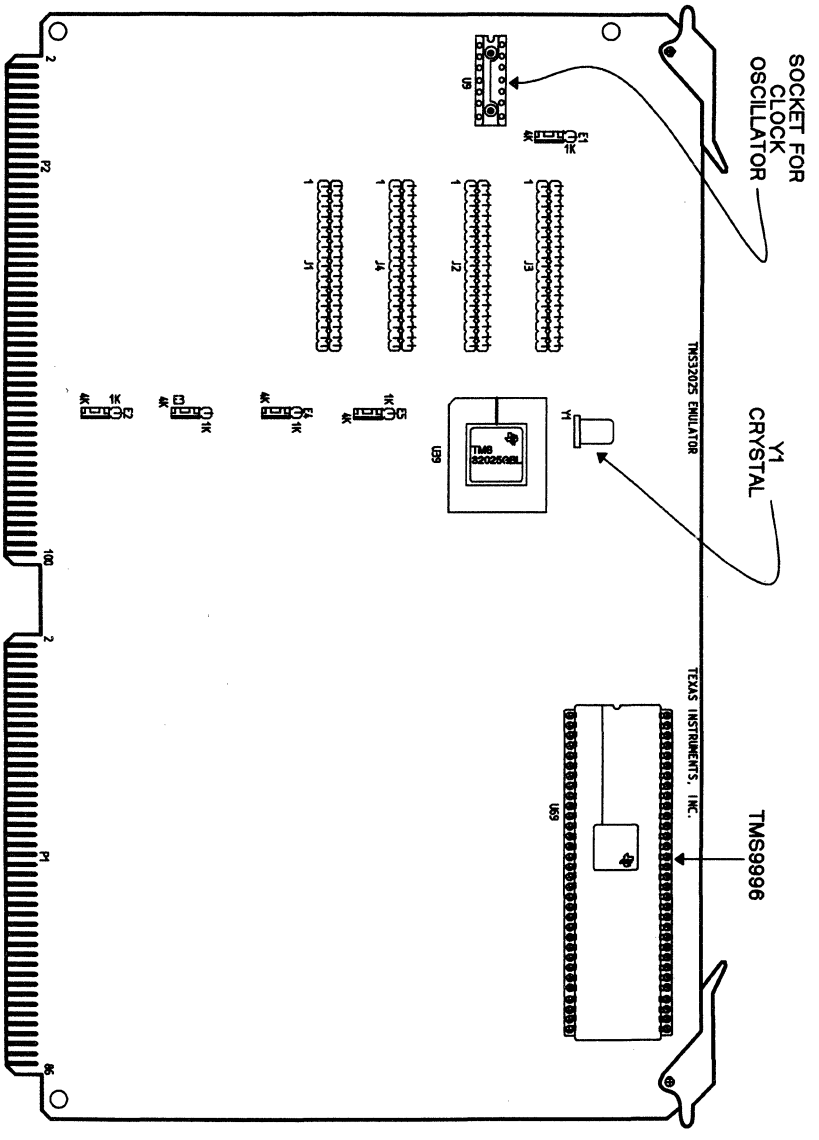


Figure 9-18. TMS320C2x Emulator Card

9.4.1 Card Preparation Before Installation

Before installing the card, you may want to change the random-access memory and, possibly, the clock frequency. You may also have to install the target-connector cable.

9.4.1.1 Random-Access Memory

The factory installs a four-kilobyte by 4-bit RAM in each of locations U31 through U38 (Figure 9-18) and jumpers E1 - E5 in the 4K position. This forms addresses >0 through >FFFF (0 - 4095) in the target-system program-memory space and addresses >400 through >13FF in the data-memory space.

> **Note:**
>
> Because data-memory address decoding is incomplete, enabling the onboard data RAMs also enables addresses >1400 through >1FFF.

You can install other 1K or 4K RAM devices in these sockets (Table 9-7). To enable 1K devices, move each jumper from the plug end marked 4K to the plug end marked 1K.

Table 9-7. Data/Program Memory RAM Options

COMPONENT	MANUFACTURER	RAM SIZE	E1-E5 JUMPER POSITION	MAXIMUM SINGLE-CYCLE ACCESS FREQUENCY
CY7C169-25PC	Cypress Semiconductor	4Kx4	4K	10 MHz
IMS1421S40	INMOS	4Kx4	4K	7.4 MHz
IMS1421S50	"	4Kx4	4K	6.7 MHz
AM2149-35DC	Advanced Micro Devices	1Kx4	1K	8.2 MHz
AM2149-45DC	"	1Kx4	1K	7.5 MHz
AM2149-55DC	"	1Kx4	1K	6.7 MHz

> **Caution: All RAM installed must be the same size and organization.**

> **Note:**
>
> Insert pin 1 of 1K RAM into 20-pin socket pin 2; that is, leave pins 1 and 20 empty.

Table 9-8 shows the address for all possible memory options.

Table 9-8. Target-System RAM Location

PROCESSOR ADDRESS	PROGRAM MEMORY		DATA MEMORY	
	1K OPTION	4K OPTION	1K OPTION	4K OPTION
Block 1: 0- 3FF	X	X	On-chip†	On-chip†
Block 2: 400- 7FF		X	X	X
Block 3: 800- BFF		X		X
Block 4: C00- FFF		X		X
Block 5: 1000-13FF				X
Block 6: 1400-17FF			Block 2	Block 2
Block 7: 1800-1BFF				Block 3
Block 8: 1C00-1FFF				Block 4

†Always internal to the TMS320C2x.

The last three entries under DATA-MEMORY RAM/4K OPTION deserve further comment. The Emulator ignores address bit A12 in the address decode; therefore data-memory locations >0400 - >07FF decode the same as locations >1400 - >17FF, locations >0800 - >0BFF decode the same as locations >1800 - >1BFF, and locations >0C00 - >0FFF decode the same as locations >1C00 - >1FFF. Of course, locations >0000 - >0300 also decode the same as locations >1000 - >13FF, but locations >0000 - >0300 are on the TMS320C2x chip, not in onboard RAM.

This scheme provides contiguous data-memory space without losing the 1K of memory from >0000 to >0300.

9.4.1.2 System Frequency

You can clock your target system from the Emulator-card crystal, the target-system clock source, or from an integrated-circuit crystal oscillator that you can install on the Emulator card at location U9. You select the clock source by means of the INIT command (Page 4-75).

Caution:

Whatever the source, the system frequency must match the TMS320C2x clock specifications listed in the TMS320C2x User's Guide (SPRU0012).

The Emulator-card crystal automatically takes over if you power up the Emulator with the target cable disconnected or if you apply V_{CC} to the target-system socket X2/CLKIN pin. Do NOT unplug the cable with power applied!

Using the target-system clock requires driving the X2/CLKIN pin with that signal as well as selection with the INIT command.

An integrated-circuit crystal oscillator must have output on pin 8, ground on pin 7, +5 V on pin 14, and no connection to pin 1.

9.4.2 Emulator Card Installation

> **Caution:**
>
> **Never remove or install circuit cards with power applied to the XDS unit.**

> **Caution:**
>
> **Always work in a clean, static-free area, following normal static-discharge handling procedures.**

> **Caution:**
>
> **Remove any other emulator cards from the XDS unit before installing the TMS320C2x Emulator card.**

For installation procedure, refer to Section 9.1.6.2.

9.4.3 Major Components

9.4.3.1 Microprocessors and Microcomputers

The TMS320C2x emulator card contains an SE9996 microprocessor, TMS320C2x digital signal processor, and associated components for complete emulation of either device. The SE9996 controls emulation and user interfacing, the TMS320xx does the actual emulation.

9.4.3.2 Memories

Refer to Section 9.4.1.1.

9.4.3.3 Programmable Array Logic

The emulator card contains a number of programmable array logic (PAL)[28] devices.

A PAL device consists of a matrix of fusible diodes, AND gates, OR gates, and buffer/inverters that produce both an output and its complement.

In the XDS application, Texas Instruments reprograms the matrix by fusing selected vertical-horizontal diode junctions to form sophisticated combinatorial logic that does not need to be clocked or enabled.

---

[28] PAL is a trademark of Monolithic Memories, Inc.

9.4.4 Operation

A command from the keyboard goes through Port A through J41 on the Memory Expansion/Communications card into the XDS/22 backplane connector onto the TMS320C2x emulator card and into the SE9996. The SE9996 then causes the TMS320C2x to execute the commands. After execution, the TMS320C2x stops and waits for further SE9996 action.

Because the devices share data and addresses busses, the SE9996 knows where the TMS320C2x stopped and can exchange data with it.

9.4.5 Target-System Cable

A TMS320C2x XDS target cable (Figure B-1) consists of a printed-wiring board (PWB), 2-foot (0.6-meter) cable, and a target connector that plugs into your target system where the emulated device will eventually go, either directly (PLCC, FN package) or through a supplied PGA (pin-grid array) adapter. This combination of PLCC cable and PGA adapter is also available for replacement under Texas Instruments part number TMDS3288825. A similar cable with a ruggedized PGA connector rather than a PLCC connector is available under Texas Instruments part number TMDS32888820.

