User's Guide

TMS320 Family Simulator

1988

TEXAS INSTRUMENTS

TMS320 Family Simulator

User's Guide

1988

Digital Signal Processor Products

TMS320 Family Simulator User's Guide

Texas Instruments

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to or to discontinue any semiconductor product or service identified in this publication without notice. TI advises its customers to obtain the latest version of the relevant information to verify, before placing orders, that the information being relied upon is current.

TI warrants performance of its semiconductor products to current specifications in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Unless mandated by government requirements, specific testing of all parameters of each device is not necessarily performed.

TI assumes no liability for TI applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Contents

Section *Page*

Illustrations

Figure *Page*

Tables

Table *Page*

Section 1

Introduction

The TMS320 family consists of three generations of digital signal processors. The first generation includes several processors such as the TMS320C10, TMS320C15, TMS320C17, TMS320E15, TMS320E17, and TMS320C14/E14 (CPU only). The TMS32020 and TMS320C25 are the second-generation processors. The TMS320C30 is the third-generation processor. Device user's guides are available from your local sales office.

Four of the first-generation processors mentioned above are new compatible CMOS additions to the line of industry-leading TMS320 digital signal processors (DSP). The TMS320C15 and its EPROM version, the TMS320E15, incorporate a TMS320C10 CPU and feature double the on-chip RAM (from 144 to 256 words) and 2.5 times the on-chip program memory (from 1.5K to 4K words of ROM/EPROM). The TMS320C17 and TMS320E17 also offer this larger on-chip memory space of 256 words of RAM and 4K words of ROM/EPROM, plus two serial ports and a coprocessor interface.

The TMS320 Simulators are software programs that simulate operation of the TMS320 family of high-performance digital signal processors for effective software development. This manual includes information for the TMS320C10 Simulator, TMS32020 Simulator, and the TMS320C25 Simulator. The TMS320C10 Simulator can be used for simulation of all the first-generation processors.

The TMS320 Simulators are currently available on a 1600 BPI magnetic tape for the VAX/VMS[1] operating system and on a 5 1/4-inch floppy disk for the TI PC/MS-DOS[2] and IBM PC/PC-DOS[3] operating systems. The PC configuration requires a minimum of 512K words of RAM for the the second generation Simulators.

A TMS320 hotline is available to assist you with technical questions about TMS320 family products or development tools. The phone number is 713-274-2320.

Topics in this section include:

Section **Page**

[1] VAX and VMS are trademarks of Digital Equipment Corporation.

[2] MS is a trademark of Microsoft, Incorporated.

[3] IBM and PC-DOS are trademarks of International Business Machines Corporation.

1.1 TMS320 Simulator Description

Each of the three TMS320 Simulator software programs simulate TMS320 operation. The simulators allow program verification and monitoring of the state of the TMS320. Simulation speed is on the order of thousands of instructions per second (VAX/VMS) or hundreds of instructions per second (TI/IBM PC runnning MS/PC-DOS).

The simulators use TMS320 object code, produced by the TMS320 Macro Assembler/Link Editor. Input and output files may be associated with the port addresses of the I/O instructions in order to simulate I/O devices connected to the processor. Each interrupt flag can be set periodically at a user-defined interval for simulating an interrupt signal. Before initiating program execution, breakpoints may be defined and the trace mode set.

During program execution, the internal registers and memory of the simulated TMS320 are modified as each instruction is interpreted by the host computer. Execution is suspended when one of the following conditions exists:

1) A breakpoint or error is encountered.

2) A branch to 'self' is detected.

3) Execution is halted.

Once program execution is suspended, the internal registers and both program and data memories can be inspected and/or modified. The trace memory can also be displayed. A record of the simulation session can be maintained in a journal file so that it may be re-executed to regain the same machine state during another simulation session.

1.2 Key Features

These key features highlight simulator flexibility for effective software development:

- Simulation of the TMS320 microprocessors
  - TMS320C10 and other first generation processors
  - TMS32020
  - TMS320C25
- Interrupt generation at user-specified intervals
- File-associated I/O with 8 ports (TMS320C10) or 16 ports (TMS32020 and TMS320C25)
- Programmable breakpoints on:
  - Instruction acquisition
  - Memory reads and writes (data or program)
  - Data patterns on the data-bus or the program-bus
  - Error conditions
- Trace on:
  - Accumulator
  - Program counter
  - Auxiliary registers
- Single-stepping of instructions
- Data and program memory modification and display:
  - Change an entire block at any time
  - Initialize memory before a program is loaded
- Modification and inspection of registers
- Timing analysis relative to clock rate
- Error messages for:
  - Illegal opcodes
  - Invalid data entry
- Execution of commands from a journal file
- Save-states for restarting simulation (TMS320C25)
- File associated serial port for transmission and reception of ASCII data files (TMS320C25)

Note:

The TMS320C15, TMS320E15, TMS320C17, and TMS320E17 each have 256 words of RAM and 4K words of ROM, unlike the other first-generation processors, which have 144 words of RAM and 1.5K of ROM. When using the TMS320C10 Simulator to simulate these four processors, enter the ERAM command at the command prompt. This will expand the RAM from 144 words to 256 words, and the ROM from 1.5K words to 4K words.

1.3 Style and Symbol Conventions

These style and symbol conventions are used to present information clearly and concisely:

- The symbol <CR> indicates that a carriage return (or Enter key) should be pressed.
- Hexadecimal values are preceded by a greater than (>) sign.
- Screen displays are shown in a `special font.`
- User inputs (responses) are underscored.

1.4 How to Use This Manual

This user's guide is a reference for development engineers who have an understanding of TMS320 assembly language and a background knowledge of the TMS320C10, TMS32020, or TMS320C25 microprocessor. See the *TMS320C1x User's Guide* and the *TMS320C10 Assembly Language Programmer's Guide*, or the *TMS320C2x Second-Generation User's Guide* for more information.

The following lists each section and briefly describes the section contents.

- **Section 2 - Installation and Execution Verification.** Installation and execution verification procedures for the VAX/VMS TMS320 Simulator and the IBM PC/PC-DOS and TI PC/MS-DOS TMS320 Simulator. Entering commands and loading a file.

- **Section 3 - Simulator Commands.** Control operations unique to the VAX/VMS and MS/PC-DOS operating systems. A command set summary grouped by function. Individual descriptions of each command.

- **Section 4 - Sample Debugging Sessions.** Debugging examples for the three simulators to demonstrate the finding and correcting of program errors.

- **Appendix A - Simulator Stop Codes.** Definitions of run-time stop codes that can occur during program execution.

Section 2

Installation and Execution Verification

This section describes the simulator installation and verification procedures. The TMS320 Simulator can be installed on the VAX/VMS or the TI/IBM MS/PC-DOS operating systems. Note that the TMS320C10 Simulator requires 256K words of memory, the TMS32020 requires 512K words, and the TMS320C25 requires 640K words (a simulator with limited DATA memory arrays will run within 512K words of memory).

Two methods verify that a working TMS320 Simulator has been installed correctly. Method 1 consists of entering a short series of commands from the keyboard and checking for the desired result. In Method 2, an absolute, tagged, and linked (if necessary) object file is loaded into the simulator via the L (load) command.

The first part of both verification methods is the same. Because the procedures in the verification methods differ according to the device, execution verification for the TMS320C10 and the TMS32020/TMS320C25 are presented in separate subsections.

Topics in this section include:

Section **Page**

Note:

The TMS320C10 simulator program memory can only be loaded with *ASCII tagged object files with an absolute origin.* However, the TMS32020/C25 simulators can be loaded with *Common Object File Format (COFF) files,* as well as ASCII tagged object files with an absolute origin.

2.1 VAX/VMS TMS320 Simulator Installation

In the following pathnames, <userid> refers to the name of a directory; for example, DUA2:[<userid>]. In the examples, MFA0 is the tape drive name and DUA2 is the hard disk drive name. Tape drive and disk drive names may differ. Consult your system manager for the correct device names.

2.1.1 TMS320C10 Simulator

- Restore the simulator from tape to directory.

  1) Place the distribution tape on a tape drive and enter these commands:

     ```
     ALLOC MFA0: <CR>
     MOUNT MFA0:SIM320 <CR>
     ```

     To indicate a successful mount, the screen displays:

     ```
     SIM320 MOUNTED ON MFA0
     ```

  2) Read the SIM320 saveset from the tape:

     ```
     CREATE/DIR <simulator directory>  <CR>
     SET DEFAULT <simulator directory>  <CR>
     BACKUP/VERIFY/LOG MFA0:SIM320.BCK *.*;* <CR>
     ```

     The SIM320 saveset is now copied into your directory, with subdirectory structure maintained.

  3) Dismount and deallocate the tape:

     ```
     DISMOUNT MFA0:  <CR>
     DEALLOCATE MFA0:  <CR>
     ```

- Execute the simulator by entering the following command:

  ```
  SIM  <CR>
  ```

  The system should now be executing the simulator, and the initial simulator screen should be displayed.

2.1.2 TMS32020 and TMS320C25 Simulators

- Restore the simulators from tape to disk.

1) Place the distribution tape on a tape drive and enter these commands at the terminal:

```
ALLOCATE MFA0:  <CR>
MOUNT MFA0:SIM25  <CR>
```

To indicate a successful mount, the screen displays:

```
SIM25 MOUNTED ON MFA0
```

2) To read the SIM25 saveset from the tape, note that the TMS32020 Simulator is included in this saveset, enter:

```
CREATE/DIR <simulator directory>  <CR>
SET DEFAULT <simulator directory>  <CR>
BACKUP/VERIFY/LOG MFA0:SIM25.BCK *.*;* <CR>
```

3) To dismount and deallocate the tape enter:

```
DISMOUNT MFA0:  <CR>
DEALLOCATE MFA0:  <CR>
```

- Execute the TMS32020/C25 Simulator by the following procedure:

1) To make the simulator available for a number of users, the System manager may wish to make a system assignment similar to the following.

```
$ SIM25 :== $[<system-tools-directory>]sim25.exe
```

This system assignment allows the SIM25.EXE file and the associated SCREEN.DAT files to be placed in the *system-tools-directory* in order to properly invoke the simulator.

2) The user may also wish to add the following line to the LOGIN. COM file.

```
 $ DEFINE IPCDIR [<user-tools-directory>]
```

This statement will force the simulator to search for the SCREEN.DAT file in some other user directory if this file cannot be found in the current execution directory.

3) Another option for executing the simulator is to add the following lines in the user's LOGIN.COM file.

```
$ SIM25 :== $[<directory-for-simulator>]sim25.exe
$ DEFINE IPCDIR [<directory-for-simulation>]
```

For this second option, the SIM25.EXE and SCREEN.DAT files must be in the *directory-for-simulation*. If IPCDIR is not defined, then the SCREEN.DAT file must be in the current default directory.

4) Either of these options allow you to begin running the simulator by entering:

```
RUN SIM25    <CR>
```

The system should now be executing the simulator, and the initial simulator screen should be displayed.

The simulator will initially be in the TMS320C25 mode as indicated at the right top of the screen display. To enter the TMS32020 simulator, please refer to the **SIM** command in the Simulator Commands section of this manual.

Note:

Some customers have reported problems running VMS programs that were generated under the VMS operating system earlier than 4.5 version. If the SIM25.EXE does not run, we have provided a way for you to regenerate SIM25 on your VMS system. To rebuildSIM25.EXE, type

```
@LINKSIM
```

We recommend that you rebuild SIM25 **only** if the simulator does not run.

2.2 TI/IBM MS/PC-DOS TMS320 Simulator Installation

These instructions are for hard disk systems and dual floppy drive systems. To install the TMS320C10, TMS32020, or the TMS320C25 Simulator, follow these steps:

- Make a backup diskette of the simulator.
- On **hard disk** systems, copy the simulator onto the hard disk. Enter:

  1) `COPY A:*.* C:*.*/V <CR>`

- For the TMS32020/C25 simulator the user should create (or modify) a CONFIG.SYS file.

  1) If using an IBM PC, this file should contain the lines:

  ```
  DEVICE=ANSI.SYS
  FILES=20
  ```

  2) If using the TI PC, this file should contain the lines:

  ```
  FILES=20
  ```

  After the user has re-booted the PC once, the new CONFIG.SYS file will take effect permanently.

- Execute the simulator.

  1) For TMS320C10 execution, enter:

  `SIM <CR>`

  2) For TMS32020/C25 execution, enter:

  `SIM25 <CR>` for the IBM PC

  `SIM25T <CR>` for the TI PC.

The system should now be executing the simulator,and the initial simulator screen should be displayed.

The simulator will initially be in the TMS320C25 mode as indicated at the right top of the screen display. To enter the TMS32020 simulator, please refer to the **SIM** command in the Simulator Commands section of this manual.

Note:

The PC version of the TMS32020/C25 simulator has the restriction that only 14 files of any kind may be open at any one time. While the users can assign a file to any one of 16 input and output ports, they are restricted to a maximum of 14 total files. Having files open for other purposes (e.g., journal file collection) further restricts the number of total files that are available for port assignment. Increasing the `FILES =` in the CONFIG.SYS file over 20 will not allow the simulator to use more than 20 files simultaneously.

2.3 Command Line Entries (TMS32020/C25)

Once the appropriate system assignments have been made during the installation instructions, several entries can now be made when the simulator is invoked. The format of command line entries is as follows: (The "|" symbol stands for "OR").

```
SIM25 [-a d0-d7:b0-b7] [-c] [-j <filename>] [-m 1|0]
                     [-t <nrows>]
```

Each of these commands is discussed separately below.

1) The **-a d0-d7:b0-b7** command is for PC USE ONLY. This command sets attributes for screen color. The following table and example explains the entry sequence.

Colors	Display Attributes	Bold Attributes
black	d0	b0
blue	d1	b1
green	d2	b2
cyan	d3	b3
red	d4	b4
magenta	d5	b5
yellow	d6	b6
white	d7	b7

Example: When invoking the simulator enter:

```
SIM25 -a d4 -a b7 <CR>
```

When the opening simulator screen appears, the display will be red with bold white headings.

2) The **-c** command turns off the clock simulation run. This command improves the speed of the simulation for runs where clock speed is not relevant.

Example: When invoking the simulator enter:

```
SIM25 -c <CR>
```

Note:

Although the clock counter (CLK) still appears on the display and appears to be working, the clock is no longer accurate and the speed of the simulator is improved.

3) The **-j <filename>** command loads and runs a given journal file.

   Example: When invoking the simulator enter:

   ```
   SIM25 -j TEST.JNL <CR>
   ```

   This command starts the simulator and immediately executes the journal file TEST.JNL.

4) The **-m 0|1** command sets the memory configuration. If a 0 is entered, the first 4K words are configured as internal ROM. An entry of 1 will configure the first 4K words as external ROM.