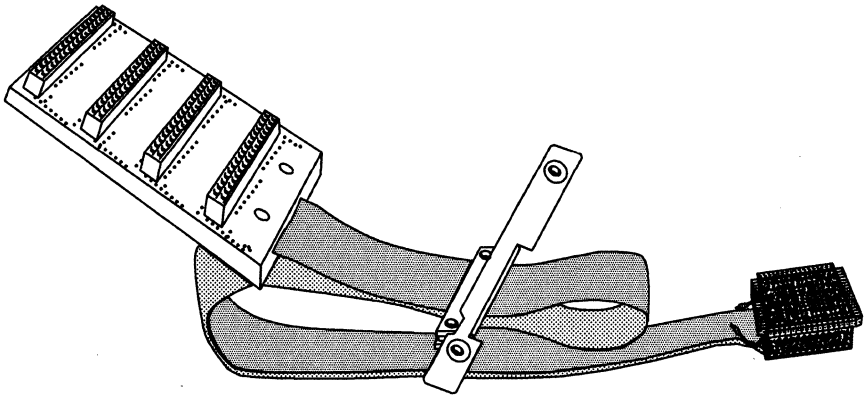


Figure 9-19. XDS/22 Target-System Cable

The PWB plugs into the male connectors on a 2540755-0001 patch board (Figure B-3).

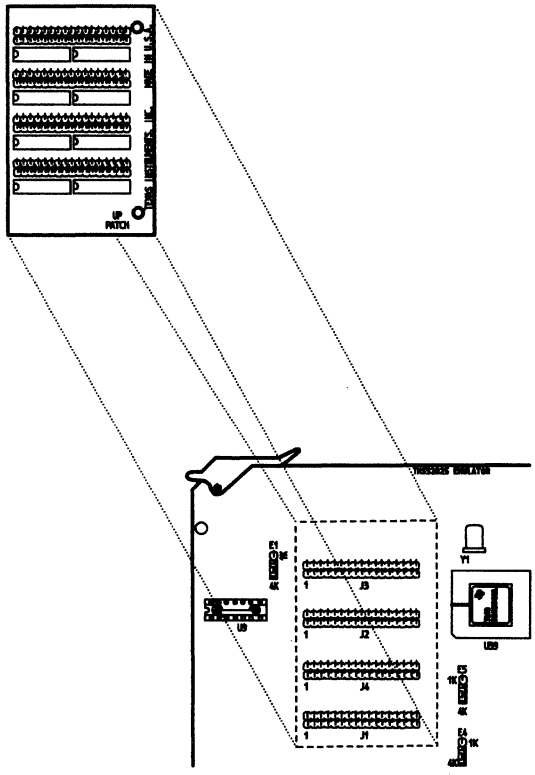


Figure 9-20. XDS/22 Target-System Cable Patch Board

The female connectors on the patch board plug into male headers J1 - J4 on the Emulator card (Figure 9-21)

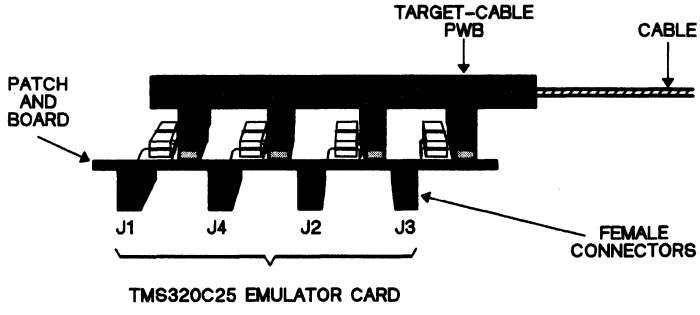


Figure 9-21. XDS/22 Target-System Cable - Emulator Card Connection

9.4.5.1 TMS320xx Target-Cable Connector

The target-cable connector supplied with a TMS32020 emulator has a 68-pin grid array with TMS32020 on the back side (left, Figure 9-22). Refer to the TMS32020 Users Guide, SPRU004B, for the pinout.

Caution:

The target-system connector pins bend and break easily.

Caution:

For electrical and physical protection, always keep the target-connector plugged into conductive foam or the plastic device furnished with the cable.

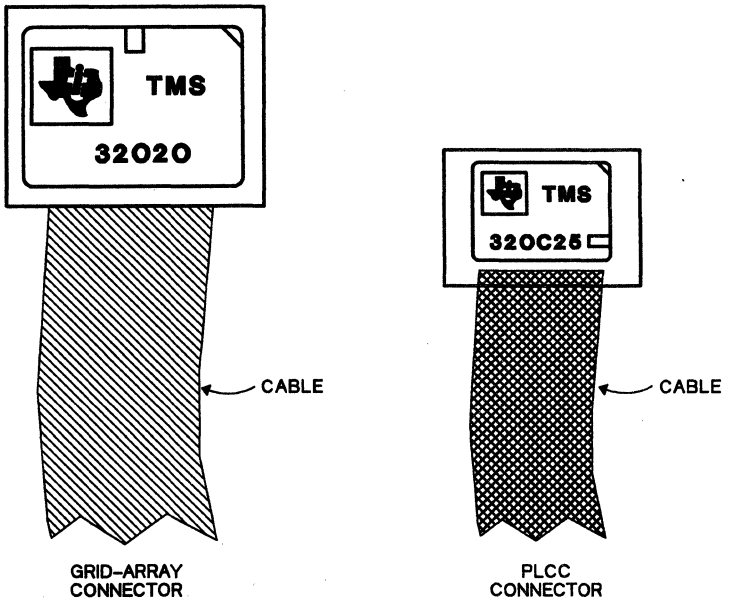


Figure 9-22. XDS/22 Target-System Cable Connectors

9.4.5.2 TMS320C2x Target-System Connector

The cable supplied with a TMS320C2x emulator or with a TMS32020 emulator upgrade kit has a 68-pin PLCC (plastic-leaded chip carrier) target-system connector for a TMS320C2x device (right, Figure 9-22) with the FN surface-mount package. You can also use this cable for a PGA device by installing the Texas Instruments part number 2248117-0008 converter socket included in your XDS spare-parts kit on the PLCC, then plugging that socket into your target-system PGA socket.

Section 5.9 describes connecting either target connector to or removing either from your target system.

9.4.5.3 Installation of Target Cable on Emulator Card

1) Unwrap target-cable.
2) Remove emulator card as described in Section 9.1.5.2.
3) Place on clean, static-free work surface with card ejectors toward you.
4) Install patch board on target-cable PWB (Figure 9-21).
5) Holding target cable with female sockets on patch board down and target-system connector toward you, carefully align patch board with J1 - J4 on right side of emulator card so that cable exits card past ejectors.
6) Carefully, but firmly, press patch board onto J1-J4 pins.

> **Caution:**
>
> **For electrical and physical protection, always protect the target-connector when not plugged into the target system.**

9.4.5.4 Removal of Target Cable from Emulator Card

1) After removing the emulator card (Section 9.1.5.2), remove target-system connector from target system by gently prying up on each corner in turn with a small flat-bladed screwdriver.

> **Caution:**
>
> **Do not pry on any adapter that may be installed on the target system.**

2) Remove target-system cable PWB from emulator card by gently pulling up at each end. Repeat as necessary. Take your time.

9.4.6 TMS320C2x Status-Light Assignments

With a TMS320C2x Emulator card installed, the four XDS operator-panel status lights show the following:

1 Emulator executing IDLE instruction. Extinguishes when emulator acknowledges an interrupt or enters the control mode.

The Breakpoint/Trace or Breakpoint/Trace/Timing card must be installed for this light to operate.

2 BIO (Branch on I/O Active) pin on TMS320C2x active.

3 TMS320C2x acknowledges target-hold interrupt by sending $\overline{\text{HOLDA}}$ (Hold Acknowledge) at logic 0 (active low).

4 Target system ready or target cable not connected to target system.

9.4.7 Target-System to Emulator Signal-Transfer Considerations

9.4.7.1 General

Table 9-9 details electrical characteristics for all emulator signals that go over the target cable. All signals pass directly to the target system, except those detailed later in this section. Directly-transferred signals are identical to normal TMS320C25 signals, except for some extra noise and a slight delay caused by the target-system cable.