   Example: When invoking the simulator enter:

   ```
   SIM25 -m 1 <CR>
   ```

   Instead of the next screen prompting the user to configure the memory, this command starts the simulator and configures the first 4K words memory as external.

5) The **-t <nrows>** command sets the length of the trace buffer sample to be taken if Trace mode is to be used.

   Example: When invoking the simulator enter:

   ```
   SIM25 -t 10 <CR>
   ```

   This command starts the simulator and sets the trace buffer to accept only 10 rows of data.

2.4 TMS320C10 Execution Verification

TMS320C10 Simulator execution can be verified by one of two methods. The first part of this verification procedure, described below, is the same for both methods.

1) Log onto the simulator from the host operating system:

   On **VAX/VMS** systems (version 3.7), activate the TMS320C10 Simulator by entering the following command after the system prompt:

   ```
   RUN [<userid.dir>]SIM.EXE   <CR>
   ```

   On **MS/PC-DOS** systems (versions 2.0 and up), create a path with the correct PC path command, then enter:

   ```
   SIM  <CR>
   ```

2) Immediately after logon, all RAM and ROM locations are initialized to 0 (the TMS320C10 ADD instruction).

3) Next, the simulator prompts for the processing mode that is to be simulated: microcomputer or microprocessor mode.

   Microprocessor mode, with program memory addresses >0 to >1535 off-chip, is the default mode. It may be selected by entering a zero or a carriage return after the system prompt, ENTER COMMAND (HELP=<CR>). You will see the following display:

   ```
                (C) Copyright Texas Instruments
                      Incorporated  1984

                  SIMULATION OF THE TMS320C10
                         VERSION # X.X

   0 - MICROPROCESSOR MODE (ADDR 0-1535, OFF CHIP)
   1 - MICROCOMPUTER  MODE (ADDR 0-1535, ON  CHIP)

   ENTER VALUE TO SELECT MODE OF OPERATION
   0 <CR>

   YOU ARE IN THE MICROPROCESSOR MODE (ADDR 0-1535,
   OFF CHIP)

   ENTER COMMAND (HELP=<CR>):
   ```

4) After choosing the processing mode, proceed with verification Method 1 (Section 2.4.1) or Method 2 (Section 2.4.2).

2.4.1 Method 1: Entering Commands

The following example loads the TMS320C10 IN and OUT instructions into program ROM via the ROM (modify/inspect program ROM) command. The simulator reads a single number and outputs that same number. Note that the BIAQ (breakpoint on instruction acquisition) command is used to stop execution. Without the BIAQ command, the simulator would continue to execute all ADD instructions until the end of the ROM locations was reached.

```
ENTER COMMAND (HELP=<CR>):
ROM <CR>

ENTER STARTING ADDRESS (IN HEX)
0 <CR>

    0 =    0
4210 <CR>            *Opcode for the TMS320C10 IN instruction.*
    0 = 4210
+ <CR>               *Move to the next ROM address.*
    1 =    0
4D10 <CR>            *Opcode for the TMS320C10 OUT instruction.*
    1 = 4D10
Q <CR>               *End the ROM modification.*

ENTER COMMAND (HELP=<CR>):
BIAQ <CR>            *Set the end of the simulation.*

BREAK ON INSTRUCTION ACQUISITION
ENTER THE ADDRESS (IN HEX)
3 <CR>

ENTER COMMAND (HELP=<CR>):
R <CR>               *Run the simulation.*
```

The first clock shows that the IN instruction, opcode=4210, is loaded into the simulator. The value >56 was chosen arbitrarily; RAM location >10 is encoded in the IN opcode.

```
>>PC=     0   OPCODE=4210    IN          PREVIOUS PC=      0

          ARP     AR0     AR1    TREG      PREG      ACC      CLK
INTEGER    0       0       0      0         0         0        1
HEX        0       0       0      0         0         0        1

>>STK=    0   0     0     0     DP = 0  INTF= 0  OV = 0
                                BIO= 1  INTM= 0  OVM= 0

ENTER INPUT VALUE (IN HEX) OR "-" TO RETURN TO MAIN
56 <CR>                *Load value into RAM location >10.*
* * * OUTPUT VALUE (IN HEX) IS 56
```

Execution breaks when the simulator attempts to fetch an instruction at location >3.

```
>>PC=     3   OPCODE=   0    ADD          PREVIOUS PC=       2

           ARP      ARO      AR1    TREG       PREG       ACC       CLK
INTEGER     0        0        0      0          0          0         6
HEX         0        0        0      0          0          0         6

>>STK=    0    0     0     0    DP = 0   INTF= 0  OV = 0
                                BIO= 1   INTM= 0  OVM= 0

>>> INSTRUCTION ACQUISITION BREAK POINT #  1 <<<

ENTER COMMAND (HELP=<CR>):
RAM <CR>             *Verify that RAM location >10 contains >56.*

ENTER STARTING ADDRESS (IN HEX)
10 <CR>

    10 =   56
Q <CR>               *Quit the simulation.*

ENTER COMMAND (HELP=<CR>):
```

2.4.2 Method 2: Loading a File

The second method loads an absolute file into the simulator to verify simulator execution.

1) Create, edit, assemble, and link (if necessary) a TMS320C10 source code file on the host system. For more information on writing, assembling, and linking the code, refer to the *TMS320C10 Assembly Language Programmer's Guide* (literature number SPRU002).

   The following example accesses a program called INOUT.MPO, which contains the TMS320C10 IN and OUT instructions.

   The source file for INOUT.ASM is:

```
AORG    0
IN      >0010,2     Read a word from a peripheral on port
                    address 2 into ROM location >0010
                    (opcode=4210).

OUT     >0010,5     Write a word from ROM location >0010
                    to a peripheral on port 5
                    (opcode = 4D10).
```

2) After INOUT.ASM has been assembled, linked (if necessary), and tagged on the host system, it is loaded into the simulator via the L (load) command. *Note that when any file is loaded into the simulator, it is important that the file be an absolute, ASCII tagged object file.* The following display shows the series of prompts produced by the L (load) command.

```
ENTER COMMAND (HELP=<CR>):
```

```
L <CR>

ENTER A NEW OBJECT FILE
INOUT.MPO <CR>

* * * * LOADING PROGRAM "NO$IDT"     * * * *

ENTER COMMAND (HELP=<CR>):
```

3) Once the absolute object file is loaded, set a breakpoint with the BIAQ (breakpoint on instruction acquisition) command and use the R (run) command to run the simulation. The displays are shown below.

```
ENTER COMMAND (HELP=<CR>):
BIAQ <CR>       *Set the end of the simulation.*

BREAK ON INSTRUCTION ACQUISITION
ENTER THE ADDRESS (IN HEX)
3 <CR>

ENTER COMMAND (HELP=<CR>):
R <CR>          *Run the simulation.*
```

The first clock shows that the IN instruction, opcode=4210, is loaded into the simulator. The value >56 was chosen arbitrarily; RAM location >10 is encoded in the IN opcode.

```
>>PC=     0   OPCODE=4210    IN          PREVIOUS PC=    0

            ARP    AR0    AR1    TREG    PREG     ACC   CLK
INTEGER      0      0      0      0       0        0     1
HEX          0      0      0      0       0        0     1

>>STK=    0   0    0    0    DP  = 0  INTF= 0  OV  = 0
                             BIO= 1   INTM= 0  OVM= 0

ENTER INPUT VALUE (IN HEX) OR "-" TO RETURN TO MAIN
56 <CR>         *Load value into RAM location >10.*
* * * OUTPUT VALUE (IN HEX) IS 56
```

Execution breaks when the simulator attempts to fetch an instruction at location >3.

```
>>PC=     3   OPCODE=    0   ADD         PREVIOUS PC=    2

            ARP    AR0    AR1    TREG    PREG     ACC   CLK
INTEGER      0      0      0      0       0        0     6
HEX          0      0      0      0       0        0     6

>>STK=    0   0    0    0    DP  = 0  INTF= 0  OV  = 0
                             BIO= 1   INTM= 0  OVM= 0

>>> INSTRUCTION ACQUISITION BREAK POINT # 1 <<<

ENTER COMMAND (HELP=<CR>):
```

4) Use the RAM (modify/inspect individual data RAM) command to verify that the desired value has been placed in the specified memory location. The display produced by the RAM command is:

```
ENTER COMMAND (HELP=<CR>):
RAM <CR>          *Verify that RAM location >10 contains >56.*

ENTER STARTING ADDRESS (IN HEX)
10 <CR>

  10 =  56
Q <CR>            *Quit the simulation.*
```

2.5 TMS32020/C25 Execution Verification

TMS32020/C25 simulator execution can be verified by one of two methods. The first part of this verification procedure, described below, is the same for both methods.

1) Log onto the simulator from the host operating system:

   On **VAX/VMS** systems (version 3.7), activate the TMS320C25 Simulator by typing the following command after the system prompt:

   ```
   RUN [<userid.dir>]SIMC25  <CR>
   ```

   On **MS/PC-DOS** systems (versions 2.0 and up), memory requirements should be considered. The simulator will operate on a PC with only 512K words of memory, however, we strongly urge users to use systems with 640K words of memory. To execute the simulator enter:

   ```
   SIM25 <CR>
   ```

   The simulator will ask whether an IBM PC or TIPC is in use. Enter:

   0 if using an IBM PC or a "close compatible" of an IBM PC.

   1 if using a Texas Instruments Professional Computer.

   Note that proper functioning of this software has been verified only on the IBM PC/XT, IBM PC/AT, and Texas Instruments Professional computers.

2) Next, the simulator prompts for the memory configuration, selecting first 4K words as internal or external program ROM.

   Selecting internal program ROM is the default, and can be done by entering a zero or a carriage return after the system prompt, COMMAND. The display is:

   ```
                  (C)   COPYRIGHT  TEXAS INSTRUMENTS
                          INCORPORATED 1986
                         TMS320C2x SIMULATOR
                         RELEASE x.x  xx.xxx

   0 : First 4K words INTERNAL program ROM
   1 : First 4K words EXTERNAL program ROM

   Enter the memory configuration (0 or 1):
   <CR>

   First 4k WORDS are mapped on Internal ROM

   COMMAND:
   ```

3) After choosing the memory map configuration, the following initial conditions result:

   - All program ROM locations are initialized to 0 (the TMS320C25 ADD instruction).

- The program counter (PC) should be set to >0. To ensure that program execution begins at the first location in the program, each L (load) command should be followed by the RS (reset) command. This places >0 in the PC; location >0 is the reset vector and should branch to the starting program location. Program ROM (>0 to >1F) is for interrupts and reserved for other system uses. All executable code should start to load from >20.

- The Data Memory Page Pointer (DP) is set to 4.

  a) If on-chip memory block B0 is data RAM, the DP points to data RAM address 512.

  b) If on-chip memory block B0 is program ROM, note that data memory addresses 512 to 767 (DP4 and DP5) do not exist.

- The BIO (I/O branch control) pin is initialized to 1 (branch control increments to next instruction).

- All other registers and flags are initialized per the *TMS320C2x Second-Generation User's Guide.*

If an ST command (display/update register status) is performed after logging on to the simulator, the following display appears:

```
COMMAND: ST <CR>
```

```
PC :0000 ADD >0200          ------------C25 MP-----------------------------
-1 :0000 ADD >0200          |IFR :000000  |STO:0604 ST1 :07F0 BIO 1
-2 :0000 ADD >0200          |IMR :000000  |ARB:0    ARP :0    CNFD
-3 :0000 ADD >0200          |---MMRS------|CRY:0    DP  :04   FO :0
------------------------STACK--|DRR :0000    |FSM:1    INTM:1    OV :0
   AR0:   0000  |SK0: 0000 |DXR :0000    |OVM:1    PM  :1    SXM:1
   AR1:   0000  |SK1: 0000 |TIM :FFFF    |TC :0    TXM :0    XF :1
   AR2:   0000  |SK2: 0000 |PRD :FFFF    |OUTP:0000
   AR3:   0000  |SK3: 0000 |GREG:0000    |RPTC:   0   CLK:     0
   AR4:   0000  |SK4: 0000 |-----------------------------------------
   AR5:   0000  |SK5: 0000 |     TREG:      0000
   AR6:   0000  |SK6: 0000 |     PREG:  00000000
   AR7:   0000  |SK7: 0000 |     ACC :  00000000
--------------------------------------------------------------------------
```

4) Now, proceed with execution verification Method 1 (Section 2.5.1) or Method 2 (Section 2.5.2).

2.5.1 Method 1: Entering Commands

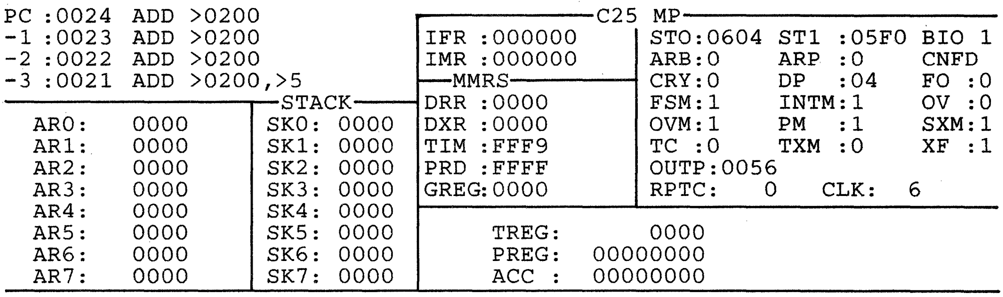
In the following example, the TMS320C25 IN and OUT instructions are loaded into program ROM via the ROM (modify/inspect program ROM) command. The simulator reads a single number and outputs that same number. Note that the BIAQ (breakpoint on instruction acquisition) command is used to stop execution. Without the BIAQ command, the simulator would continue to execute all ADD instructions until the end of the ROM locations was reached.

```
COMMAND:  ROM <CR>
Enter starting address (in Hex)   :20 <CR>

   20 = 0 8210  <CR> *Opcode for the TMS320C25 IN instruction.*
   20 = 8210 +  <CR> *Move to the next ROM address.*
   21 = 0 E510  <CR> *Opcode for the TMS320C25 OUT instruction.*
   21 = E510 Q  <CR> *Quit the ROM modification.*

COMMAND: BIAQ <CR>   *Set the end of the simulation.*

Break on Instruction Acquisition
Enter the address (in Hex): 23 <CR>

COMMAND: PC <CR>

Present value for program counter:   0
Enter a new value for the PC (in Hex) :  20 <CR>

COMMAND:  R <CR>     *Run the simulation.*
```

The first clock shows that the next instruction to be executed by the simulator (PC) is 20. The value >56 was chosen arbitrarily; RAM location >210 is encoded in the IN opcode.

```
PC :0020  ALD >0200                 ------------------C25 MP-------------------------
-1 :0000  ADD >0200                 | IFR :000000    |STO:0604  ST1  :07F0  BIO 1
-2 :0000  ADD >0200                 | IMR :000000    |ARB:0     ARP  :0     CNFD
-3 :0000  ADD >0200                 |----MMRS--------|CRY:1     DP   :04    FO :0
----------------------------STACK---| DRR :0000      |FSM:1     INTM:1      OV :0
  AR0:    0000   |SK0: 0000         | DXR :0000      |OVM:1     PM   :1     SXM:1
  AR1:    0000   |SK1: 0000         | TIM :FFFF      |TC  :0    TXM  :0     XF :1
  AR2:    0000   |SK2: 0000         | PRD :FFFF      |OUTP:0000
  AR3:    0000   |SK3: 0000         | GREG:0000      |RPTC:    0   CLK:   0
  AR4:    0000   |SK4: 0000         |---------------------------------------------------
  AR5:    0000   |SK5: 0000         |     TREG:       0000
  AR6:    0000   |SK6: 0000         |     PREG:   00000000
  AR7:    0000   |SK7: 0000         |     ACC :   00000000
-----------------------------------------------------------------------------------------
   56 <CR>
```

The value >56 has been loaded into RAM location 210. Execution breaks when the simulator attempts to fetch an instruction at location >23. The screen now looks like the following.



```
PC :0024 ADD >0200                                    ─────────────C25 MP──────────────────
-1 :0023 ADD >0200                  IFR :000000      │ ST0:0604  ST1 :05F0  BIO 1
-2 :0022 ADD >0200                  IMR :000000      │ ARB:0     ARP :0     CNFD
-3 :0021 ADD >0200,>5               ──MMRS──────────  │ CRY:0     DP  :04    FO :0
──────────────────────STACK──────── DRR :0000        │ FSM:1     INTM:1     OV :0
  AR0:    0000    │ SK0: 0000       DXR :0000        │ OVM:1     PM  :1     SXM:1
  AR1:    0000    │ SK1: 0000       TIM :FFF9        │ TC :0     TXM :0     XF :1
  AR2:    0000    │ SK2: 0000       PRD :FFFF        │ OUTP:0056
  AR3:    0000    │ SK3: 0000       GREG:0000        │ RPTC:   0    CLK:   6
  AR4:    0000    │ SK4: 0000      ──────────────────────────────────────────────
  AR5:    0000    │ SK5: 0000          TREG:      0000
  AR6:    0000    │ SK6: 0000          PREG:  00000000
  AR7:    0000    │ SK7: 0000          ACC :  00000000
──────────────────────────────────────────────────────────────────────────────
```

```
>> instruction acquisition breakpoint #     1 <<

COMMAND: RAM <CR>            *Verify that RAM location >210 contains >56.*

Enter start DATA address (in Hex) 210 <CR>

 210 = 56 Q <CR>             *Quit the simulation.*
```

2.5.2 Method 2: Loading a File

The second method loads an absolute file into the simulator.

1) Create (or edit), assemble, and link (if necessary) a TMS320C25 source code file on the host system. For more information on writing, assembling, and linking the code, refer to the *TMS32020 User's Guide* and/or the *TMS320C25 User's Guide*.

   The following example accesses a program called INOUT.MPO, which contains the TMS32020/C25 IN and OUT instructions.

   The source file for INOUT.ASM is shown below.

```
AORG     >20
IN       >0010,2     Read a word from a peripheral on port
                     address 2 into ROM location >0010
                     (opcode=8210).

OUT      >0010,5     Write a word from ROM location >0010
                     to a peripheral on port 5
                     (opcode = E510).
```

2) Assemble INOUT.ASM on the host system and load it into the simulator via the L (load) command. *Note that when any file is loaded into the simulator, it is important that the file be an absolute, ASCII tagged object file or a COFF file.* The following display shows the series of prompts produced by the L (load) command.

```
COMMAND:  L <CR>

Enter a new object file     :  INOUT.MPO <CR>

**** LOADING PROGRAM "NO$IDT     " ****

COMMAND:
```

3) Once the absolute object file is loaded, set a breakpoint with the BIAQ (breakpoint on instruction acquisition) command and use the R (run) command to run the simulation. The displays are:

```
COMMAND:  BIAQ <CR>*Set the end of the simulation.*

Break on Instruction Acquisition
Enter the address (in Hex) :  23 <CR>

COMMAND:  PC <CR>
Present value for program counter:   0
Enter a new value for the PC (in Hex) :  20 <CR>

COMMAND:  R <CR>    *Run the simulation.*
```

The first clock shows that the next instruction to be executed by the simulator (PC) is 20. The value >56 was chosen arbitrarily; RAM location >210 is encoded in the IN opcode.

```
PC :0020 ADD >0200          ------------C25 MP-------------------------------
-1 :0000 ADD >0200          |IFR :000000  |ST0:0604 ST1 :07F0  BIO 1
-2 :0000 ADD >0200          |IMR :000000  |ARB:0    ARP :0     CNFD
-3 :0000 ADD >0200          |----MMRS-----|CRY:1    DP  :04    FO :0
-----------------------STACK-|DRR :0000    |FSM:1    INTM:1     OV :0
  AR0:   0000  |SK0: 0000   |DXR :0000    |OVM:1    PM  :1     SXM:1
  AR1:   0000  |SK1: 0000   |TIM :FFFF    |TC :0    TXM :0     XF :1
  AR2:   0000  |SK2: 0000   |PRD :FFFF    |OUTP:0000
  AR3:   0000  |SK3: 0000   |GREG:0000    |RPTC:    0   CLK:   0
  AR4:   0000  |SK4: 0000   |-------------------------------------------
  AR5:   0000  |SK5: 0000   |    TREG:      0000
  AR6:   0000  |SK6: 0000   |    PREG:  00000000
  AR7:   0000  |SK7: 0000   |    ACC :  00000000
-------------------------------------------------------------------------
      56 <CR>
```

The value >56 has been loaded into RAM location >210. Execution breaks when the simulator attempts to fetch an instruction at location >23.

```
PC :0024 ADD >0200           ┌─────────────C25 MP──────────────────────────
-1 :0023 ADD >0200           │IFR :000000    │STO:0604 ST1 :05F0 BIO 1
-2 :0022 ADD >0200           │IMR :000000    │ARB:0    ARP :0    CNFD
-3 :0021 ADD >0200,>5        ├───MMRS────────┤CRY:1    DP  :04   FO :0
─────────────────────┬───STACK──┤DRR :0000      │FSM:1    INTM:1    OV :0
  AR0:    0000       │SK0: 0000 │DXR :0000      │OVM:1    PM  :1    SXM:1
  AR1:    0000       │SK1: 0000 │TIM :FFF9      │TC :0    TXM :0    XF :1
  AR2:    0000       │SK2: 0000 │PRD :FFFF      │OUTP:0056
  AR3:    0000       │SK3: 0000 │GREG:0000      │RPTC:   0    CLK:   6
  AR4:    0000       │SK4: 0000 ├───────────────┴─────────────────────────────
  AR5:    0000       │SK5: 0000 │     TREG:       0000
  AR6:    0000       │SK6: 0000 │     PREG:   00000000
  AR7:    0000       │SK7: 0000 │     ACC :   00000000
─────────────────────┴──────────┴──────────────────────────────────────────
```

```
>> instruction acquisition breakpoint #     1 <<
```

4) Now use the RAM (modify/inspect individual data RAM) command to verify that the desired value has been placed in the specified memory location. The display produced by the RAM command is:

```
COMMAND: RAM <CR>  *Verify that RAM location >210 contains >56.*

Enter start DATA address (in Hex) 210 <CR>

 210 = 56 Q <CR>   *Quit the simulation.*
```

Section 3

Simulator Commands

The TMS320 Simulator commands perform specific simulator functions. This section describes special control operations unique to the VAX/VMS and MS/PC-DOS operating systems. A command set summary that lists commands according to function is provided for easy reference. These commands are also described individually.

Each command page contains a description of the command, an example display with appropriate responses, and an explanation of the display. *Note that although the format of the display varies depending on the simulator you are using, the information remains the same.*

Topics in this section include:

Section **Page**

Note:

Commands must be entered in UPPERCASE LETTERS. If a command is entered in lowercase letters, the simulator will issue an `***INVALID COMMAND***` error message.

Before attempting to execute the simulator, verify that the simulator has been correctly installed by performing either execution verification Method 1 or Method 2 (see Section 2).

3.1 Control Operations

Special control operations are provided to halt command execution in mid-operation and return to the simulator prompt.

VAX/VMS

- <CNTRL-C> halts any run mode process and returns the user to the simulator prompt.
- <ESC> halts any command mode process and returns the user to the simulator prompt.

MS/PC-DOS

- <ESC> halts any process (run or command mode) and returns the user to the simulator prompt.

On the TMS320C10 **VAX/VMS** operating systems, entering a carriage return while executing any command except EX, JF, or L leaves the present value unchanged. If a carriage return is entered while using the JF (journal file) command, a new file called FOR088.DAT is created and stored in the current user directory. On the TMS320C10 MS/PC-DOS and TMS32020/C25 VAX/VMS and MS/PC-DOS an error message will be displayed when a carriage return is entered while using the JF command. Entering a carriage return within the EX (execute) or L (load) commands produces an error message for all simulators. File names are subject to host system constraints. For example, file names may contain up to nine alphanumeric characters (uppercase A–Z and 0–9).

On **MS/PC-DOS** operating systems, entering a carriage return while executing any command leaves the present value unchanged. File names are subject to host system constraints; for example, file names must be up to eight characters long and contain no blanks.

3.2 Command Set and Menu Summary

The TMS320 simulator command set summary in Table 3-1 is arranged according to function, alphabetized within each functional grouping. If a command is not supported by all TMS320 simulators, the simulators that do support the command are noted in parentheses.

Example: EH Execution Help Menu (TMS320C25)

The EH command is supported only by the TMS320C25 according to this example.

The contents of eleven display menus provide the groupings. Four of the help menus, DH, EH, TH, and UTLH, are unique to the TMS320C25. The eleven display menus are:

- Display Main Menu (DM)
- Breakpoint Help Menu (BH)
- Display Help Menu (DH), TMS320C25 only
- Execution Help Menu (EH), TMS320C25 only
- Modify/Inspect Memory Help Menu (MH)
- Modify/Inspect Registers/Flags Help Menu (RH)
- Input/Output Help Menu (IOH)
- Modify/Inspect Status Registers/Pins Help Menu (STH), TMS32020 and TMS320C25
- Trace Help Menu (TH), TMS320C25 only
- Interrupt/Timing Help Menu (TICH), TMS32020 and TMS320C25
- Utilities Help Menu (UTLH), TMS320C25 only

Table 3-1. Simulator Command Set Summary

MAIN MENU COMMANDS	
Command	Function†
DC	Display Variable Value and Format (TMS320C25)
DCL	Enable Digital Command Language (TMS320C25 VAX/VMS)
DM or <CR>	Display Main Menu
DT	Display Trace Buffer
JF	Select Journal File
SIM	Change Simulator Mode (TMS32020, TMS320C25)
ST	Display/Update Register Status
STR	Save Trace Buffer
SW	Switch from Screen to File (TMS32020/C25)
TR	Toggle Trace
Z	Zero Clock Counter
ZRAM	Set RAM Contents to Zero (TMS32020/C25)

HELP COMMANDS	
Command	Function†
BH	Breakpoint Help
DH	Display Controller Help Menu (TMS320C25)
EH	Execution Help Menu (TMS320C25)
IOH	Input/Output Help Menu
MH	Modify/Inspect Memory Help Menu
RH	Modify/Inspect Registers/Flags Help Menu
STH	Modify/Inspect Status Registers/Pins Help Menu (TMS32020/C25)
TH	Trace Help Menu (TMS320C25)
TICH	Interrupt/Timing Help Menu (TMS32020/C25)
UTLH	Utilities Help Menu (TMS32020/C25)

EXECUTION COMMANDS	
Command	Function†
C	Continue Simulation
ERAM	Expand RAM and ROM (TMS320C10)
EX	Execute Commands from Given File
L	Load New Object File
LC	Load New COFF File
NB	Set Number of Instructions Until Break (TMS320C10, TMS320C25)
Q	Quit Simulation
R	Run Simulation
RS	Reset Simulator
SS	Perform Single-Step Execution

† If a command pertains to one or two processors only, this is indicated in parentheses.

Table 3-1. Simulator Command Set Summary (Continued)

BREAKPOINT COMMANDS	
Command	Function†
BDP	Breakpoint on Data Pattern When Read/Write from/to Data RAM
BDR	Breakpoint on Data RAM Read
BDRW	Breakpoint on Data RAM Read and Write
BDW	Breakpoint on Data RAM Write
BER	Breakpoint on an Error Condition
BIAQ	Breakpoint on Instruction Acquisition
BPP	Breakpoint on Data Pattern When Read from Program ROM
BPR	Breakpoint on Program ROM Read
DB	Display Breakpoints
RB	Remove Breakpoint

INPUT/OUTPUT COMMANDS	
Command	Function†
LF	List Files Assigned to Ports (TMS320C10)
LI	List Files Assigned to Input Ports (TMS32020/C25)
LO	List Files Assigned to Output Ports (TMS32020/C25)
RCV	Assign RCV Channel to File (TMS320C25)
RCVC	Close the RCV Channel File
RSI	Reset Selected Input Port File
SI	Select Input Port File
SO	Select Output Port File
XMT	Assign XMT Channel to File (TMS320C25)
XMTC	Close the XMT Channel File (TMS320C25)

INTERRUPT/TIMING COMMANDS	
Command	Function†
DWAIT	Specify Wait Cycles for External Data Memory (TMS32020, TMS320C25)
IOWAIT	Specify Wait Cycles for External I/O Memory (TMS32020, TMS320C25)
PWAIT	Specify Wait Cycles for External Program Memory (TMS32020, TMS320C25)
TIC	Specify Number of Clock Tics until Next Interrupt (TMS320C10)
TIC0–TIC2	Specify Number of Clock Tics (TIC0–TIC2) until Interrupt (TMS32020, TMS320C25)
ZTIC	Disable Tic Commands

† If a command pertains to one or two processors only, this is indicated in parentheses.