Table 9-9. TMS320C2x Emulator Target Connector Electrical Characteristics

DATA/ADDRESS BUSSES

PIN SIGNATURE	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
D0-D15	I_{ih}	V_{cc}=5V, V_i=2.4V			51	µA
	I_{il}	V_{cc}=5V, V_i=0.6V			-0.56	mA
	I_{oh}	V_{cc}=5V, V_i=2.4V			-160	µA
	I_{ol}	V_{cc}=5V, V_i=0.6V			1.70	mA
A0-A15	I_{oh}	V_{cc}=5V, V_i=2.4V			-115	µA
	I_{ol}	V_{cc}=5V, V_i=0.6V			1.00	mA

INTERFACE-CONTROL SIGNALS

PIN SIGNATURE	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
$\overline{PS}$	I_{oh}	V_{cc}=5V, V_o=2.4V			-400	µA
	I_{ol}	V_{cc}=5V, V_o=0.6V			0.94	mA
$\overline{DS}$	I_{oh}	V_{cc}=5V, V_o=2.4V			-400	µA
	I_{ol}	V_{cc}=5V, V_o=0.6V			0.94	mA
$\overline{IS}$	I_{oh}	V_{cc}=5V, V_o=2.4V			-550	µA
	I_{ol}	V_{cc}=5V, V_o=0.6V			1.2	mA
$R\overline{W}$	I_{oh}	V_{cc}=5V, V_o=2.4V			-550	µA
	I_{ol}	V_{cc}=5V, V_o=0.6V			1.2	mA
$\overline{STRB}$	I_{oh}	V_{cc}=5V, V_o=2.4V			-505	µA
	I_{ol}	V_{cc}=5V, V_o=0.6V			0.93	mA
$\overline{READY}$	I_{oh}	V_{cc}=5V, V_o=2.4V			91	µA
	I_{ol}	V_{cc}=5V, V_o=0.6V			-1.40	mA

MULTIPROCESSING SIGNALS

PIN SIGNATURE	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
$\overline{BR}$	I_{oh}	V_{cc}=5V, V_o=2.4V			-280	µA
	I_{ol}	V_{cc}=5V, V_o=0.6V			1.9	mA
$\overline{HOLD}$	I_{oh}	V_{cc}=5V, V_o=2.4V			56	µA
	I_{ol}	V_{cc}=5V, V_o=0.6V			-0.31	mA
$\overline{HOLDA}$	I_{oh}	V_{cc}=5V, V_o=2.4V			-1.9	mA
	I_{ol}	V_{cc}=5V, V_o=0.6V			19	mA

Table 9-9. TMS320C2x Emulator Target Connector Electrical Characteristics (Concluded)

INTERRUPT AND MISCELLANEOUS SIGNALS

PIN SIGNATURE	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
$\overline{RS}$	I_{ih} I_{il}	V_{cc}=5V, V_O=2.4V V_{cc}=5V, V_O=0.6V			51 -0.56	µA mA
$\overline{BIO}$	I_{ih} I_{il}	V_{cc}=5V, V_O=2.4V V_{cc}=5V, V_O=0.6V			45 -0.45	µA mA
$\overline{IACK}$/IAQ	I_{oh} I_{ol}	V_{cc}=5V, V_O=2.4V V_{cc}=5V, V_O=0.6V			-200 0.90	µA mA
$\overline{MSC}$/VPA	I_{oh} I_{ol}	V_{cc}=5V, V_O=2.4V V_{cc}=5V, V_O=0.6V			-200 1.0	µA mA

SUPPLY/OSCILLATOR SIGNALS

PIN SIGNATURE	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
X1	Not connected on target connector					
X2/CLKIN	I_{ih} I_{il}	V_{cc}=5V, V_O=2.4V V_{cc}=5V, V_O=0.6V			20 -0.20	µA mA
$\overline{SYNC}$	I_{oh} I_{ol}	V_{cc}=5V, V_O=2.4V V_{cc}=5V, V_O=0.6V			110 -0.11	µA mA
CLKOUT1	I_{oh} I_{ol}	V_{cc}=5V, V_O=2.4V V_{cc}=5V, V_O=0.6V			-280 1.5	µA mA
CLKOUT2	I_{oh} I_{ol}	V_{cc}=5V, V_O=2.4V V_{cc}=5V, V_O=0.6V			-240 1.3	µA mA
V_{cc}	I_{oh}	V_{cc}=5V2.4V			1.1	mA

OTHER SIGNALS

PIN SIGNATURE	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
Inputs	I_{ih} I_{il}	V_{cc}=5V, V_O=2.4V V_{cc}=5V, V_O=0.6V			10 -10	µA µA
Outputs	I_{oh} I_{ol}	V_{cc}=5V, V_O=2.4V V_{cc}=5V, V_O=0.6V			-300 2.0	µA mA

The $\overline{BIO}$, $\overline{HOLD}$, $\overline{HOLDA}$, READY, $\overline{RS}$, $\overline{SYNC}$, and X2/CLKIN signals require further processing.

9.4.7.2 Branch Signal

Branch signal $\overline{BIO}$ requires the timing shown in Figure 9-23.

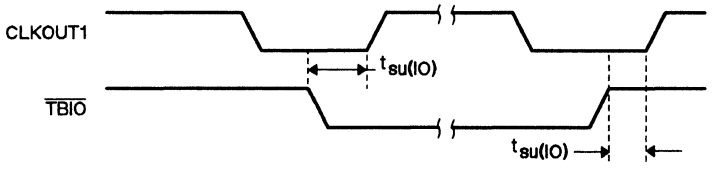


Figure 9-23. Target BIO Switching Parameters

$T_{su(IO)}$ is the target-system $\overline{BIO}$ setup time before CLKOUT1 goes high. It is a minimum of 15 ns more than $t_{su(IN)}$ for the emulated device (refer to its Users Guide).

9.4.7.3 Clock and Sync Signals

Clock and Sync signals $\overline{SYNC}$ and X2/CLKIN have a matched delay such that the timing listed in the device Users Guide works with the emulator also.

9.4.7.4 Hold Signals

Hold signals $\overline{HOLD}$ and $\overline{HOLDA}$ require the timing shown in Figure 9-24.

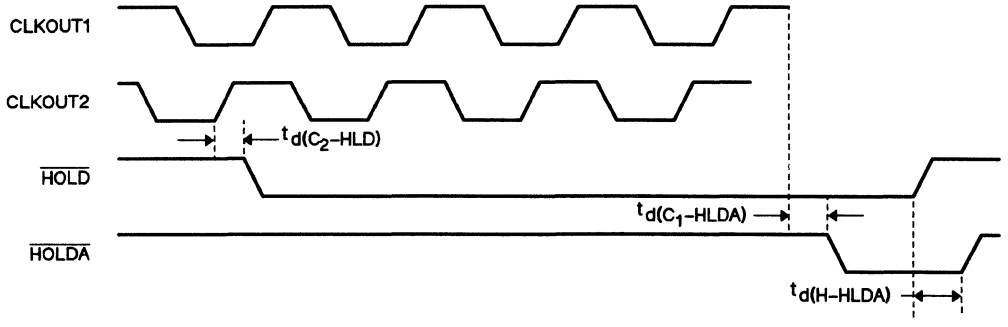


Figure 9-24. HOLD and HOLDA Switching Parameters

- The time for $\overline{HOLD}$ to become valid after CLKOUT2 goes high is a maximum of 25 ns more than the device $t_{d(C2H-H)}$ specification.
- The time for $\overline{HOLDA}$ to become valid after CLKOUT1 goes low is from 6 to 25 ns more than the device $t_{d(C1L-AL)}$ specification.
- The time from $\overline{HOLD}$ going high to $\overline{HOLDA}$ going high is a maximum of 5 ns more than the device $t_{d(HH-AH)}$ specification.

9.4.7.5 READY Signal

Ready signal $\overline{\text{READY}}$ requires the timing shown in Figure 9-25.

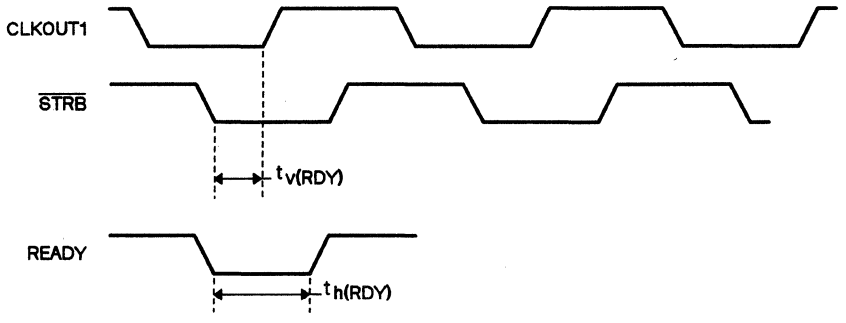


Figure 9-25. Ready Switching Parameters

$\overline{\text{READY}}$ valid after $\overline{\text{STRB}}$ goes low is maximum of 11 ns more than the device $t_{d(SL-R)}$ specification. $\overline{\text{READY}}$ hold time after $\overline{\text{STRB}}$ goes low is a minimum of 4 ns *less* than the device $t_{(SL-R)}$ specification.

9.4.7.6 Reset Signal

Reset signal $\overline{\text{RS}}$ requires the timing shown in Figure 9-26.

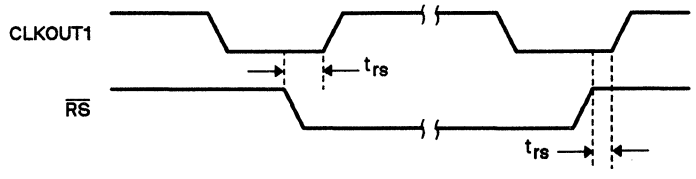


Figure 9-26. Reset Switching Parameters

Reset time is a minimum of 12 ns more than the device $t_{su(IN)}$ specification.

9.4.8 Conversion to TMS32020 Emulation

To convert a TMS320C2x emulator card, either as supplied with a TMS320C2x XDS/22 or as upgraded from a TMS32020 Emulator card with upgrade kit 2540760-0001, follow this procedure:

1) Turn XDS/22 power off.

2) Power target system off, then unplug target-cable connector from system as follows:

   a) If target-system connector is pin-grid array, carefully lift up on each corner, one at a time, with a small flat-bladed screwdriver. Be sure to lift on the connector, not the socket soldered to your system.

   b) If target connector is PLCC, insert a pointed object such as a sheet-metal marking scribe in either long notched and lift up on connector, then repeat in other long notch. Repeat until connector comes free.

3) At XDS/22, loosen two thumbscrews holding target cable to XDS strain-relief system.

4) Unseat Emulator card by pulling out on inside edge of card-ejector tabs.

5) Pull Emulator card out of XDS. Be careful not to snag any cables.

6) Set Emulator card on clean, static-free work surface, component side up.

7) Remove TMS320C2x from Emulator card as described above for removing target-cable connector from target system.

8) Insert TMS32020 in socket with notched corner toward backplane-connector P2.

9) Replace Emulator card in XDS/22 as described in Section 9.1.6.2.

9.5 Maintenance

> **Note:**
>
> Texas Instruments assumes no liability or contractual requirements by publication of maintenance schedules, procedures, and time intervals in this section; and intends these procedures only as a *guide* for qualified service personnel. **Procedures subject to change without notice.**

The only regular user maintenance is normal cleaning of the system cabinet and air filters.

9.5.1 Cleaning Cabinet

Texas Instruments recommends that you clean the cabinet monthly, or as necessary.

1) Turn power switch OFF and unplug line cord.
2) Wipe all unit surfaces with damp, lint-free cloth. For heavy dirt build-up, use mild soap solution. Avoid spraying soap solution on unit.
3) Clean air intake and exhaust areas.
4) Plug line cord back in.

9.5.2 Cleaning Air Filters

Texas Instruments recommends that you check the air filters monthly, or as necessary.

1) Turn power switch OFF and *unplug unit from power source.*
2) Remove top cover as described in Section 9.1.1.3 on Page 9-4.
3) Remove air filter from each side of top cover by removing four retaining nuts. Lift each filter from mounting screws, with assistance from screwdriver, if necessary.
4) Hold each filter up to window or overhead light. If either filter very dirty, clean both as follows:
   a) Wash each filter in mild soap and water solution.
   b) Remove as much water as possible by blowing *dry* compressed air through filter or patting with paper towels.
   c) When completely dry, spray with water-soluble adhesive following directions on can label.
5) Installation is reverse of removal.

> **Caution:**
>
> **Each filter must be completely dry before reinstallation.**

Caution:

Do NOT operate an XDS/22 without both air filters installed! In addition to blocking dust and dirt, these filters prevent accidental entrance of small objects through the ventilation slots.

9.6 XDS/22 System Repair Guide

- If an XDS/22 does not power up when turned on, that is, the POWER light on the Operator Panel doesn't light and the fan doesn't run, refer to Section 9.6.1.
- If either the fan runs, but the POWER light doesn't light, or vice-versa, refer to Section 9.7.

> **Warning:**
>
> **Do not operate your XDS/22 if the fan doesn't work.**

- If the XDS/22 powers up, but the associated terminal does not display anything, refer to Section 9.6.2.
- If the XDS/22 and terminal work, but the target system doesn't respond, refer to Section 9.6.3.
- If one or more XDS/22 units do not work properly in a multiprocessing application, refer to Section 9.6.4.
- If an XDS/22 does not work with a host computer, refer to Section 9.6.5.

9.6.1 XDS/22 Does not Power Up.

If an XDS/22 does not power up when turned on, that is, the POWER light on the Operator Panel doesn't light and the fan doesn't run, check the following:

1) Line cord defective or not connected to outlet or unit.
2) No power at outlet (check with voltmeter or plug known-good lamp or apparatus into outlet).
3) Chassis fuse blown (Refer to Section 9.6.6).

If none of these apply, refer to Section 9.7.

9.6.2 System Menu not Displayed

If the XDS/22 powers up, but the associated terminal does not display the system menu, follow this procedure:

1) If the terminal displays nothing, follow normal troubleshooting procedures for dead equipment.
2) If the terminal screen displays a cursor, but doesn't display the XDS menu, do this:
   a) Self-test the terminal, if possible.
   b) Be sure its communication settings are seven data bits, two stop bits, and even parity.
3) If the terminal appears to be operating, power it off, then power the XDS off by pressing the lower half of its POWER switch, then disconnect the terminal - XDS cable and check its wiring against the appropriate drawing in Section 7.1.

4) If the cable is OK, or after repairing it, reconnect it and power the system up again.

> **Note:**
>
> Remember to wait at least 10 seconds after the terminal cursor appears, and to press RETURN twice.

5) If the problem remains, turn power off, remove card-cage cover (Page 9-12), then unseat Memory Expansion/Communications card in slot 4 by pulling out on inside edge of each card-ejector tab simultaneously, then reseat card by pressing on both card-ejector tabs simultaneously.

6) Remove and replace each of J41, J45, and J47.

7) Power the system up again. If the terminal doesn't display the system menu now, turn power off, then check settings of S3 and S4 against the appropriate tables in Section 7.1.

8) If OK, or after resetting them, reseat the Memory Expansion/Communications, then power the system up again.

9) If the problem remains, turn power off, then unseat and reseat emulator card in slot 2.

10) Power the system up again. If the terminal doesn't display the system menu now, turn power off, then unseat the Breakpoint/Trace/Timing card *and leave it unseated.*

11) Power the system up again. If the terminal doesn't display the system menu now, turn power off, reseat the Breakpoint/Trace/Timing card, then power the system up again. If the menu still doesn't appear, replace the card-cage cover (Page 9-12) and refer to Section 9.7.

9.6.3 Target System Doesn't Respond

If the XDS/22 and terminal work, but the target system doesn't respond, follow this procedure:

1) Check target-system power.

2) If OK, power XDS and target system off, then remove target connector from system and check for broken or bent pins.

3) If OK, remove emulator card from XDS as follows and check that target-system cable is properly connected to card:

   a) Remove card-cage cover (Page 9-12).

   b) Loosen two captive screws holding target-system cable strain-relief bracket.

   c) Unseat emulator card by pulling out on card-ejector tabs.

   d) Slide emulator card out of card cage.

   e) Carefully remove target cable from emulator card and check for bent or broken pins on card.

f) If OK, carefully replace cable on card.

g) Move card-ejector tabs parallel to card sides.

h) Slide emulator card into card-cage slot 2 until card just touches sockets, then back card out slightly until you can rotate outside edge of each ejector tab into small square hole on side of card cage (Figure 9-9).

i) Press card into backplane card connectors by pressing on card-ejector tabs.

If target system still does not respond, refer to Section 9.7.

9.6.4 Multiprocessing Application Doesn't Work

Check XDS and target-system power at any unit that cannot be assigned an ID. If any unit or units fails to operate properly in Group mode, check continuity from pin 25 to pin 25 in each connecting cable.

9.6.5 XDS/22 Doesn't Work with Host Computer

Table 9-10. System Problems Diagnostic Chart

SYMPTOM	PROBABLE CAUSE(S)
Unable to communicate with host computer.	Parameters set by IPORT command not compatible with host computer. Incorrect cabling or connected to wrong port. Wrong communications card switch settings (Page 9-17). Host not sending proper signals (DSR, CTS).
HOST PORT OFF-LINE message displayed	DSR or CTS lines to XDS low. Change switch settings or have host send proper signals.
UPLOAD-COMPLETE message occurs during upload.	Emulator received upload-start character from host. Use different handshake or IHC command to select NONE for protocol in UL and DL commands.
ERRORS INVALID TAG displayed in Terminal Mode but no download performed	Emulator received download-start character from host or user terminal. Use different handshake or IHC command to define new control characters and select NONE for protocol in UL and DL commands.
Object file not in memory after download	Check all DL parameters, especially DESTINATION.
Characters lost during data transfers.	Does host respond to XON/XOFF or RTS/CTS? If RTS/CTS, are lines connected?
Emulator exits HOST Mode without entry of CNTL/E	Does terminal (intelligent or personal computer) send CNTL/E for other control functions or use hardware lines #17 or #25?
Unable to communicate with modem.	Does modem use DSR signal (line #20)? If so, change switches to drive line high. Emulator must enable any RS-232 signals that terminal would.
DL or UL does not work work correctly with IHC	Execute IHC command before executing DL or UL commands if NONE or TEK protocol used.
Last record of file on update not complete	If upload-end character destroys last record, use IHC command to select different control characters.

9.6.6 Fuse Replacement

Only the AC line fuse on the XDS/22 rear panel is replaceable. Use this procedure:

1) Press OFF - 0 (lower) segment of POWER switch, then unplug line cord from receptacle.

2) Remove power cord from back of unit.

3) The power-cord connector also houses the fuse (Figure 9-3). Pry tab outward with small screwdriver.

4) Pull drawer unit out from power connector.

Warning:

Do not connect XDS/22 to power source with fuse drawer pulled out.

5) Lift narrow tab and pull fuse platform from drawer unit.

6) Replace fuse, if necessary, with appropriate selection from Table 9-11. Do **NOT** overfuse!