Table 3-1. Simulator Command Set Summary (Continued)

MODIFY/INSPECT MEMORY COMMANDS	
Command	**Function†**
LRAM	Load RAM Data from an External File (TMS320C25)
POP	Restore Simulator State (TMS320C25)
PUSH	Save Simulator State (TMS320C25)
RAM	Modify/Inspect Individual Data RAM
RAMH	Display Data RAM in Hexidecimal
RAMI	Display Data RAM in Integer
ROM	Modify/Inspect Individual Program ROM
ROMH	Display Program ROM in Hexidecimal
ROMI	Display Program ROM in Integer
SGN	Generate Test Data (TMS320C25 VAX/VMS)
SRAM	Store RAM Data to an External File (TMS320C25)
VIEW	Display Stack Contents (TMS320C25)

MODIFY/INSPECT REGISTERS/FLAGS COMMANDS	
Command	**Function†**
ACC	Modify/Inspect Accumulator
AR	Modify/Inspect Auxiliary Registers
BIO	Modify/Inspect I/O Branch Control
CC	Modify/Inspect Clock Counter
FSM	Modify/Inspect Frame Sync Mode (TMS320C25)
INTF	Modify/Inspect Interrupt Flag Register (TMS320C10)
INTFS	Modify/Inspect Interrupt Flags (TMS32020, TMS320C25)
INTM	Modify/Inspect Interrupt Mode Register
INTMS	Modify/Inspect Interrupt Masks (TMS32020, TMS320C25)
P	Modify/Inspect P Register
PC	Modify/Inspect Program Counter
RPTC	Modify/Inspect Repeat Instruction Counter (TMS32020, TMS320C25)
SK	Modify/Inspect Stack
T	Modify/Inspect T Register
TINT	Modify/Inspect Timer Interrupt Flag (TMS32020, TMS320C25)
TINTM	Modify/Inspect Timer Interrupt Mask (TMS32020, TMS320C25)

† If a command pertains to one or two processors only, this is indicated in parentheses.

Table 3-1. Simulator Command Set Summary (Concluded)

MODIFY/INSPECT STATUS REGISTERS/PINS COMMANDS	
Command	Function†
ARB	Modify Auxiliary Register Pointer Buffer (TMS32020, TMS320C25)
ARP	Modify/Inspect Auxiliary Register Pointer
CNF	Modify/Inspect RAM Configuration Control Bit (TMS32020, TMS320C25)
CY	Modify/Inspect Carry Bit (TMS320C25)
DP	Modify/Inspect Data Memory Page Pointer
FO	Modify/Inspect Format Bit (TMS32020, TMS320C25)
HM	Modify/Inspect Hold Mode Bit (TMS320C25)
INTM	Modify/Inspect Interrupt Mode Register
OV	Modify/Inspect Overflow Flag
OVM	Modify/Inspect Overflow Mode Register
PM	Modify/Inspect Product Shift Mode (TMS32020, TMS320C25)
RINTM	Modify/Inspect RINTM Bit (TMS320C25)
SXM	Modify/Inspect Sign-Extension Mode Bit (TMS32020, TMS320C25)
TC	Modify/Inspect Test/Control Flag Bit (TMS32020, TMS320C25)
TXM	Modify/Inspect Transmit Mode Bit (TMS32020, TMS320C25)
XF	Modify/Inspect XF Pin (TMS32020, TMS320C25)
XINTM	Modify/Inspect XINTM Bit (TMS320C25)

† If a command pertains to one or two processors only, this is indicated in parentheses.

3.3 Individual Command Descriptions

This section describes each simulator command, presented in alphabetical order. If a command is not supported by all TMS320 simulators, the simulators that do support the command are noted in parentheses.

Example: RCVC Close RCV Channel File (TMS320C25)

The RCVC command is supported only by the TMS320C25 according to this example.

A description, example display, and explanation of the display are given for each command. User responses are underlined. Differences in particular command functions for the TMS320C10, TMS32020, or TMS320C25 are indicated in separate examples. Depending on the simulator you are using, your screen displays may vary from the example displays. The information, though, remains the same.

Modify/Inspect Accumulator ACC

Description ACC allows you to inspect and change the accumulator value. After the present accumulator value is displayed, a new value can be entered. Entering a carriage return leaves the value unchanged. ACC is listed in the RH menu.

Example 1

```
ENTER COMMAND (HELP=<CR>):
ACC <CR>

PRESENT ACCUMULATOR VALUE
>        0
ENTER NEW VALUE (IN HEX)
10 <CR>

ENTER COMMAND (HELP=<CR>):
```

To verify the modification, repeat the ACC command or enter the ST command to display/update the value in the accumulator.

Description AR allows you to inspect and change auxiliary register values. After prompting for the auxiliary register number, the register is displayed for inspection and/or modification. AR is in the RH menu.

Example This display appears for the TMS320C10.

```
ENTER COMMAND (HELP=<CR>):
AR <CR>

ENTER THE AUXILIARY REGISTER NUMBER (0 - 1) OR ENTER "-"
TO TERMINATE
1 <CR>
AR1 = >0
ENTER NEW VALUE (IN HEX)
7 <CR>
ENTER THE AUXILIARY REGISTER NUMBER (0 - 1) OR ENTER "-"
TO TERMINATE
- <CR>

ENTER COMMAND (HELP=<CR>):
```

Example This display appears for the TMS32020/C25.

```
            COMMAND:  AR <CR>

Enter the aux-reg number (0-7) or enter "-" to terminate : 1 <CR>
AR          1:    0
Enter new value (in Hex) : 7 <CR>
Enter the aux-reg number (0-7) or enter "-" to terminate : - <CR>

            COMMAND:
```

To verify the modification, repeat the AR command or enter the ST command to update the values in the AR registers.

Note:

Although the screen for the TMS32020 simulator shows values for 8 auxiliary registers, remember the TMS32020 simulator has only 5 auxiliary registers. Therefore, although the prompt asks for a value between 0 and 7, only a number between 0 and 4 will have an effect on the simulator.

Modify/Inspect Auxiliary Register Pointer Buffer (TMS32020/C25) ARB

Description ARB displays and allows modification of the auxiliary register pointer buffer. ARB is listed in the TMS32020 and TMS320C25 Simulator STH menu.

Example 1 This display appears for the simulator.

```
COMMAND:  ARB <CR>

Present value of AR-Reg pointer buffer :          0
Enter new value (0 - 7) :  2 <CR>

COMMAND:
```

Note that when the ARB value is changed, the Auxiliary Register Pointer (ARP) is also changed to the new value.

To verify the modification, repeat the ARP command or enter the ST command. The arrow (==>) now points at the updated auxiliary register value.

ARP Modify/Inspect Auxiliary Register Pointer

Description ARP allows you to inspect and change the auxiliary register pointer value. The ARP command is in the STH menu of all of the simulators.

The TMS320C10 has two auxiliary registers, AR0 and AR1, so ARP can be set to 0 or 1 for the TMS320C10. The TMS32020 has five auxiliary registers, AR0–AR4, so ARP can be set to 0–4 for the TMS32020. The TMS320C25 has eight auxiliary registers, AR0–AR7, so ARP can be set to 0–7 for the TMS320C25.

Table 3-2. Auxiliary Register Pointer Values

ARP	Result	TMS320C10	TMS32020	TMS320C25
0	AR0 – auxiliary register 0	Y	Y	Y
1	AR1 – auxiliary register 1	Y	Y	Y
2	AR2 – auxiliary register 2	N/A	Y	Y
3	AR3 – auxiliary register 3	N/A	Y	Y
4	AR4 – auxiliary register 4	N/A	Y	Y
5	AR5 – auxiliary register 5	N/A	N/A	Y
6	AR6 – auxiliary register 6	N/A	N/A	Y
7	AR7 – auxiliary register 7	N/A	N/A	Y

Example This display appears for the TMS320C10.

```
ENTER COMMAND (HELP=<CR>):
ARP <CR>

PRESENT VALUE OF THE AUXILIARY REGISTER POINTER
1
ENTER NEW VALUE (0,1)
0 <CR>

ENTER COMMAND (HELP=<CR>):
```

Example This display appears for the TMS32020/C25

```
Command ARP <CR>

Present value of the AR-Reg pointer buffer : 1
Enter new value (0-7): 0 <CR>

Command
```

To verify the modification, note the arrow movement to the left of the auxiliary register display.

Note:

Although the screen for the TMS32020 simulator shows values for 8 auxiliary registers(0-7), the TMS32020 simulator has only 5 auxiliary registers. Therefore, only a value entered in the first 5 registers (0-4) will have an effect on the simulator.

BDP — Breakpoint on Data Pattern When Read/Write from/to Data RAM

Description

BDP suspends simulator execution when a specified bit pattern is read from or written to data RAM. The system prompts for a pattern of ones, zeros, and Xs. The Xs correspond to the don't care state. Be sure to use Xs, not blanks; when a blank is entered for a bit position, the following bits are also treated as blanks, thus producing a bad bit pattern.

Use the C command to resume simulation. BDP is listed in the BH menu.

Example 1

```
ENTER COMMAND (HELP=<CR>):
BDP <CR>

BREAK ON DATA RAM R/W
ENTER BIT PATTERN OF 16 BITS (0,1,X)
FIRST BIT IS MSB
111100001111XXXX <CR>

THE 16 BITS ENTERED ARE:
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 1  1  1  1  0  0  0  0  1  1  1  1  X  X  X  X

ENTER COMMAND (HELP=<CR>):
```

When the value >F0FX is read from or written to data RAM, the simulator stops execution.

Use the DB command to display the breakpoint and its reference number. To remove the breakpoint, use the RB command.

Breakpoint on Data RAM Read — BDR

Description

BDR suspends simulator execution when a data memory address within a user-specified address range is read. The system prompts for the beginning and ending hexadecimal addresses. The maximum value for any range is >8F for the TMS320C10 Simulator and >FFFF for the TMS32020 and TMS320C25 Simulators.

Use the C command to resume simulation. BDR is listed in the BH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
BDR <CR>

BREAK ON DATA RAM READ
ENTER THE BEGINNING ADDRESS (IN HEX)
5 <CR>
ENTER THE ENDING ADDRESS (IN HEX)
F <CR>

ENTER COMMAND (HELP=<CR>):
```

When a data RAM address within the inclusive range >5 – >F is read, the simulator stops execution after processing the read.

Use the DB command to display the breakpoint and its reference number. To remove the breakpoint, use the RB command.

BDRW Breakpoint on Data RAM Read and Write

Description

BDRW suspends simulator execution when an address within a user-specified address range is read from or written to. The system prompts for the beginning and ending hexadecimal addresses. The maximum value for any range is >8F on the TMS320C10 Simulator and >FFFF on the TMS32020 and TMS320C25 Simulators.

Use the C command to resume simulation. BDRW is listed in the BH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
BDRW <CR>

BREAK ON DATA RAM READ AND WRITE
ENTER THE BEGINNING ADDRESS (IN HEX)
5 <CR>
ENTER THE ENDING ADDRESS (IN HEX)
F <CR>

ENTER COMMAND (HELP=<CR>):
```

When a data RAM address within the inclusive range >5 – >F is read from or written to, the simulator stops execution after processing the read or write.

Use the DB command to display the breakpoint and its reference number. To remove the breakpoint, use the RB command.

Breakpoint on Data RAM Write — BDW

Description

BDW suspends simulator execution when a data RAM address within a specified range is written to. The system prompts for the beginning and ending hexadecimal addresses. The maximum value for any range is >8F on the TMS320C10 Simulator and >FFFF on the TMS32020 and TMS320C25 Simulators.

Use the C command to resume simulation. BDW is listed in the BH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
BDW <CR>

BREAK ON DATA RAM WRITE
ENTER THE BEGINNING ADDRESS (IN HEX)
5 <CR>
ENTER THE ENDING ADDRESS (IN HEX)
F <CR>

ENTER COMMAND (HELP=<CR>):
```

When a data RAM address within the inclusive range >5 – >F is written to, the simulator stops execution after processing the write instruction.

Use the DB command to display the breakpoint and its reference number. To remove the breakpoint, use the RB command.

BER — Breakpoint on an Error Condition

Description BER lists eight breakpoints (shown in the following example) that may be toggled on or off. BER is in the BH menu.

The simulator prompts for the error condition number to be changed. If the error condition is ON, it will be toggled off. If OFF, the condition will be toggled on. Entering a carriage return terminates the BER command.

If the error condition is listed as ON, it causes a break in the simulation. On the TMS320C10 Simulator, all conditions default to OFF except error conditions 5 and 8. Error condition 5 reflects a hardware limitation, and error condition 8 defaults to ON because attempted writes to on-chip ROM are impossible. On the TMS320C25 Simulator, all conditions default to OFF. Use the C command to resume simulation.

Example This display appears for the TMS320C10 Simulator.

```
ENTER COMMAND (HELP=<CR>):
BER <CR>

          BREAKON ERROR CONDITIONS
          ------- ----- ----------

1) STACK OVERFLOW   = OFF     2) STACK UNDERFLOW = OFF
3) AR OVERFLOW      = OFF     4) AR UNDERFLOW    = OFF
5) MPY 8000 X 8000 = ON      6) ACC OVERFLOW    = OFF
7) PROGRAM MEMORY ADDRESS >1535 = OFF
8) ATTEMPTED TBL WRITE INTO CHIP ROM = OFF

ENTER CONDITION # TO BE TOGGLED
4 <CR>

ENTER CONDITION # TO BE TOGGLED
<CR>

ENTER COMMAND (HELP=<CR>):
```

Auxiliary register underflow or a multiply of >8000 by >8000 halts simulator execution. However, if an auxiliary register overflow occurs, the simulator continues executing since the break on AR OVERFLOW is still off.

The DB command does not list breakpoints set by the BER command.

Example This display appears for the TMS32020/C25 Simulator.

```
COMMAND:  BER <CR>

   breakon  ERROR conditions
   -------  ----- ----------

   1) STACK OVERFLOW  = OFF
   2) STACK UNDERFLOW = OFF
   3) OVERFLOW        = OFF

   0) Return to Main

Enter condition # to be toggled  :  3 <CR>

Enter condition # to be toggled  :  0 <CR>

COMMAND:
```

The DB command does not list breakpoints set by the BER command.

Description The BH command displays the available breakpoint commands. A maximum of 20 breakpoints may be assigned at any one time. The BH command is listed in the DM menu.

Example 1 This display appears for the TMS320C10 Simulators.

```
ENTER COMMAND (HELP=<CR>):
BH <CR>

BREAKPOINT COMMANDS ARE:

BDP  = BREAKPOINT ON DATA PATTERN WHEN R/W FROM/TO
DATA RAM
BDR  = BREAKPOINT ON DATA RAM READ
BDRW = BREAKPOINT ON DATA RAM READ AND WRITE
BDW  = BREAKPOINT ON DATA RAM WRITE
BER  = BREAKPOINT ON AN ERROR CONDITION
BIAQ = BREAKPOINT ON INSTRUCTION ACQUISITION
BPP  = BREAKPOINT ON DATA PATTERN WHEN READ FROM
       PROGRAM ROM
BPR  = BREAKPOINT ON PROGRAM ROM READ
DB   = DISPLAY ALL BREAKPOINTS
RB   = REMOVE A BREAKPOINT


ENTER COMMAND (HELP=<CR>):
```

Any command may be entered after the `ENTER COMMAND (HELP=<CR>)` prompt.

Example 2 This display appears for the TMS32020/C25 Simulator.

```
COMMAND:  BH <CR>

BREAKPOINT COMMANDS ARE:

BDP      :bkpt on data pattern when r/w from/to data
         :Ram
BDR:BDW  :bkpt on data RAM READ:WRITE
BDRW     :bkpt on data RAM READ and WRITE
BER      :bkpt on an ERROR condition
BIAQ     :bkpt on Instruction ACQuisition
BPP      :bkpt on Data Pattern when read from Program
         :ROM
BPR      :bkpt on program ROM read
DB       :display all breakpoints
RB       :remove a breakpoint


COMMAND:
```

Any command may be entered after the `COMMAND:` prompt.

Breakpoint on Instruction Acquisition BIAQ

Description BIAQ suspends simulator execution when an instruction is fetched from a given location in program memory. The system prompts for the beginning hexadecimal address. BIAQ is listed in the BH menu.

Example 1

```
ENTER COMMAND (HELP=<CR>):
BIAQ <CR>

BREAK ON INSTRUCTION ACQUISITION
ENTER THE ADDRESS (IN HEX)
10 <CR>

ENTER COMMAND (HELP=<CR>):
```

When an instruction is fetched from program memory location >10, the simulator stops execution after processing the instruction.

Use the DB command to display the breakpoint and its reference number. To remove the breakpoint, use the RB command.

BIO — Modify/Inspect I/O Branch Control

Description

BIO allows you to inspect and change $\overline{\text{BIO}}$, the I/O branch control pin, used to monitor peripheral device status. BIO is useful as an alternative to an interrupt when it is necessary not to disturb time-critical loops. BIO is in the RH menu.

BIO may have one of the values listed in Table 3-3.

Table 3-3. BIO Value and Results

BIO	Result
0	Performs a branch when the BIOZ instruction occurs in the program.
1	Increments PC to the next instruction.

Example

```
ENTER COMMAND (HELP=<CR>:
BIO <CR>

PRESENT VALUE OF THE I/O BRANCH CONTROL
1
ENTER NEW VALUE (0,1)
0 <CR>

ENTER COMMAND (HELP=<CR>):
```

Breakpoint on Data Pattern When Read from Program ROM — BPP

Description BPP halts simulator execution when a specified bit pattern is read from program ROM. The system prompts for a pattern of ones, zeros, and Xs. The Xs correspond to the don't care state. Be sure to use Xs, not blanks; when a blank is entered for a bit position, the following bits are also treated as blanks, thus producing a bad bit pattern.

Use the C command to resume simulation. BPP is listed in the BH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
BPP <CR>

BREAK ON PROGRAM ROM READ
ENTER BIT PATTERN OF 16 BITS (0,1,X)
FIRST BIT IS MSB
010100001XXXXXXX <CR>

THE 16 BITS ENTERED ARE:
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
 0  1  0  1  0  0  0  0  1  X  X  X  X  X  X  X

ENTER COMMAND (HELP=<CR>):
```

When an LST indirect instruction (>50XX) is read from program ROM, the simulator stops execution after processing the instruction.

The simulator only looks at the first word of a two-word instruction (such as a branch instruction).

Use the DB command to verify that the breakpoint has been set. Use the RB command to remove breakpoints.

BPR Breakpoint on Program ROM Read

Description

BPR suspends simulator execution when a program ROM address within an address range is read from. The system prompts for the beginning and ending hexadecimal addresses.

Use the C command to resume simulation. BPR is listed in the BH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
BPR <CR>

BREAK ON PROGRAM ROM READ
ENTER THE BEGINNING ADDRESS (IN HEX)
1 <CR>
ENTER THE ENDING ADDRESS (IN HEX)
F <CR>

ENTER COMMAND (HELP=<CR>):
```

When a program ROM address is read in the inclusive range >1 - >F, the simulator stops execution after processing the instruction.

Use the DB command to verify that the breakpoint has been set. Use the RB command to remove breakpoints.

Continue the Simulation C

Description

C resumes simulation after a break or interruption. The C command is listed in the DM menu. Simulation may be interrupted by:

- Encountering a breakpoint.
- Branching to self.
- Reaching the limit on the number of instructions that may be executed (NB command).
- Entering <CNTRL-C> on VAX/VMS systems.
- Pressing any key on MS/PC-DOS systems running the TMS320C10 Simulator.
- Entering <CNTRL-Y> on MS/PC-DOS systems running the TMS32020/C25 Simulators.

Like the SS command, the C command automatically implements the ST command, displaying the various register values when simulation terminates.

Example

```
ENTER COMMAND (HELP=<CR>):
C <CR>
```

After the simulation is interrupted, the simulator displays the current status of the registers (see the ST command for sample displays).

```
ENTER COMMAND (HELP=<CR>):
```

Description

CC allows you to inspect and change the clock counter value. The clock counter counts the number of clock tics (from CLKOUT, equal to CLKIN ÷4) occurring within the program. When used with single-step execution, the clock counter is useful in debugging time-critical portions of a program. CC is listed in the RH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
CC <CR>

PRESENT VALUE FOR CLOCK COUNTER
>      50
ENTER A NEW VALUE FOR THE CLOCK COUNTER (IN HEX)
10 <CR>

ENTER COMMAND (HELP=<CR>):
```

To verify the modification, repeat the CC command or enter the ST command. Use the Z command to clear the clock counter.

Modify/Inspect RAM Configuration Control Bit (TMS32020/C25) CNF

Description CNF allows you to inspect and change status register ST1's CNF bit (the RAM configuration control bit). This configures the on-chip memory block B0 as either data RAM or program ROM. Selecting block B0 as data RAM is the default; this can be accomplished by entering <0> or a carriage return. CNF is in the TMS32020 and TMS320C25 STH menu.

Example This display appears for the TMS32020/C25.

```
COMMAND:  CNF <CR>

Present value of ram configuration control bit : 0
 0 - Onchip memory block b0 is Data      RAM
 1 - Onchip memory block b0 is Program   ROM
Enter new value :  1  <CR>

COMMAND:
```

CY — Modify/Inspect Carry Bit (TMS320C25)

Description CY allows you to inspect and change the carry bit in the ST1 status register. Entering a carriage return leaves the value unchanged. CY is listed in the STH menu.

Example

```
COMMAND:  CY <CR>

Present value of the CARRY flag:1
Enter new value (0,1)          :0 <CR>

COMMAND:
```

To verify the modification, repeat the CY command or enter the ST command to update the value in the accumulator.

Description DB lists all current breakpoints set by the BDP, BPP, BDR, BDRW, BDW, BIAQ, BPP, and BPR commands. A listing of all breakpoint reference numbers, the commands used to set these breakpoints, their addresses, and their values is displayed. The breakpoint reference number is required to remove a breakpoint with the RB command. DB is in the BH menu.

Example This display appears for the TMS320C10 and TMS32020.

```
ENTER COMMAND (HELP=<CR>):
DB <CR>


   REF#  SET BY   ADDRESS  VALUE
   ----  ---- --  -------  -----

 1      BDP                >111100001111XXXX
 2      BIAQ      >10
 3      BDR       >5 - >F

ENTER COMMAND (HELP=<CR>):
```

This listing indicates that three breakpoints are now set by BDP, BIAQ, and BDR during the current simulation session.

Example This display appears for the TMS320C25.

```
COMMAND:  DB <CR>


 1        BPR         >     1 - >    F
 2        BPR         >     F - >    1


COMMAND:
```

This listing indicates that two breakpoints are now set by BPR during the current simulation session.

Description

DC allows format control of the display register. On VAX/VMS systems, you may choose to use decimal integer, fixed-point, or hexadecimal notation. Each assumes one of 32 binary point positions. Note that displayed registers are the accumulator, auxiliary registers, P register, and T register.

On MS/PC-DOS systems, you may choose to use decimal integer or hexadecimal notation. The machine-state display is automatically updated when DC is entered; the ST (display/update register status) command need not be entered.

To use the DC command, enter <DC> from the command line, or <D> from the single-step line. DC is in the DH and UTLH menus.

Example 1

This display appears for the TMS320C25 running on a VAX/VMS system.

```
COMMAND:  DC <CR>

   Display control manager
   0 : AR0  4 : AR4  8 : T-Reg C : AR(0-7)
   1 : AR1  5 : AR5  9 : P-Reg D : Default
   2 : AR2  6 : AR6  A : ACC   E : Exit
   3 : AR3  7 : AR7

0: Decimal    1 - 31: Fixed point   32: Hex
Select NUMBER of target :  3
Select display mode  0-32 :  1
Select NUMBER of target :  E

COMMAND:
```

Example 2

This display appears for the TMS320C25 running on an MS/PC-DOS system.

```
COMMAND:  DC <CR>


   Display control manager
   0 : AR0  4 : AR4  8 : T-Reg C : AR(0-7)
   1 : AR1  5 : AR5  9 : P-Reg D : Default
   2 : AR2  6 : AR6  A : ACC   E : Exit
   3 : AR3  7 : AR7

0: Decimal    1: Hex
Select NUMBER of target :  3
Select display mode  0 or 1:  1
Select NUMBER of target :  E

COMMAND:
```

Enable Digital Command Language (TMS320C25 VAX/VMS) DCL

Description The DCL command activates the VAX/VMS Digital Command Language (DCL) from the simulator. This command allows you to edit, assemble, link, and load from the simulator. DCL is listed in the TMS320C25 DM menu.

Example 1

```
COMMAND:  DCL <CR>


TO RETURN TO MAIN type "EXIT"
DCL COMMAND=>
DIR *.MPO:
Directory DUA2:[TEST.ASM]

MISC.MPO;2          MISC.MPO;1

Total of 2 files.   DCL COMMAND=>
EXIT

COMMAND:
```

DH — Display Help Menu (TMS320C25)

Description DH displays commands that control notation, and display and update the status registers. DH is located in the TMS320C25 DM menu.

Example This display appears for the TMS320C25.

```
COMMAND:  DH <CR>

   DM,<CR> :  display main menu
   ST      :  status of registers
   DC      :  display controller

COMMAND:
```

Display Main Menu DM

Description The DM command displays the general commands and other help menus. The TMS320C10 Simulator has four other help menus (BH, IOH, MH, and RH) listed in the DM menu. The TMS32020/C25 Simulator has ten other help menus (BH, DH, EH, IOH, MH, RH, STH, TH, TICH, and UTLH). The menus list commands according to function.

Any command from the menus may be executed any time the ENTER COMMAND (HELP=<CR>) prompt appears. Entering a carriage return after this prompt displays the main menu.

Example This display appears for the TMS320C10.

```
ENTER COMMAND (HELP=<CR>):
DM <CR>

AVAILABLE COMMANDS ARE:
   BH       = BREAKPOINT HELP
   DM <CR>  = DISPLAY MAIN MENU
   DT       = DISPLAY THE TRACE BUFFER
   EX       = EXECUTE COMMANDS FROM A GIVEN FILE
   IOH      = I/O HELP
   JF       = SELECT JOURNAL FILE
   L        = LOAD NEW OBJECT FILE
   MH       = MODIFY/INSPECT MEMORY HELP
   NB       = NUMBER OF INSTR TILL BREAK
   Q        = QUIT SIMULATION
   R,C      = RUN OR CONTINUE SIMULATION
   RH       = MODIFY/INSPECT REGISTERS/FLAGS HELP
   RS       = RESET SIMULATOR
   SS       = SINGLE-STEP EXECUTION
   ST       = STATUS OF REGISTERS
   STR      = SAVE THE TRACE BUFFER
   TIC      = NUMBER OF CLOCK TICS TILL INTERRUPT
   TR       = TOGGLE TRACE MODE (ON OR OFF)
   Z        = ZERO CLOCK COUNTER
   ZTIC     = DISABLES THE TIC COMMAND

ENTER COMMAND (HELP=<CR>):
```

Example This display appears for the TMS32020/C25.

```
COMMAND:  DM <CR>

AVAILABLE COMMANDS ARE:
   TH   : Trace help        UTLH: Utilities help
   BH   : Breakpoint help   DCL : Activate DCL
   EH   : Execution help
   MH   : Memory help
   RH   : Registers/Flags help
   DH   : Display help
   IOH  : I/O help
   TICH: Interrupt/Timing help
   STH  : Status register/pin help

COMMAND:
```

Description

DP displays and allows modification of the data memory page pointer. The TMS32020 and TMS320C25 Simulators expect a decimal number from 0 to 511. For the TMS320C10 Simulator, DP may have one of the values listed in Table 3-4. DP is in the STH menu.

Table 3-4. Data Page Pointer Values for the TMS320C10 Simulator

DP	Result
0	Page = 0, words 0–127 are referenced.
1	Page = 1, words 128–143 are referenced.

Example 1

```
ENTER COMMAND (HELP=<CR>):
DP <CR>

PRESENT VALUE OF THE DATA MEMORY PAGE POINTER
1
ENTER NEW VALUE
0 <CR>

ENTER COMMAND (HELP=<CR>):
```

To determine the status of the data memory page pointer, use the ST command or repeat the DP command.

Description DT displays the trace buffer. The trace mode must be set to ON for the trace buffer contents to be displayed (the TR command sets the trace mode). DT is listed in the TMS320C10 Simulator DM menu and in the TMS32020/C25 Simulator TH menu.

Example This example is for the TMS320C10 Simulator.

```
ENTER COMMAND (HELP=<CR>):
DT <CR>

  PC= 1   ACC=  2  AR0= FF AR1=  0
  PC= 2   ACC=  4  AR0= CD AR1=  0
  PC= 3   ACC=  6  AR0= A8 AR1=  0
  PC= 6   ACC=  8  AR0= 10 AR1=  0
  PC= 7   ACC= 10  AR0= 10 AR1= C6

ENTER COMMAND (HELP=<CR>):
```

Example This example is for the TMS32020/C25 Simulator.

```
     COMMAND: DT <CR>

PC=0001 ACC=00000002
        AR0=00FF AR1=0000 AR2=0000 AR3=0000
        AR4=0000 AR5=0000 AR6=0000 AR7=0000
PC=0002 ACC=00000004
        AR0=00CD AR1=0000 AR2=0000 AR3=0000
        AR4=0000 AR5=0000 AR6=0000 AR7=0000
PC=0003 ACC=00000006
        AR0=00A8 AR1=0000 AR2=0000 AR3=0000
        AR4=0000 AR5=0000 AR6=0000 AR7=0000
PC=0006 ACC=00000008
        AR0=0010 AR1=0000 AR2=0000 AR3=0000
        AR4=0000 AR5=0000 AR6=0000 AR7=0000
PC=0007 ACC=00000010
        AR0=0010 AR1=00C6 AR2=0000 AR3=0000
        AR4=0000 AR5=0000 AR6=0000 AR7=0000

     COMMAND:
```

The register values are expressed in hexadecimal. A warning appears at the beginning of the display if the trace exceeds 256 states.