Table 9-11. Fuse Replacement Guide

XDS MODEL	FUSE RATING			TI PART NUMBER	EQUIVALENT
	VOLTS	AMPS	TYPE		
2310990-1	125	3	SB	410822-60	Buss MDX-3
2310990-2	250	2	SB	2220531-4	Schurter 034.3120

7) Replace fuse platform in fuse drawer and push in until it locks in place.

8) Insert fuse drawer into power bulkhead until it is flush with the bulkhead.

9) Insert power cord in fuse holder/connector, then plug line cord into power receptacle.

10) Press POWER switch on. If line fuse holds, proceed with normal operation.

11) If line fuse opens (fan not running), press POWER switch lower segment, then follow procedure from Section 9.1.5.2, Page 9-11, and unseat all XDS cards.

12) Replace line fuse as previously described, reconnect power, then press POWER switch on. If line fuse opens, press POWER switch lower segment, unplug line cord, and refer to Section 9.7 to send equipment in for factory exchange or repair.

13) If line fuse holds, press POWER switch lower segment, and reseat emulator card.

14) Press POWER switch on.

15) If line fuse opens, press POWER switch off and send emulator card in for exchange or repair as detailed in Section 9.7.

16) If line fuse holds, repeat previous two steps, but reseat Memory/Communications card.

17) Repeat these steps for each remaining card. If any card causes the fuse to open, send that card in for exchange or repair as detailed in Section 9.7. After receiving the repaired or exchange card, don't forget to replace the fuse before attempting to operate the XDS.

If these procedures do not isolate the problem, refer to Section 9.7.

9.7 Factory-Repair Information

The Texas Instruments Microprocessor and Microcontroller Products Division (MMPD) Factory-Repair Center at 9901 South Wilcrest, Houston, TX, 77099, (713) 879-2285, offers warranty repair or exchange at no charge[29] and non-warranty repair at standard labor and material rates for all current XDS products.

Texas Instruments also offers expedited service on exchange at an additional cost (Section 9.7.5).

Texas Instruments considers an XDS/22 to be in warranty if all these conditions apply:

1) You notify Texas Instruments of the problem within 90 days of purchase from Texas Instruments or an authorized distributor

2) The Factory-Repair Center receives the unit in standard Texas Instruments configuration with no customer additions

3) Factory-Repair Center inspection shows problem(s) not caused by accident, alteration, improper installation, improper testing, misuse, neglect, or unauthorized repair.

If you believe that your unit is in warranty according to these guidelines, please read the remainder of this topic through Section 9.7.6. If you believe that your unit is not in warranty, please read Section 9.7.1 and Section 9.7.7.

> **Note:**
>
> Texas Instruments accepts no responsibility for customer-installed changes, including, but not limited to, customer-generated software. Texas Instruments also reserves the right to *not* repair and return at customer expense any product that cannot be tested to Texas Instruments specifications because of customer changes.

This Section contains the following topics:

[29] Except shipping.

9.7.1 Shipping Information

For any factory repair, do this:

1) Contact the Factory-Repair Center at (713) 879-2285 and ask for a Return Authorization Number (RAN) and an exchange/repair questionnaire.

2) Fill out the questionnaire. Be sure to include the RAN. Make a copy for your records.

3) Pack the XDS carefully and securely (preferably with the original packing material in the original shipping box) and send post or freight prepaid to:

   Texas Instruments Incorporated
   Microprocessor and Microcontroller Products Division
   Factory-Repair Center, MS 6400
   9901 South Wilcrest
   Houston, TX 77099

4) The package must contain the exchange/repair questionnaire completely filled out with the following information:

   - RAN (Remember that Texas Instruments cannot accept returned equipment without this number)
   - Customer name, contact name, and telephone number
   - For warranty repair:

     a) Proof of date of purchase (required)
     b) Payment for return freight, if applicable (Section 9.7.4).
     c) Payment for expedited exchange, if that service requested.

   - For non-warranty repair, payment (Section 9.7.4), if applicable, including expediting charges, if that service requested.
   - Model number (XDS/22) and serial number (from model plate on rear)
   - "Ship-to" information, including address, amount of insurance, and shipping method. Texas Instruments ships UPS insured (minimum amount) unless you specify otherwise.
   - Invoicing address, if applicable (Section 9.7.4).
   - Request for return of same serial-numbered unit, if desired.
   - Symptoms (please be as specific as possible) and type of service requested.

5) Make a copy of the waybill for your records.

Upon receipt, Texas Instruments inspects the equipment. If it does not meet the warranty conditions described on Page 9-49, Texas Instruments will notify you immediately as described in Section 9.7.7.

9.7.2 Warranty Exchange or Repair

If accepted for warranty repair, and you asked for return of a specific XDS/22, Texas Instruments will repair that specific unit. Current *maximum* repair time is 45 working days.

If accepted for warranty repair, but you did not ask for return of a specific XDS/22, Texas Instruments reserves the option to repair your returned unit or exchange it for an equivalent unit. Normal *maximum* exchange time is 10 working days; expedited exchange time is 1 working day.

Upon completion of warranty exchange or repair, Texas Instruments returns the repaired or exchange XDS/22 as described in Section 9.7.3.

9.7.3 Repair Completion and Product Return

Texas Instruments tests a repaired system or board or an exchange system or board before returning it.

Texas Instruments returns the repaired or exchanged product to your "ship-to" address. Shipment is F.O.B. the Factory-Repair Center at 9901 South Wilcrest, Houston, TX, 77099 by your requested shipping method (UPS, with minimum insurance, if you do not specify anything).

> **Note:**
>
> You may receive a refurbished product in exchange. Any such unit meets Texas Instruments standards for refurbished products, but may have minor visual blemishes, such as touched-up paint.

9.7.4 Charges and Payment

The Factory-Repair Center offers both system- and board-level service with charges as listed in Table 9-12. System-level service usually includes upgrading to the latest product-revision level at no additional charge.[30]

You pay freight and insurance charges on the returned product.

Payment can be included with your unit in any of these forms:

- Certified, cashier, or company check payable to Texas Instruments Incorporated
- Money order payable to Texas Instruments Incorporated
- Company purchase order.

For a purchase order, Texas Instruments will bill your company at its invoicing address. The return-shipping date is the invoice date.

If you do not include payment and have not established credit with the Semiconductor Group of Texas Instruments, the return shipment will be C.O.D.

---

[30] This does not include specific separately-marketed upgrades for XDS/22 units.

Table 9-12. Factory Repair and Replacement Charges

SERVICE	WARRANTY CHARGE	NON-WARRANTY CHARGE	MAXIMUM ESTIMATED REPAIR TIME
SYSTEM LEVEL			
Normal repair	N/C	$350.00	45 working days
Normal exchange	N/C	$350.00	10 working days
Expedited repair	$200.00	$500.00	1 working day
BOARD LEVEL			
Normal repair	N/C	$200.00	45 working days
Normal exchange	N/C	$350.00	10 working days
Expedited repair	$200.00	$500.00	1 working day

9.7.5 Expedited Exchange

9.7.5.1 Of Returned Units

The Factory-Repair Center accepts expedited exchange requests *if and only if* it has the product in inventory. The Center sends the exchange item not later than 1 working day after receipt of the bad unit.

9.7.5.2 Of Non-Returned Units

Texas Instruments can also send the exchange unit before receiving your unit if you have credit with the Semiconductor Group. Call the Factory-Repair Center at (713) 879-2285 with the number of a purchase order for the current retail price, plus freight (F.O.B. Repair Center), and insurance.

After determining that your unit is repairable, Texas Instruments credits you with the difference between the retail price and the exchange charge.

9.7.6 Warranty on Repaired or Exchange Products

Texas Instruments guarantees workmanship and new or refurbished parts in repaired or exchange products for 30 days from date of shipment or for the remainder of the original warranty period, whichever is longer.

The foregoing warranty for goods is in lieu of all warranties, express, implied, or statutory, including, but not limited to, any implied warranties of merchantability and fitness for a particular purpose, and of any other warranty obligation on the part of Texas Instruments.

9.7.7 Non-Warranty Exchange or Repair

Texas Instruments also offers exchange or repair service for XDS products that do not qualify for warranty repair as described on Page 9-49.

After receipt, Texas Instruments evaluates the repair cost at current labor and material rates. If that repair cost is:

1) Greater than the minimum Texas Instruments repair cost (Table 9-12), or
2) Greater than the unit's current retail price,

Texas Instruments notifies you and requests authorization for repair, exchange, or return F.O.B. the Factory-Repair Center.

Repair charges include any removal or repairs of customer-installed changes required for the unit to meet Texas Instruments specifications for that unit.

Call the Factory-Repair Center for information about non-warranty repair of units no longer in production.

Upon completion of non-warranty exchange or repair, Texas Instruments returns the repaired or exchange XDS/22 as described in Section 9.7.3.

Pagination

10. System Messages and Error Codes

Table 10-1. TMS320C2x Emulator Error Messages

CODE	MESSAGE	DEFINITION
0XXX	SCAN PASS ERROR	Error found during first command-buffer pass. No commands executed.
01XX	TERMINATION ERROR	Invalid terminator character.
0100	INVALID BUFFER TERMINATOR	Invalid buffer-terminator character.
0101	INVALID TERMINATOR	Invalid terminator character.
0105	TERMINATOR NOT EXPECTED	Parenthesis-mode command entry omitted one or more parameters.
0108	END OF BUFFER EXPECTED	Data entered after * in command line.
02XX	COMMAND ERROR	Unrecognizable command entered.
0202	INVALID COMMAND	Command name misspelled, run together with other commands, or not recognized.
0209	INDEX NOT EXPECTED	Found number index associated with non-indexed command or register.
0210	INVALID WITH OTHER INPUT	Input that must be only command in input buffer found with other input.
03XX	SYNTAX ERROR	Invalid separator character.
0304	INVALID SEPARATOR	Character used is not valid.
0306	SEPARATOR NOT EXPECTED	Separator character not expected.
0307	SEPARATOR EXPECTED	Invalid separator character.
04XX	PARAMETER ERROR	Parameter entered incorrectly.
0403	INVALID PARAMETER	Invalid characters in parameter entry.
040F	PARAMETER TOO LARGE	Parameter value entered exceeds limit for that parameter.
05XX	VARIABLE ERROR	Error in variable entry.
050A	VARIABLE INDEX NOT EXPECTED	Index added to non-indexed variable.
050B	INVALID VARIABLE INDEX	Entry either no index or non-decimal index.
050C	VARIABLE INDEX TOO LARGE	Index greater than maximum allowable value.
050D	VARIABLE NOT WRITABLE	Value assigned to read-only variable.
06XX	COMMAND USAGE ERRORS	Command used out of expected sequence, or execution of another command required before execution of current command.
0601	SNAP MUST BE FIRST	SNAP must be first command in procedure.

Table 10-1. TMS320C2x Emulator Error Messages (Continued)

CODE	MESSAGE	DEFINITION
0602	CURSOR CONTROL NOT INITIALIZED	ICC command must execute before executing SNAP.
1XXX	EXECUTION PASS ERROR	Error detected during second pass. All preceding commands executed.
10XX	LOAD ERROR	Errors detected while loading user object module into emulator RAM.
1000	INVALID TAG	Invalid tag found during object-module loading.
1010	CHECKSUM ERROR	Checksum value at end of object-code line with 7-tag did not agree with line contents.
1011	UPLOAD END ADDRESS LESS THAN START ADDRESS	Upload-end address must be greater than start address.
1012	UPLOAD ERROR - 15 RETRIES	Tektronics hex-tagged object record transmitted unsuccessfully 15 times in response to requests to retransmit record.
1013	EXTERNAL REFERENCE TAG IN OBJECT MODULE	External-reference tag found during downloading of TI object file ignored. Message prints after download to inform user of unresolved object-file reference. Program execution possible, but may result in error.
1014	ASR PROTOCOL INVALID	ASR handshake not supported when uploading to PROM or USER port.
1015	INVALID TEK HANDSHAKE	No valid TEK handshake on upload.
1016	USE EMULATOR #1	Command valid only for #1 emulator in multi-processing chain issued when another emulator in foreground.
1112	LOG DEVICE OFFLINE	LOG command turned off because character couldn't be transmitted to log device. Check EIA connection.
1113	PROM PORT OFFLINE	PROM upload aborted because CTS and/or DSR signals on Port C inactive. Check EIA cable and Communication Card switch settings.
1114	HOST PORT OFFLINE	HOST Mode aborted because CTS and/or DSR signals on Port D inactive. Check EIA cable and Communication Card switch settings.
1120	ADDRESS IS NOT RAM	Attempt to set software breakpoint in ROM.
1121	SOFTWARE BREAKPOINT ALREADY SET	Attempt to set second software breakpoint at command address
1122	SOFTWARE BREAKPOINT OVERFLOW	Software breakpoint limit exceeded. If new break-point to be set, clear one or more existing break-points first.
1123	NON-EXISTENT BREAK-POINT	Attempt to clear breakpoint never set.
1130	NO BREAKPOINT CARD	Command execution requires installation of Breakpoint/Trace card.

Table 10-1. TMS320C2x Emulator Error Messages (Continued)

CODE	MESSAGE	DEFINITION
1131	USE BP,BPM,TR,TRM	BTT or IBTT command entered with BT card installed.
1132	USE BTT,IBTT	BP,BPM,TR, or TRM command entered with BTT card installed.
1133	BTT CARD REQUIRED	DTIME or XTIME command entered with BT card installed.
1140	PARITY ON MEMORY EXPANSION	Memory Expansion/Communications card missing or detected parity error. Can occur if external clock signal interrupted. To check, reinitialize and use emulator clock.
1141	MODE CONFLICT	Overlapping memory block already mapped for substitution. Reinitialize.
1145	NON-EXISTENT BREAKPOINT	Attempt to clear breakpoint not set by SSB command.
1155	NO MEMORY CARD IN UNIT	No or bad Memory/Expansion Communications card in XDS/22 or command requiring memory issued with XDS/11.
1316	ARM - COMMAND NOT PERMITTED	Command requiring emulator CPU to halt attempted in Alternate-Run mode with MEMORY-COMMAND-PERMITTED parameter disabled.
1317	ARM - INVALID COMMAND	Command entered that is never permitted in Alternate-Run mode. Refer to ARM command description.
1501	INVALID TERMINATOR	Characters following label or before mnemonic are not spaces, or character after mnemonic is not space or carriage return.
1502	INVALID MNEMONIC	
1503	INDIRECT ADDRESSING REQUIRED	This mnemonic requires indirect addressing.
1504	"+" or "-" REQUIRED	"0" or "BR" addressing mode requires "+" or "-".
1505	INVALID EXPRESSION	
1506	INVALID REGISTER NUMBER	Number must be 0-4 for TMS32020 or 0-7 for TMS320C25
1507	ALPHA REQUIRED	
1508	"'" REQUIRED	
1509	DUPLICATE LABEL	Label already exists.

Table 10-1. TMS320C2x Emulator Error Messages (Concluded)

CODE	MESSAGE	DEFINITION
1510	LABEL TABLE FULL	No additional symbols can be accepted.
1511	OPERAND REQUIRED	Mnemonic requires operand.
1512	INVALID SHIFT COUNT	Shift count must be 0,1, or 4 for TMS32020, or 0-7 for TMS320C25.
1513	DEFINED EXPRESSION REQUIRED	Label must be previously defined
1514	OPERAND TOO LARGE	
1515	ASSEMBLING INTO ROM	
17XX	MULTIPROCESSING ERRORS	Command execution ended because of conflict during execution.
1701	INVALID WITHOUT IMP CMD	Initialize with IMP command first.
1705	EMULATOR #X NOT READY TO RUN	Emulator X, x = 1-9, told to run, but is waiting to output after execution of TRUN or GRUN command.
1710	LAST EMU = 1	IMP command executed with emulator not in Multiprocessing mode; or problem with connections between multiprocessor units.
1715	INVALID WITH IMP	Command invalid in Multiprocessing mode.
2xxx	Emulator error	Emulator hardware or software problem. Pressing RESET or turning power off, then back on may solve problem.
2010	SYSTEM STACK OVERFLOW	Emulator software problem. Press RESET.

Table 10-2. TMS320C2x Emulator Warning Messages

CODE	DEFINITION
0001	Trailing characters in operand field ignored.
0002	Data too large for one byte.
0003	Evaluated expression data too large for two bytes.

A. XDS Reference Material

This Appendix contains the following:

A.1 Functional Command Grouping

Table A-1. Emulator Commands Grouped by Function

Initialization	
DEC	Decimal Format
HEX	Hexadecimal Format
ICC	Initialize Cursor Control
IHC	Initialize Host Control
IMD	Initialize Multiprocessing
INIT	Initialize Emulation
IPORT	Initialize Parameters
IPRM	Initialize Parameters
ITR	Initialize Target RAM
LOAD	Load Stored Parameters
MAG	Magnitude Format
MAP	Map Expansion Memory
RCC	Restore Cursor Controls
RESTART	Restart
SNAP	Freeze Display
Memory	
DDM	Display Data Memory
DPM	Display Program Memory
FILL	Fill Memory with Data
FIND	Find Data in Memory
IDM	Inspect Data Memory
IPM	Inspect Program Memory
MDM	Modify Data Memory
MPM	Modify Program Memory
Communications	
DIO	Display I/O
MIO	Modify I/O
Offload	
DL	Download
UL	Upload
Run	
CRUN	Continue Run
GHALT	Group Halt (MP Mode)
GRUN	Group Run (MP Mode)
RTR	Run on Target Reset
RUN	Continuous Execution
SS	Single-Step Execution
STOP	Stop Alternate-Run Mode
THALT	Total Halt (MP Mode)
TRUN	Total Run (MP Mode)