DWAIT — Specify Wait Cycles for External Data Memory (TMS32020/C25)

Description DWAIT selects the number of wait cycles for external data memory response to provide a more flexible and accurate timing analysis. DWAIT is in the TMS32020 and TMS320C25 Simulator TICH menu.

Example 1

```
ENTER COMMAND (HELP=<CR>):
DWAIT <CR>

PRESENT VALUE OF DWAIT IS
0
ENTER NEW VALUE (0 - 99)
2 <CR>

ENTER COMMAND (HELP=<CR>):
```

Execution Help Menu (TMS320C25) EH

Description EH displays commands that control notation, and display and update the status registers. EH is located in the TMS320C25 DM menu.

Example This display appears for the TMS320C25.

```
COMMAND:  EH <CR>

   R/C :   Run or Continue simulation
   SS  :   single step
   EX  :   execute commands from a given file
   L   :   load new object file
   LC  :   load new COFF file
   NB  :   number of instr till break
   RS  :   reset simulator
   SIM :   change simulator mode
   Q   :   quit simulation

COMMAND:
```

ERAM — Expand RAM and ROM (TMS320C10)

Description ERAM expands the size of the RAM and ROM to allow simulation of the TMS320C15, TMS320E15, TMS320C17, and TMS320E17. RAM is expanded from 144 words to 256 words, and ROM is expanded from 1.5K to 4K.

Example Assume that the Simulator has been initialized to be in microcomputer mode.

```
ENTER COMMAND (HELP=<CR>):
ERAM <CR>

YOU ARE IN THE MICROCOMPUTER MODE (ADDR 0-4096, ON
CHIP)
RAM INCREASED TO 256 WORDS

ENTER COMMAND (HELP=<CR>):
```

Execute Commands from Given File EX

Description EX executes simulator commands from a journal file. EX is listed in the TMS320C10 Simulator DM menu and in the TMS32020/C25 Simulator EH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
EX <CR>

ENTER FILE NAME
JOURNAL.TXT <CR>
```

(The commands in the file JOURNAL.TXT are displayed as they are executed.)

```
ENTER COMMAND (HELP=<CR>):
```

<CNTRL-C> may be used while running TMS320C10, TMS32020, and TMS320C25 Simulators on a VAX/VMS system to exit this command and return to the simulator prompt. <CNTRL-Y> may be used while running the TMS320C25 Simulator on an MS/PC-DOS system.

FO Modify/Inspect Format Bit (TMS32020/C25)

Description FO displays and allows modification of the format bit. FO is listed in the TMS32020 and TMS320C25 Simulator STH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
FO <CR>

PRESENT VALUE OF THE FORMAT PIN
0
ENTER NEW VALUE (0 OR 1)
1 <CR>

ENTER COMMAND (HELP=<CR>):
```

Modify/Inspect Frame Sync Mode (TMS320C25) FSM

Description FSM allows you to inspect and change the frame synchronization mode in the ST1 status register. This bit indicates whether the serial port will operate with or without frame sync pulses. FSM is in the STH menu.

Example

```
COMMAND:  FSM <CR>

Present value of frame synch mode bit : 1
Enter new value  :  0 <CR>

COMMAND:
```

HM Modify/Inspect Hold Mode Bit (TMS320C25)

Description HM allows you to inspect and change the hold mode bit in the ST1 status register. HM is in the STH menu.

Example

```
COMMAND:  HM <CR>

Present value of hold mode bit :1
Enter new value   :  0 <CR>

COMMAND:
```

Modify/Inspect Interrupt Flag Register (TMS320C10) INTF

Description

INTF displays and allows modification of the interrupt flag register. INTF is listed in the TMS320C10 Simulator RH menu.

The INTF and INTM (modify/inspect interrupt mode register) commands work together to determine if interrupts have occurred. INTM reflects an internal condition that indicates whether or not an interrupt flag may be serviced. The INTF and INTM value combinations that cause an interrupt are listed below.

INTF	INTM	INTERRUPT
0	0	NO
0	1	NO
1	0	YES
1	1	NO

Example

```
ENTER COMMAND (HELP=<CR>):
INTF <CR>

PRESENT VALUE OF THE INTERRUPT FLAG REGISTER
0
ENTER NEW VALUE (0,1)
1 <CR>

ENTER COMMAND (HELP=<CR>):
```

The interrupt flag register (INTF) value is changed to logic 1. Note that when simulator execution resumes, an interrupt will occur if the interrupt mode register (INTM) equals logic 0.

To check the current status of the INTF register, use the ST command.

INTFS — Modify/Inspect Interrupt Flags (TMS32020/C25)

Description INTFS allows you to inspect and change one of the three interrupt flags (0–2). First, a prompt appears for the interrupt flag number, and then the flag can be viewed or changed. INTFS is in the TMS32020 and TMS320C25 Simulator RH menu.

The INTFS and INTMS value combinations that cause an interrupt are listed below. These combinations refer to the corresponding INTFS and INTMS values. For example, INTFS(1) = 1 and INTMS(2) = 1 would not cause an interrupt; however, INTFS(1) = 1 and INTMS(1) = 1 would cause an interrupt.

INTFS (0-2)	INTMS (0-2)	INTERRUPT
0	0	NO
0	1	NO
1	0	NO
1	1	YES

Example This display appears for the TMS32020/C25.

```
          COMMAND:  INTFS <CR>

  Enter the IRT flg number (0 - 2) or enter "-" to terminate : 1 <CR>
  INTF1 = 0
  Enter new value (0 or 1) :  1 <CR>
  Enter the IRT flg number (0 - 2) or enter "-" to terminate : - <CR>

          COMMAND:
```

Description

INTM displays and allows modification of the interrupt flag mode register. INTM is found in the TMS320C10 Simulator RH menu and in the TMS32020 and TMS320C25 Simulator STH menu.

The INTM and INTF (modify/inspect interrupt flag register) commands work together to determine if interrupts have occurred. INTM reflects an internal condition that indicates whether or not an interrupt flag may be serviced. The INTF and INTM value combinations that cause an interrupt are listed below.

INTF	INTM	INTERRUPT
0	0	NO
0	1	NO
1	0	YES
1	1	NO

For the TMS32020 and TMS320C25 Simulators, INTM may have one of the values listed in Table 3-5.

Table 3-5. TMS32020 and TMS320C25 Interrupt Mode Register Values

INTM	Result
0	Enables all unmasked interrupts.
1	Disables all maskable interrupts.

Example

```
ENTER COMMAND (HELP=<CR>):
INTM <CR>

PRESENT VALUE OF THE INTERRUPT MODE REGISTER
0
ENTER NEW VALUE (0,1)
1 <CR>

ENTER COMMAND (HELP=<CR>):
```

The interrupt mode register value (INTM) is changed to logic 1. Note that when INTM = 1, no interrupts may occur regardless of the the interrupt flag register value.

To display/update the current status of the INTM register, use the ST command or repeat the INTM command.

INTMS — Modify/Inspect Interrupt Mask (TMS32020/C25)

Description

INTMS displays and allows modification of one of the three interrupt masks (0–2). The simulator first prompts for the interrupt mask number. Then, the mask can be changed. INTMS is listed in the TMS32020 and TMS320C25 Simulator RH menu.

The INTFS and INTMS value combinations that cause an interrupt are listed below. These combinations refer to the corresponding INTFS and INTMS values. For example, INTFS(1) = 1 and INTMS(2) = 1 would not cause an interrupt; however, INTFS(1) = 1 and INTMS(1) = 1 would cause an interrupt.

INTFS (0-2)	INTMS (0-2)	INTERRUPT
0	0	NO
0	1	NO
1	0	NO
1	1	YES

Example

This display appears for the TMS32020/C25.

```
        COMMAND:  INTMS <CR>

Enter the IRT mask number (0 - 2) or enter "-" to terminate : 1 <CR>

INTM1 = 0
Enter new value (0 or 1) 1 <CR>
Enter the IRT mask number (0 - 2) or enter "-" to terminate : - <CR>

        COMMAND:
```

Input/Output Help Menu IOH

Description The IOH menu displays all available input and output commands. IOH is listed in the DM menu.

Example This display appears for the TMS320C10.

```
ENTER COMMAND (HELP=<CR>):
IOH <CR>

IO HELP COMMANDS ARE:
   LF  = LIST OF FILES ASSIGNED TO PORTS
   SI  = SELECT INPUT PORT FILE
   RSI= RESET SELECTED INPUT PORT FILE
   SO  = SELECT OUTPUT PORT FILE

ENTER COMMAND (HELP=<CR>):
```

Example This display appears for the TMS32020/C25.

```
COMMAND:  IOH <CR>

   LI   : List of the files assigned to input  ports
   LO   : List of the files assigned to output ports
   RSI  : Reset selected input port file
   SI   : Select input port file
   SO   : Select output port file
   RCV  : Assign serial port RECEIVE  channel
   XMT  : Assign serial port TRANSMIT channel
   RCVC : Close serial port RECEIVE  channel
   XMTC : Close serial port TRANSMIT channel

COMMAND:
```

Any command from any menu may be entered after the ENTER COMMAND (HELP=<CR>) prompt.

Note:

The RCV, XMT, RCVC, and XMTC commands are only for the TMS320C25 simulator.

IOWAIT — Specify Wait Cycles for External I/O Memory (TMS32020/C25)

Description

IOWAIT selects the number of wait cycles for external input/output memory response to provide a more flexible and accurate timing analysis. IOWAIT is listed in the TMS32020 and TMS320C25 Simulator TICH menu.

Example 1

```
ENTER COMMAND (HELP=<CR>):
IOWAIT <CR>

PRESENT VALUE OF IOWAIT IS
0
ENTER NEW VALUE (0 - 99)
2 <CR>

ENTER COMMAND (HELP=<CR>):
```

Select Journal File JF

Description

The JF command creates, names, and saves a journal file. A journal file acts as the "keeper" of a session's commands. After executing the JF command, every command within a session is recorded in the journal file until the Q (quit) command is entered. A journal file (see Section 3) is useful when testing requires the same processes to be repeated.

If a journal file already exists, entering a carriage return after the prompt for a new file name leaves the file unchanged.

If a journal file does not exist and a carriage return is entered after the prompt for a file name, the TMS320C10 VAX/VMS simulator creates a default file called FOR08.DAT. The TMS320C10 MS/PC-DOS Simulator and the TMS32020/C25 VAX/VMS and MS/PC-DOS Simulators will display an error message.

Example

```
ENTER COMMAND (HELP=<CR>):
JF <CR>

A JOURNAL FILE HAS NOT BEEN CREATED
ENTER FILE NAME
JOURNAL.TXT <CR>

ENTER COMMAND (HELP=<CR>):
```

Example

This display appears when a journal file has been accessed in the current session. If a carriage return is entered after the prompt for the new file name, the simulator appends commands to the current journal file.

```
ENTER COMMAND (HELP=<CR>):
JF <CR>

JOURNAL FILE = JOURNAL

ENTER NEW FILE NAME
NEWJOU.TXT <CR>

ENTER COMMAND (HELP=<CR>):
```

When the JF command is entered, the rest of the simulator session is recorded in the journal file until simulation is halted with a Q (quit) command.

Example

Data RAM locations >0 to >5 are to be loaded with the value >55, and the source program modified at least three times. To avoid loading the same data RAM locations each time a modification is made, a journal file is created using the JF command. The following display shows how the journal file is created.

```
ENTER COMMAND (HELP=<CR>):
JF <CR>

A JOURNAL FILE HAS NOT BEEN CREATED
ENTER FILE NAME
```

```
DATA.TEN <CR>

ENTER COMMAND (HELP=<CR>):
```

From this point on, all commands entered until a Q (quit) is issued are recorded in the journal file DATA.TEN. To load data RAM locations >0 to >5 with >55, the RAM (modify/inspect individual program RAM) command is used.

```
ENTER COMMAND (HELP=<CR>):
RAM <CR>

ENTER STARTING ADDRESS (IN HEX)
0 <CR>
     0 =   0
55 <CR>                         *Change the value to 55.*
     0 =  55
+ <CR>                          *Move to the next RAM address.*
     1 =   0
55 <CR>                         *Change the value to 55.*
     1 =  55
+ <CR>                          *Move to the next RAM address.*
     2 =   0
55 <CR>                         *Change the value to 55.*
     2 =  55
<CR>                            *Repeat the + option.*
     3 =   0
55 <CR>                         *Change the value to 55.*
     3 =  55
<CR>                            *Repeat the + option.*
     4 =   0
55 <CR>                         *Change the value to 55.*
     4 =  55
<CR>                            *Repeat the + option.*
     5 =   0
55 <CR>                         *Change the value to 55.*
     5 =  55
Q <CR>                          *Quit the RAM command.*

ENTER COMMAND (HELP=<CR>):
Q <CR>                          *End the session and create the journal file.*
```

The next time these data RAM locations and values are required, simply execute the journal file DATA.TEN with the EX (execute) command, as shown below.

```
ENTER COMMAND (HELP=<CR>):
EX <CR>

ENTER FILE NAME
DATA.TEN <CR>

ENTER COMMAND (HELP=<CR>):
```

Note that the simulation commands in the sample file DATA.TEN are displayed as they are executed.

Journal files can be a powerful tool for improving the quality of a simulation session. They can be used to make I/O assignments, set sequences of breakpoints, set up operating conditions, and so forth. Two known limitations exist and should be avoided.

1) Nesting executions of journal files. For example, journal file A.TXT performs a few functions and executes journal file B.TXT; when A.TXT is executed, an error occurs because B.TXT is executed within A.TXT. (This is not an error on the MS/PC-DOS version.)

2) Responding with the existing journal file name when prompted for the new journal file name. (This is not an error on the MS/PC-DOS version; however, the file will be overwritten from the point where the journal file name is specified.)

On a VAX system, both of these situations halt the simulation session, returning to the host system prompt.

L Load New Object File

Description

L loads an object file for the simulation. Any file that is loaded into the simulator must be an absolute, tagged, and linked (if necessary) object file.

Memory is not automatically initialized prior to loading an object file. If a file is loaded that does not overwrite the entire previous contents of memory, a portion of the previous object file will remain in memory.

Example 1

```
ENTER COMMAND (HELP=<CR>):
L <CR>

ENTER A NEW OBJECT FILE
NAME.OBJ <CR>

* * * * LOADING PROGRAM "NAME.OBJ" * * * *

ENTER COMMAND (HELP=<CR>):
```

The new object file NAME.OBJ is loaded into program ROM. Future simulations will access the program that is in the file NAME.OBJ.

Load New COFF Files (TMS32020/C25) LC

Description LC loads a Common Object File Format (COFF) file for the simulator.

Memory is not automatically initialized prior to loading a COFF file. If a file is loaded that does not overwrite the entire previous contents of memory, a portion of the previous file will remain in memory.

Example

```
Command:  LC <CR>

Enter a new COFF object file:  COFF.OBJ <CR>

* * * * LOADING PROGRAM "COFF.OBJ" * * * *

Command:
```

The new COFF object file COFF.OBJ is loaded into program ROM. Future simulations will access the program that is in the file COFF.OBJ.

LF List Files Assigned to Ports (TMS320C10)

Description LF lists the input and output files associated with the eight input and eight output ports. LF is listed in the TMS320C10 Simulator IOH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
LF <CR>

Input Port #     File Name
----- ---- -    ---- ----

0                NONE
1                NONE
2                NONE
3                NAME.INP
4                NONE
5                NONE
6                NONE
7                NONE

Output Port #    File Name
------ ---- -    ---- ----

0                NONE
1                NONE
2                NONE
3                NONE
4                NONE
5                NAME.OUT
6                NONE
7                NONE

ENTER COMMAND (HELP=<CR>):
```

All of the ports and their associated files are displayed. The ports that have associated files are input port 3 and output port 5.

List Files Assigned to Input Ports (TMS32020/C25) LI

Description LI lists the files associated with the input ports (0–15). Enter <CNTRL-S> to interrupt and view the file list display. LI is listed in the TMS32020 and TMS320C25 Simulator IOH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
LI <CR>

Input Port #     File Name
----- ---- -     ---- ----

0                NONE
1                NONE
2                NONE
3                NAME.INP
4                NONE
5                NONE
6                NONE
7                NONE
8                NONE
9                NONE
10               NONE
11               NONE
12               NONE
13               NONE
14               NONE
15               NONE

ENTER COMMAND (HELP=<CR>):
```

All of the input ports and their associated files are displayed. In this example, input port 3 is the only port associated with a file.

LO List Files Assigned to Output Ports (TMS32020/C25)

Description LO lists the files associated with the output ports (0–15). Enter <CNTRL-S> to interrupt and view the file list display. LO is located in the TMS32020 and TMS320C25 Simulator IOH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
LO <CR>


Output Port #    File Name
------ ---- -    ---- ----

0                NONE
1                NONE
2                NONE
3                NONE
4                NONE
5                NAME.OUT
6                NONE
7                NONE
8                NONE
9                NONE
10               NONE
11               NONE
12               NONE
13               NONE
14               NONE
15               NONE

ENTER COMMAND (HELP=<CR>):
```

All of the output ports and their associated files are displayed. In this example, output port 5 is the only port associated with a file.

Load RAM Data from External File (TMS320C25) — LRAM

Description

This command loads RAM data from a text file. It prompts for the input filename, starting address, and ending address in hexadecimal format, which will be loaded from a text file.

You may create a file using the SRAM command, which stores the RAM data to an external file. The LRAM command loads the stored file into simulator memory. For this reason, the file you load using the LRAM command must have the same format as the file created using the SRAM command. LRAM is in the UTLH menu.

Example

```
COMMAND:  LRAM <CR>

ENTER FILE NAME :  TEXT1.TXT
Enter STARTING address (in Hex) :  10
Enter END      address (in Hex) :  16
```

To show the results of the LRAM command, the RAMH command is executed below.

```
COMMAND:  RAMH <CR>

Enter start DATA address (in Hex) :  10
 10 19728 19731 19733 19747 19781 19781 19796     0
 18    20     0     0     0     0     0     0     0
 20     0     0     0     0     0     0     0     0
 28     0     0     0     0     0     0     0     0
 30     0     0     0     0     0     0     0     0
 38     0     0     0     0     0     0     0     0
 40     0     0     0     0     0     0     0     0
 48     0     0     0     0     0     0     0     0

COMMAND:
```

In this display, the 19728 in address location >10 is a hexadecimal number, as are the other values displayed.

MH — Modify/Inspect Memory Help Menu

Description The MH command displays the commands that display/modify memory locations. MH is located in the DM menu.

Example This display appears for the TMS320C10.

```
ENTER COMMAND (HELP=<CR>):
MH <CR>

MODIFY AND INSPECT MEMORY COMMANDS ARE:
   RAM    = MODIFY/INSPECT INDIVIDUAL DATA RAM LOCATIONS
   ROM    = MODIFY/INSPECT INDIVIDUAL PROGRAM ROM LOCATIONS
   RAMH   = DISPLAY DATA RAM IN HEX
   RAMI   = DISPLAY DATA RAM IN INTEGER
   ROMH   = DISPLAY PROGRAM ROM IN HEX
   ROMI   = DISPLAY PROGRAM ROM IN INTEGER

ENTER COMMAND (HELP=<CR>):
```

Any command from any menu may be entered after the ENTER COMMAND (HELP=<CR>) prompt.

Example This display appears for the TMS32020/C25.

```
COMMAND:  MH <CR>

         MODIFY AND INSPECT MEMORY COMMANDS ARE:
   RAM   : modify individual data RAM locations
   ROM   : modify individual program ROM locations
   RAMH  : display data RAM in hex
   RAMI  : display data RAM in integer
   ROMH  : display program ROM in hex
   ROMI  : display program ROM in integer
   ZRAM  : zero fill RAM(#6-#FFFF)

COMMAND:
```

Any command from any menu may be entered after the COMMAND prompt.

Number of Instructions Until Break (TMS320C10/C25) NB

Description NB suspends execution after a specified number of instructions have been executed. Breakpoints set with this command only affect one execution, and then the NB buffer is cleared. Enter <C> to resume the simulation. NB is listed in the TMS320C10 DM menu and in the TMS320C25 EH menu.

The new number is entered as a decimal integer. If NB = 0 (default), this command has no effect. Entering a carriage return for the new value leaves the present value unchanged.

Example 1

```
ENTER COMMAND (HELP=<CR>):
NB <CR>

ENTER NUMBER OF INSTRUCTIONS TILL BREAK
5 <CR>

ENTER COMMAND (HELP=<CR>):
```

OV — Modify/Inspect Overflow Flag Register

Description

OV allows you to inspect and change the overflow flag register value. This command is in the TMS320C10 Simulator RH menu and in the TMS32020 and TMS320C25 Simulator STH menu.

OV may be set to one of the values shown in Table 3-6.

Table 3-6. Overflow Flag Values and Results

OV	Result
0	No overflow or underflow has occurred.
1	An overflow or underflow has occurred. The program caused the accumulator to exceed the positive limit, >7FFFFFFF, causing overflow, or the negative limit, >80000000, causing underflow.

Example

```
ENTER COMMAND (HELP=<CR>):
OV <CR>

PRESENT VALUE OF THE OVERFLOW FLAG REGISTER
1
ENTER NEW VALUE (0,1)
0 <CR>

ENTER COMMAND (HELP=<CR>):
```

The overflow flag is changed to a logic 0 (i.e., no overflow occurred) in this example.

Use the ST command to verify the status of the overflow flag register.

Modify/Inspect Overflow Mode Register OVM

Description

The OVM command allows you to inspect and change the overflow mode register value. OVM is in the TMS320C10 Simulator RH menu and in the TMS32020 and TMS320C25 Simulator STH menu.

OVM may be set to one of the values shown in Table 3-7.

Table 3-7. Overflow Mode Values

OVM	Result
0	Allows an overflow or underflow to occur.
1	Results in saturation on an overflow or underflow.

Example

```
ENTER COMMAND (HELP=<CR>):
OVM <CR>

PRESENT VALUE OF THE OVERFLOW MODE REGISTER
1
ENTER NEW VALUE (0,1)
0 <CR>

ENTER COMMAND (HELP=<CR>):
```

The overflow mode is changed from a logic 1 to a logic 0. This prevents the accumulator from saturating on an overflow.

Use the ST command to check the status of the overflow mode register.

Description The P command allows you to inspect and modify the P register. P is listed in the RH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
P <CR>

PRESENT VALUE OF THE P REGISTER
>        0
ENTER NEW VALUE (IN HEX)
890F <CR>

ENTER COMMAND (HELP=<CR>):
```

Use the ST command to verify the status of the P register.

Modify/Inspect Program Counter PC

Description

The PC command allows you to inspect and change the program counter. PC is listed in the RH menu.

When the program counter is modified, the change is not reflected in the ST command output until the simulation is executed. Since the TMS320C10 program counter is only twelve bits, a maximum value of >FFF can be entered. The TMS32020 program counter and the TMS320C25 program counter, however, are 16 bits, so a maximum value of >FFFF can be entered.

Example

```
ENTER COMMAND (HELP=<CR>):
PC <CR>

PRESENT VALUE FOR PROGRAM COUNTER
> 23
ENTER A NEW VALUE FOR THE PROGRAM COUNTER (IN HEX)
0 <CR>

ENTER COMMAND (HELP=<CR>):
```

In this example, the program counter is changed from >023 to >000; the simulator will begin execution at >000.

PM — Modify/Inspect Product Shift Mode (TMS32020/C25)

Description

PM displays and allows modification of the product shift-mode field of status register one. PM is listed in the TMS32020 and TMS320C25 Simulator STH menu.

The possible values for PM are shown in Table 3-8.

Table 3-8. Product Shift-Mode Values and Results

PM	Result
0	The 32-bit product of the multiplier is loaded into the ALU with no shift.
1	The PR output is left-shifted one bit and loaded into the ALU with the LSB zero-filled.
2	The PR output is left-shifted four bits and loaded into the ALU with the LSB zero-filled.
3	PR output is right-shifted six bits and sign-extended.

Example

```
ENTER COMMAND (HELP=<CR>):
PM <CR>

PRESENT VALUE OF THE PRODUCT SHIFT MODE
0
ENTER NEW VALUE (0 - 3)
2 <CR>

ENTER COMMAND (HELP=<CR>):
```

The PR contents remain unchanged. Shifts take place when the contents of the PR register are transferred to the ALU.

Specify Wait Cycles for External Program Memory (TMS32020/C25) PWAIT

Description PWAIT selects the number of wait cycles for an external program memory response, providing a more flexible and accurate timing analysis. PWAIT is listed in the TMS32020 and TMS320C25 Simulator TICH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
PWAIT <CR>

PRESENT VALUE OF PWAIT
0
ENTER NEW VALUE (0 - 99)
2 <CR>

ENTER COMMAND (HELP=<CR>):
```

Q — Quit Simulation

Description Q terminates the simulation session and returns to the system prompt. Q is in the TMS320C10 Simulator DM menu and in the TMS32020/C25 Simulator EH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
Q <CR>
```

Run Simulation R

Description

R begins the simulation. It is located in the DM menu. After entering R, the simulation may be interrupted by:

- Encountering a breakpoint.
- Branching to self.
- Reaching the limit on the number of instructions that may be executed (NB command).

VAX/VMS

- <CNTRL-C> halts any run mode process and returns the user to the simulator prompt.
- <ESC> halts any command mode process and returns the user to the simulator prompt.

MS/PC-DOS

- <ESC> halts any process (run or command mode) and returns the user to the simulator prompt.

Like the SS command, the R command automatically implements the ST command, displaying the various register values when simulation terminates.

Example

```
ENTER COMMAND (HELP=<CR>):
R <CR>
```

After the run is completed, the simulator displays/updates the current status of the registers (see the ST command for sample displays).

```
ENTER COMMAND (HELP=<CR>):
```

Description RAM displays and allows modification of data RAM memory. RAM is in the MH menu.

Initially, the simulator prompts for a starting address (where the scan begins). Once a beginning address is chosen, the address and contents of that location are displayed.

If no address is chosen (a carriage return is entered after the first prompt),

- The TMS320C10 Simulator starts the scan at address location 0.
- The TMS32020 and TMS320C25 Simulators begin the scan at the address location last accessed. The initial address, though, upon powerup or reset, is 0.

Memory contents and locations can be displayed or modified by entering one of the first four commands listed below. That command can be repeated as many times as necessary to complete scanning of memory.

+	Displays the next memory address and its contents.
-	Displays the previous memory address and its contents.
#	Modifies the contents of the memory address currently displayed so that it contains the hexadecimal value. Memory addresses are limited to a range of >0 to >8F on the TMS320C10 and >0 to >FFFF on the TMS32020 and TMS320C25.
@#	Displays the memory address # (where # is a valid hexadecimal address) and its contents, regardless of what address was previously displayed.
<CR>	Repeats the previous + or - command.
Q	Quits the scan.

Example

```
ENTER COMMAND (HELP=<CR>):
RAM <CR>

ENTER STARTING ADDRESS (IN HEX)
1 <CR>

    1 =   0
+ <CR>
    2 =   0
<CR>                 *Repeat the + option.*
    3 =   0
- <CR>
    2  =  0
<CR>                 *Repeat the - option.*
    1 =   0
10 <CR>
    1 =  10
@55 <CR>
* * * STARTING ADDRESS CHANGED * * *
    55 =   0
Q <CR>

ENTER COMMAND (HELP=<CR>):
```

Description RAMH displays various amounts of RAM memory in block style. The contents of each address are displayed as hexadecimal numbers. RAMH is found in the MH menu.

- The TMS320C10 Simulator, running on a VAX/VMS system, displays all the data RAM memory. The simulator requires no starting or ending address since the entire RAM is displayed.
- The TMS320C10 Simulator, running on an MS/PC-DOS system, displays half of the data RAM memory. The simulator asks if you want to see the first or second half of data RAM memory.
- If the ERAM command has been used, the RAMH command will also display the additional RAM that was created by ERAM.
- The TMS32020 and TMS320C25 Simulators, running on VAX/VMS or MS/PC-DOS operating systems, display selected portions of data RAM. The simulator prompts for the hexadecimal starting address of RAM memory.

Example This TMS320C10 example is for a VAX/VMS system.

```
ENTER COMMAND (HELP=<CR>):
RAMH <CR>

          0    1    2    3    4    5    6    7    8    9

    0    10    0    0    0    0    0    0    0    0    0
   10    20    0    0    0    0    0    0    0    0    0
   20     0    0    0    0    0    0    0    0    0    0
   30     0    0    0    0    0    0    0    0    0    0
   40     0    0    0    0    0    0    0    0    0    0
   50     0    0    0    0    0    0    0    0    0    0
   60     0    0    0    0    0    0    0    0    0    0
   70     0    0    0    0    0    0    0    0    0    0
   80     0    0    0    0    0    0    0    0    0    0
   90     0    0    0    0    0    0    0    0    0    0
  100     0    0    0    0    0    0    0    0    0    0
  110     0    0    0    0    0    0    0    0    0    0
  120     0    0    0    0    0    0    0    0    0    0
  130     0    0    0    0    0    0    0    0    0    0
  140     0    0    0    0    0    0    0    0    0    0

ENTER COMMAND (HELP=<CR>):
```

The >10 in address location 0 is a hexadecimal number, as is the >20 in location 10.