Mode	
ARM	Initiate Alternate-Run Mode
BGND	Initiate Background Mode
DISARM	End Alternate-Run Mode
HOST	Initiate Host Mode
IMP	Initiate Multiprocessing Mode
#n	Select Emulator #n
Hardware Breakpoint/Trace	
BTT	Set BTT Parameters
DBTT	Show BTT Parameters
DT	Display Trace
DTIME	Show Time Settings
FT	Find Trace
IBTT	Initialize BTT
IT	Inspect Trace
XTIME	Time Analysis
Software Breakpoint	
CASB	Clear Software Breakpoints
CSB	Clear one Software Breakpoint
DSB	Display all Breakpoints
SSB	Set one Breakpoint
Assembler	
XA	Execute Assembler
XRA	Reverse Assembler
Status	
DES	Display Emulator Status
DHS	Display Halt Status
DMAP	Display Memory Map
DPS	Display Processor Status
DTS	Display Trace Status
/n	Display Emulator MP Status
Miscellaneous	
DV	Display Values
HELP	Display Command Menu
ID	Display Firmware Revision
LOG	Print Commands and Response
SAVE	Save Parameters
Registers	
DR	Display Registers
IR	Inspect Registers
MR	Modify Registers

A.2 Data Transfer

An XDS/22 supports the following data-transfer features:

- RS-232C levels and pin definitions
- Character, record, and file-level control
- Intel, Motorola, Tektronix, TI normal, and TI compressed formats
- ASR, None, Tektronix, and VAX protocols

The XDS Memory Expansion/Communications card performs all RS-232C level conversion. The Emulator monitor program performs all formatting and protocol generation. You select the format and protocol by responding to prompts in the DL (Download), IHC (Initialize Host Control), IPORT (Initialize Port Control), and UL (Upload) commands.

A.2.1 General

A.2.1.1 Character Bit Pattern

The following listing and Figure A-1 shows the bit pattern for asynchronous transmission of any character:

- One Start bit, always logic 0, low
- Seven or eight data bits, depending on the format. Most use seven. Logic level depends on data.
- One parity bit. Logic level depends on format parity selection and character data. Bit ignored if no parity selected.
- One or two Stop bits, depending on selected format. Always logic 1, high.

Uploading emulator output must include a parity bit to match host parity requirements. Downloading does not check parity.

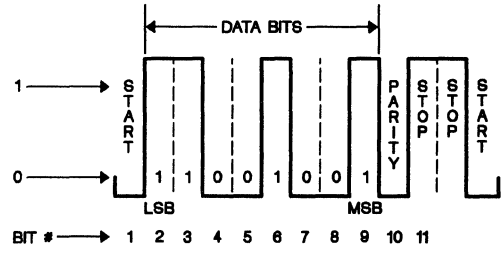


Figure A-1. Bit Pattern with Eight Bits/Character

A.2.1.2 Bit-Pattern Changes

Use the IPORT (Initialize Port) command to set number of data bits and number of stop bits to conform to the host requirements.

Object-file format also has a bearing in configuring parameter values.

A.2.2 Data-Transfer Control

A.2.2.1 Character-Level Control

The emulator supports standard I/O XON/XOFF interrupt functions except with the Tektronix format. With XON, data downloads into an 80-character buffer. After receiving 60 characters, the emulator sends XOFF. When the XDS has processed all but 20 buffer characters, it sends XON again.

You can also use the RS-232C CTS line for character-transfer control, if the external device recognizes that signal.

A.2.2.2 Record-Level Control

The monitor program supports record-level control signals for Tektronix protocols. The XDS also automatically configures the port for communications with the external device.

Special end-of-record character sequences indicate record-transfer status as follows:

ASCII 0<CR> Record received without error. (Checksum data detects data errors.)

ASCII 7<CR> Transmission error. Retransmit. Upload session aborts after 15 unsuccessful attempts.

XON/XOFF and CTS/RTS signal control does not work with the Tektronix format.

A.2.2.3 File-Level Control

Uses special records for control characters at beginning and end of each object file for control signals. The TMS320C25 Emulator supports object file-level control of each format described in the next topic.

The latter two are available for upload or download to or from a host computer.

A.2.3 Formats

The TMS320C2x Emulator supports the following formats:

1) Intel Intellec 8/MDS
2) Motorola Exorcisor format
3) Tektronix (TEK) hexadecimal format
4) TI normal ASCII format
5) TI compressed format

A.2.3.1 Intel Data Format

1) Description

The Intel data format downloads to a PROM Programmer or uploads to a host computer. The format consists of data and end-of-file records.

2) Data Record

Bytes	*Description*
1	Header/Start Character. Always a colon.
2,3	Byte-count. Number of bytes in record.
4-7	Address of first data byte.
8,9	Record type: 00 for data record.
10,nn	Each data byte formatted as two hexadecimal digits. Number of bytes must equal number in bytes 2 and 3.
nn+1,nn+2	Checksum. Two's-complement of binary sum of all previous bytes in record.
nn+3	Optional. Can have line feed or comment.
nn+4	(nn+3 if optional byte omitted.) End-of-record character: <CR>.

3) End-of-File Record

An end-of-file record contains only nine bytes similar to the first nine data-record bytes, except that the record type is 01.

4) Example

The following example illustrates a typical set of Intel data records.

```
:1E00000055005500550055005500550055005500550055005500550055005500E7
:1E001E0055005500550055005500550055005500550055005500550055005500C9
:00C00301
```

A.2.3.2 Motorola Record Formats

1) Description

- **Uses** -- Downloads to PROM Programmer or uploads to host computer.
- **Record Types** - Sign on (optional), data record and end-of-file record.
- **Structure**

  8-character prefix
  Data byte
  2-character suffix
  End-of-record character

2) Sign-on Record Format

You can begin transmission with a sign-on record. Its format is identical to the data record described below, except that the first two bytes contain S0 instead of S1.

3) Data-Record Format

Bytes	*Description*
1,2	S1. (S0 for sign-on record).
3,4	Hexadecimal number of data bytes in record, plus 3.
5-8	Hex address of first data byte in record.
9-10	First data byte in hexadecimal notation.
11-12	Second data byte in hexadecimal notation.
13-nn	Remaining data bytes in hexadecimal notation. Number must equal byte count in digits 3 and 4, less 3.
nn+1,nn+2	Checksum. One's-complement of binary summation of preceding data bytes, plus byte count and address.
nn+3 on	Optional comments, line feed, or carriage return.

4) End-of-File Record

Bytes	*Description*
1,2	S9
3,4	03
5-8	Address
9-10	Checksum. One's-complement of binary summation of address and 03.

5) Example

The following example illustrates a typical set of Motorola data records.

```
S11D00005500550055005500550055005500550055005500550055005500550091
S11D001A55005500550055005500550055005500550055005500550055005500550077
S1110034550055005500550055005500550067
S9030000FC
```

A.2.3.3 Standard Tektronix Hexadecimal Format

1) Description

The standard Tektronix hexadecimal data format consists of data blocks that contain object code expressed as hexadecimal digits followed by a termination block that contains the transfer address.

If errors occur during transmission, the emulator or the host also supplies an abort block with a message indicating the problem.

2) Data Block

A data block consists of up to 255 hexadecimal digits as follows:

Digit	*Description*
1	Slash (/) indicating standard Tektronix hex format.
2-5	Address where object code to be loaded, in high-byte, low-byte format.
6-7	Number of data bytes in block.
8-9	Checksum, modulo 256, of the address and byte count hex digits.
10-68	Two hex digits for each data byte, 30 data bytes maximum.
Last two	Second checksum, modulo 256, of the data-byte digits.

Figure A-2 shows a data block with six bytes to be loaded at >100.

```
/0100060702020202020C
```

Figure A-2. Tektronix Data-Block Example

3) Termination Block

A termination block contains a transfer address instead of a load address, a byte count of zero, and a checksum of the address digits.

Digit	*Description*
1	Slash (/) indicating standard Tektronix format.
2-5	Starting address for execution of code in preceding data blocks in high-byte, low-byte format.
6-7	Always 00 in termination block.
8-9	Checksum, modulo 256, of the execution-address digits.

Figure A-3 shows a termination block for code with transfer address of >100.

```
/10000001
```

Figure A-3. Tektronix Termination-Block Example

4) Abort Block

The standard Tektronix abort block contains two slashes and one of 69 standard messages indicating what happened as follows:

Digit	*Description*
1	Slash indicating standard Tektronix hex format.
2	Slash indicating standard Tektronix format abort record.
3-71	Message.

Figure A-4 shows an abort block.

```
// 5 CONSECUTIVE FAILURES TRANSMISSION ABORTED
```

Figure A-4. Tektronix Abort-Block Example

5) Example

The following example illustrates a typical set of data records transmitted in Tek format:

```
/0100060702020202020C
/0100060702020202020C
/10000001
```

A.2.3.4 TI Object-Record Formats (Standard and Compressed)

1) Description

Tags inform the monitor program what to do with data that follows. Table A-2 lists the tags that the emulator honors and describes what it does with each.

Table A-2. TI Formats and Object Records

TAG	FIELD 1	FIELD 2	FIELD 3	EMU ACTION
		MODULE DEFINITION		
0	PSEG Length	Program ID (8)		Ignored
M	DSEG Length	$DATA	0000	Ignored
M	Blank Common Length	$BLANK	Common #	Ignored
M	CSEG Length	Common Name (6)	Common #	Ignored
M	CBSEG Length	$CBSEG	CBSEG #	Ignored
		ENTRY POINT DEFINITION		
1	Absolute Address			Ignored
2	P-R Address			Ignored
		LOAD ADDRESS		
9	Absolute Address			Processed
A	P-R Address			Processed
S	D-R Address			Processed
P	C-R Address	Common or CBSEG#		Processed
		DATA		
B	Absolute Address			Processed
C	P-R Address			Processed
T	D-R Address			Processed
N	C-R Address	Common or CBSEG#		Processed
*	P-R Address			Processed
		EXTERNAL DEFINITIONS		
6	Absolute Value	Symbol (6)		Ignored
5	P-R Address	Symbol (6)		Ignored
W	D-R/C-R Address	Symbol (6)	Common #	Ignored
		EXTERNAL REFERENCES		
3	P-R Chain Address	Symbol (6)		Error
4	Abs.Chain Address	Symbol (6)		Error
X	D-R/C-R Chain Addr.	Symbol (6)	Common #	Error
E	Symbol Index Number	Absolute Offset		Error
		SYMBOL DEFINITIONS		
G	P-R Address	Symbol (6)		Ignored
H	Absolute Value	Symbol (6)		Ignored
J	D-R/C-R Address	Symbol (6)	Common #	Ignored
		FORCE EXTERNAL LINK		
U	0000	Symbol (6)		Ignored
		SECONDARY EXTERNAL REFERENCE		
V	P-R Address of Chain Entry	Symbol (6)		Error
Y	Abs.Addr. of Ch.	Symbol (6)		Error
Z	D-R/C-R Address of Chain	Symbol (6)	Common #	Error

Table A-2. TI Formats and Object Records (Concluded)

TAG	FIELD 1	FIELD 2	FIELD 3	EMU ACTION
CHECKSUM				
7	Value			Processed
IGNORE CHECKSUM				
8	Any Value			Processed
LOAD BIAS				
D	Absolute Address			Processed
END OF RECORD				
F				Processed
REPEAT COUNT				
R	Value	Repeat Count		Processed
PROGRAM ID				
I	P-R Address	Program ID (8)		Ignored
COBOL SEGMENT REFERENCE				
Q	Record Offset	CBSEG #		Ignored

Common Tag Characters And Operations Performed:

Tag Character 0 followed by two fields.

- The first field contains the number of bytes in the program segment (4 characters).
- The second field contains the program identifier assigned by an assembler IDT directive (8 characters).

Tag Character 7. If tag-character 7 precedes the checksum, the assembler formed an error-detection word from the two's-complement of the binary sum of the ASCII values of each object-record character, from the first tag through tag-character 7, and wrote that checksum into the record.

If Tag Character 8 preceded the checksum field, the program ignores the checksum. This lets a program edit object code, without having to recompute the checksum.

Tag Characters 9 and A are used with load addresses for data that follows. The hexadecimal field that follows the tag contains the address of the subsequent data word. Tag 9 indicates an absolute address; Tag A indicates relocatable code.

Tag Characters B and C indicate data words to be stored in memory. The hexadecimal field contains the data word to be placed in the memory location specified in the preceding load-address field, or in the memory location that follows the last word loaded.

Tag B indicates an absolute data value (instruction word or word containing text characters or absolute constants).

Tag C indicates words containing relocatable addresses.

External Reference Tag Characters 3, 4, X, E, V, Y, and Z indicate an unresolved condition requiring generation of error codes.

Upon finding these tags in an object file, the monitor program reads them, continues the download, then displays EXTERNAL REF FOUND IN OBJECT MODULE upon exiting the HOST Mode. Only external reference tags cause this error-message response. You cannot tell which file contains the tag if you have downloaded more than one file in a session.

Tag Character F indicates end-of-record. This tag can be followed by ASCII characters.

End-Of-Module Separator Record is the last record of an object-code file. The record has a colon (:) in the first character position.

2) Example of Downloaded Data Stream

The following is a TI Standard object-code format with tag characters underlined.

```
0000 8TASK     A0000B000AB0200 B0000BC0008F7EEF
: TASK      021/83 12:32:54          <-- EOM separator record
```

Each character in the printout/display represents four bits (half a byte) of memory.

The same data would appear in TI standard object-file format (first record only) in hexadecimal as follows.

```
3030  3030  3854  4153  4B20  2020  2041  3030
3030  4230  3030  4142  3032  3030  4230  3030
3042  4330  3030  3746  3745  4646
```

In hexadecimal, (first record only) the TI-compressed format would print as follows.

```
0100  0854  4153  4B20  2020  2041  0000  4200
0A42  0200  4200  0042  C000  37F7  EF46
```

A.2.4 Protocols

These formats use the following protocols:

1) ASR
2) Tektronix
3) VAX

An XDS/22 also supports no protocol.

A.2.4.1 ASR Protocol

The ASR protocol causes the emulator to act as a Texas Instruments Model 733 ASR/KSR data terminal. With this configuration, the TMS320C25 monitor program responds to DC1 = ON and DC3 = OFF as well as to RTS.

The ASR protocol has defined download and upload start and end and pass-through characters as shown in Table A-3 A pass-through character causes the XDS to ignore and not process the *next* entered character.

Table A-3. Control Characters

	PROTOCOL	
ACTION	**ASR**	**VAX†**
Download Start	CNTL/R	CNTL/V
Download End	CNTL/S	CNTL/W
Upload Start	CNTL/Q	CNTL/A
Upload End	CNTL/@	CNTL/Z
Pass-through Character	CNTL/P	CNTL/P

A.2.4.2 Tektronix Protocol

The Tektronix hex formats require the Tektronix protocol.

1) Defined Features

This protocol has the following defined features:

Zero ASCII >30 is a positive acknowledgment of data-block reception (ACK).

7 ASCII >37 indicates non-reception of a data block (NAK).

An ASCII EOL character follows either.

2) Upload Procedure

For uploading, the XDS sends a block to the host. If received without error (both block checksums agree with similar calculation done by host), the host sends ASCII 30 and the XDS responds with the next block. If the checksums don't match, the host sends ASCII 37, and the XDS resends the block.

After 5 unsuccessful tries, the host sends an abort block and transmission ends. Transmission also ends (no further ACK's) upon successful reception of a termination block.

3) Download Procedure

For downloading, the procedure is the same except that the XDS checks the received data and sends the ACK or NACK.

4) User-Selected Protocol Features

You must select other protocol features such as download start and end characters, upload start and end characters, and a "pass-through" character by executing the XDS/22 IHC (Initialize Host Control) command and responding to its prompts with your choice of characters.

A pass-through character causes the XDS to ignore and not process the *next* entered character.

A.3 Specifications

Table A-4. Physical Specifications

DEVICE	HEIGHT In(mm)	WIDTH In(mm)	DEPTH In(mm)	WEIGHT Lbs(Kilos)
XDS/22	6.5 (165)	17.5 (445)	16 (407)	40 (18)
Any card	0.5 (13)	10.82 (275)	7.75 (197)	1 (0.45)

Table A-5. Electrical Specifications

	INPUT					OUTPUT		
DEVICE	110 V	220 V	+12V	+5V	-12V	+12V	+5V	-12V
XDS/22	3A	2A				4A	20A	2A
Memory Expansion/ Communications Card -0001†			1A	1.5A	0.5A			
Breakpoint/Trace/ Timing Card				4.5A				
Emulator Card				2A				

†With Memory

Table A-6. Environmental Specifications

Operating Temperature	60°F - 90°F (15°C - 35°C)
Operating Humidity	30 - 80
Maximum Temperature Rise	12°F/ (6.6°C)/Hour
Storage Temperature	-40°F - +185°F (-30°C - 85°C)
Storage Humidity	5 - 90

Pagination

Index

A

B

C

D

E

F

G

H

I

L

M

P

R

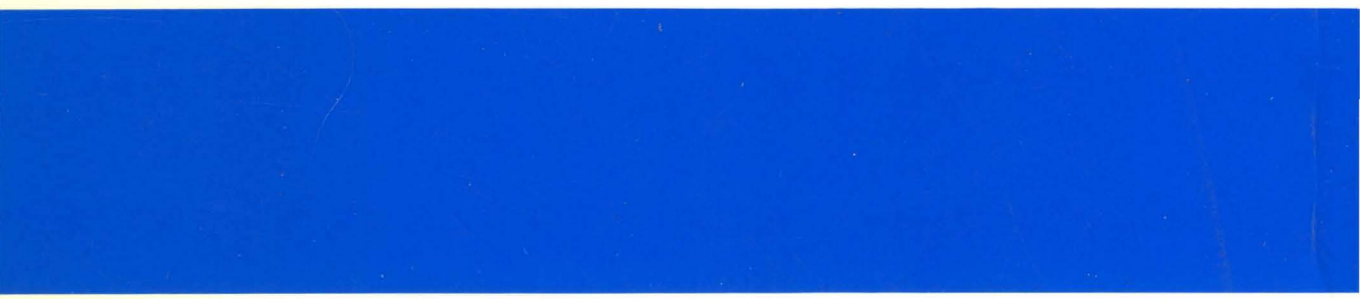
S

T

U

V

X



TEXAS INSTRUMENTS
テキサスインスツルメンツ

SCJ1271

Printed in Japan '88.2