Example This TMS320C10 example is for an MS/PC-DOS system.

```
ENTER COMMAND (HELP=<CR>):
RAMH <CR>

ENTER  0 TO DISPLAY ADDRESSES >0 - >47
       1 TO DISPLAY ADDRESSES >48 - >8F
0 <CR>

>>ADDRESS
   0   10     0     0     0     0     0     0     0
   8   20     0     0     0     0     0     0     0
  10    0     0     0     0     0     0     0     0
  18    0     0     0     0     0     0     0     0
  20    0     0     0     0     0     0     0     0
  28    0     0     0     0     0     0     0     0
  30    0     0     0     0     0     0     0     0
  38    0     0     0     0     0     0     0     0
  40    0     0     0     0     0     0     0     0

ENTER COMMAND (HELP=<CR>):
```

The >10 in address location 0 is a hexadecimal number, as is the >20 in location 8.

Example This example is for the TMS320C25 Simulator (both VAX/VMS and MS/PC-DOS).

```
COMMAND:  RAMH <CR>

Enter start DATA address (in Hex): 0 <CR>

   0   10     0     0     0     0     0     0     0
   8   20     0     0     0     0     0     0     0
  10    0     0     0     0     0     0     0     0
  18    0     0     0     0     0     0     0     0
  20    0     0     0     0     0     0     0     0
  28    0     0     0     0     0     0     0     0
  30    0     0     0     0     0     0     0     0
  38    0     0     0     0     0     0     0     0

COMMAND:
```

In this example, the >10 in address location >0 is a hexadecimal number as is the >20 in location >8. To view data RAM values beyond those displayed, re-enter the RAMH command. Respond to the prompt, "`Enter starting address (in Hex)`," with the new starting address and a carriage return. The new range will then be displayed.

Description RAMI displays various amounts of RAM memory in block style. The contents of each address are displayed as integer numbers. RAMI is found in the MH menu.

- The TMS320C10 Simulator, running on a VAX/VMS system, displays all the data RAM memory. The simulator requires no starting or ending address since the entire RAM is displayed.
- The TMS320C10 Simulator, running on an MS/PC-DOS system, displays half of the data RAM memory. The simulator asks if you want to see the first or second half of data RAM memory.
- If the ERAM command has been used, the RAMI command will also display the additional RAM that was created by ERAM.
- The TMS32020 and TMS320C25 Simulators, running on VAX/VMS or MS/PC-DOS operating systems, display selected portions of data RAM. The simulator prompts for the hexadecimal starting address of RAM memory.

Example 1 This TMS320C10 example is for a VAX/VMS system.

```
ENTER COMMAND (HELP=<CR>):
RAMI <CR>

         0    1    2    3    4    5    6    7    8    9

  0     16    0    0    0    0    0    0    0    0    0
 10     32    0    0    0    0    0    0    0    0    0
 20      0    0    0    0    0    0    0    0    0    0
 30      0    0    0    0    0    0    0    0    0    0
 40      0    0    0    0    0    0    0    0    0    0
 50      0    0    0    0    0    0    0    0    0    0
 60      0    0    0    0    0    0    0    0    0    0
 70      0    0    0    0    0    0    0    0    0    0
 80      0    0    0    0    0    0    0    0    0    0
 90      0    0    0    0    0    0    0    0    0    0
100      0    0    0    0    0    0    0    0    0    0
110      0    0    0    0    0    0    0    0    0    0
120      0    0    0    0    0    0    0    0    0    0
130      0    0    0    0    0    0    0    0    0    0
140      0    0    0    0    0    0    0    0    0    0

ENTER COMMAND (HELP=<CR>):
```

The 16 in address location 0 is an integer number, as is the 32 in location 10.

Example 2 This TMS320C10 example is for an MS/PC-DOS system.

```
ENTER COMMAND (HELP=<CR>):
RAMI <CR>

ENTER  0 TO DISPLAY ADDRESSES >0 - >47
       1 TO DISPLAY ADDRESSES >48 - >8F
0 <CR>

>>ADDRESS
   0    16      0     0     0     0     0     0     0
   8    32      0     0     0     0     0     0     0
  10     0      0     0     0     0     0     0     0
  18     0      0     0     0     0     0     0     0
  20     0      0     0     0     0     0     0     0
  28     0      0     0     0     0     0     0     0
  30     0      0     0     0     0     0     0     0
  38     0      0     0     0     0     0     0     0
  40     0      0     0     0     0     0     0     0

ENTER COMMAND (HELP=<CR>):
```

The 16 in address location >0 is an integer number, as is the 32 in location >8.

Example 3 This example is for the TMS32020/C25 Simulator (both VAX/VMS and MS/PC-DOS).

```
COMMAND:  RAMI <CR>

Enter starting address (in Hex): 0 <CR>

   0    16      0     0     0     0     0     0     0
   8    32      0     0     0     0     0     0     0
  10     0      0     0     0     0     0     0     0
  18     0      0     0     0     0     0     0     0
  20     0      0     0     0     0     0     0     0
  28     0      0     0     0     0     0     0     0
  30     0      0     0     0     0     0     0     0
  38     0      0     0     0     0     0     0     0

COMMAND:
```

The 16 in address location >0 is an integer number as is the 32 in location >8.

Description

RB removes breakpoints. RB prompts for the reference number of the breakpoint to be removed. The DB command displays breakpoints and their reference numbers. RB is listed in the BH menu.

When a breakpoint is removed, all remaining breakpoints are automatically resequenced, unless the last breakpoint is the only one deleted. If more than one RB command is performed in the same session, the reference number for each breakpoint will change as each RB command is completed.

Example 1

```
ENTER COMMAND (HELP=<CR>):
DB <CR>

REF#  SET BY   ADDRESS   VALUE
----  --- --   -------   -----

 1    BDP               >111100001111XXXX
 2    BIAQ    >10
 3    BDR     >5 - >F

ENTER COMMAND (HELP=<CR>):
RB <CR>

ENTER A BREAKPOINT REFERENCE NUMBER
2 <CR>

BREAKPOINT DELETED
BIAQ        >  10

ENTER COMMAND (HELP=<CR>):
DB <CR>

REF#  SET BY   ADDRESS   VALUE
----  --- --   -------   -----

1     BDP                >111100001111XXXX
2     BDR      >5 - >F

ENTER COMMAND (HELP=<CR>):
```

This example deletes breakpoint #2 at address >10, created by the BIAQ command.

Assign RCV Channel to File (TMS320C25) RCV

Description

RCV assigns a serial port (RCV) channel to a specified file. The simulator prompts for the number of clock tics per sample (which is per word or per byte, depending upon the mode in effect), read from the input file. The data is produced in hexadecimal ASCII format. RCV is in the TMS320C25 IOH menu.

If the channel to the ports is not specified, no interrupt will occur in the simulator. When no interrupt occurs, the simulation speed accelerates. If the channel to the ports is specified, the simulator displays the serial port register in greater brightness than before. When the input stream data is exhausted, the file is automatically closed. The constant "FFFF" appears in normal brightness.

Example

```
COMMAND:  RCV <CR>

Enter RCV data file name :  TEXT1.TXT

Min RCV IRT cycle (x100) = 16 when FO=1
Min RCV IRT cycle (x100) = 32 when FO=0
Enter RCV IRT cycle (x100 nsec):  32

   serial port RCV channel assigned.

COMMAND:
```

Description RCVC closes the RCV channel file.

Example

```
COMMAND:  RCVC <CR>

serial port RCV channel closed.

COMMAND:
```

Modify/Inspect Registers/Flags Help Menu RH

Description RH displays the commands that display/modify registers and flags. RH is listed in the DM menu.

Example This display appears for the TMS320C10.

```
ENTER COMMAND (HELP=<CR>):
RH <CR>

MODIFY REGISTERS/FLAGS COMMANDS ARE:
    ACC    = MODIFY/INSPECT ACCUMULATOR
    AR     = MODIFY/INSPECT AUXILIARY REGISTERS
    ARP    = MODIFY/INSPECT AUXILIARY REGISTER POINTER
    BIO    = MODIFY/INSPECT I/O BRANCH CONTROL
    CC     = MODIFY/INSPECT CLOCK COUNTER
    DP     = MODIFY/INSPECT DATA MEMORY PAGE POINTER
    INTF   = MODIFY/INSPECT INTERRUPT FLAG REGISTER
    INTM   = MODIFY/INSPECT INTERRUPT FLAG MODE REGISTER
    OV     = MODIFY/INSPECT OVERFLOW FLAG REGISTER
    OVM    = MODIFY/INSPECT OVERFLOW MODE REGISTER
    P      = MODIFY/INSPECT P REGISTER
    PC     = MODIFY/INSPECT PROGRAM COUNTER
    SK     = MODIFY/INSPECT STACK
    T      = MODIFY/INSPECT T REGISTER

ENTER COMMAND (HELP=<CR>):
```

Example This display appears for the TMS32020/C25.

```
COMMAND:  RH <CR>

Modify/Inspect
   ACC     : Accumulator       AR     : AUX Registers
   P       : P register        T      : T register
   PC      : Program Counter   RPTC   : RePT Counter
   SK      : Stack             CC     : Clock Counter
   TINT    : Timer IRT flg     TINTM  : Timer IRT Mask
   INTFS   : Irt flags(1-2)    INTMS  : IRT masks(0-2)
   BIO     : I/O branch CTL

COMMAND:
```

Any command may be entered after the COMMAND: prompt.

For all commands in the RH menu, entering a carriage return after the ENTER NEW VALUE prompt leaves the present register or flag value unchanged.

Description

ROM allows you to inspect and change program ROM memory. ROM is listed in the MH menu.

Initially, the simulator prompts for a starting address (where the scan begins). Once a beginning address is chosen, the address and contents of that location are displayed. If no address is chosen (a carriage return is entered after the first prompt),

- The TMS320C10 Simulator starts the scan at address location 0.
- The TMS32020 and TMS320C25 Simulators begin the scan at the address location last accessed. The initial address, though, upon powerup or reset, is 0.

Memory contents and locations can be displayed or modified by entering one of the first four commands listed below. That command can be repeated as many times as necessary to complete scanning of memory.

+ Displays the next memory address and its contents.

- Displays the previous memory address and its contents.

# Modifies the contents of the memory address currently displayed so that it contains the hexadecimal value. Memory addresses are limited to range of >0 to >8F on the TMS320C10 and >0 to >FFFF on the TMS32020 and TMS320C25.

@# Displays the memory address # (where # is a valid hexadecimal address) and its contents, regardless of what address was previously displayed.

<**CR**> Repeats the previous + or - command.

Q Quits the scan.

Note:

The ROM command does **not** inspect or modify internal Program RAM or external Program memory. After a CNFP instruction (or a CNF=1 command), the ROM command continues to display external ROM at a block b0 location. To inspect or modify block b0 under these conditions, the RAM command must be used. However, the display commands (ROMH and ROMI) will display Program memory locations and/or RAM locations in block b0 when the CNF bit is set to 1 and the address is in the range >FF00 to >FFFF.

Example 1

```
ENTER COMMAND (HELP=<CR>):
ROM <CR>

ENTER STARTING ADDRESS (IN HEX)
44 <CR>
    44 =   0
+ <CR>
    45 =   0
<CR>             *Repeat the + option.*
    46 =   0
-  <CR>
    45 =   0
<CR>             *Repeat the - option.*
    44 =   0
10 <CR>
    44 =  10
@111 <CR>

* * * STARTING ADDRESS CHANGED * * *
    111 = 0
Q <CR>

ENTER COMMAND (HELP=<CR>):
```

Description ROMH displays a selected portion of program ROM memory in block style. The contents of each address are displayed as hexadecimal numbers. VAX/VMS systems display 152 consecutive words of program ROM. MS/PC-DOS systems display 72 consecutive words of program ROM. ROMH is found in the MH menu.

Example This example is for an MS/PC-DOS system running the TMS320C10 Simulator.

```
ENTER COMMAND (HELP=<CR>):
ROMH <CR>

ENTER STARTING ADDRESS (IN HEX)
C0 <CR>

>>PC
  C0    10     0     0     0     0     0     0     0
  C8    20     0     0     0     0     0     0     0
  D0     0     0     0     0     0     0     0     0
  D8     0     0     0     0     0     0     0     0
  E0     0     0     0     0     0     0     0     0
  E8     0     0     0     0     0     0     0     0
  F0     0     0     0     0     0     0     0     0
  F8     0     0     0     0     0     0     0     0
 100     0     0     0     0     0     0     0     0

ENTER COMMAND (HELP=<CR>):
```

The >10 in location >CO is a hexadecimal number as is the >20 in location >C8. Note that the addresses are displayed in hexadecimal.

Example

This example is for a VAX/VMS system or an MS/PC-DOS system running the TMS32020/C25 Simulator.

```
COMMAND:   ROMH <CR>

Enter start PROGRAM address (in Hex): 0 <CR>

   0   10     0     0     0     0     0     0     0
   8   20     0     0     0     0     0     0     0
  10    0     0     0     0     0     0     0     0
  18    0     0     0     0     0     0     0     0
  20    0     0     0     0     0     0     0     0
  28    0     0     0     0     0     0     0     0
  30    0     0     0     0     0     0     0     0
  38    0     0     0     0     0     0     0     0

COMMAND:
```

The >10 in location >0 is a hexadecimal number as is the >20 in location>8. Note that the addresses are displayed in hexadecimal. To view values beyond those displayed, re-enter the ROMH command. Respond to the prompt, "`Enter starting address (in Hex),`" with the new starting address and a carriage return. The new range will then be displayed.

Description ROMI displays a selected portion of program ROM memory in block style. The contents of each address are displayed as integer numbers. VAX/VMS and MS/PC-DOS systems display 152 consecutive words of program ROM. ROMI is found in the MH menu.

Example 1 This example is for an MS/PC-DOS system running the TMS320C10 Simulator.

```
ENTER COMMAND (HELP=<CR>):
ROMI <CR>

ENTER STARTING ADDRESS (IN HEX)
C0 <CR>

>>PC
  C0    16     0     0     0     0     0     0     0
  C8    32     0     0     0     0     0     0     0
  D0     0     0     0     0     0     0     0     0
  D8     0     0     0     0     0     0     0     0
  E0     0     0     0     0     0     0     0     0
  E8     0     0     0     0     0     0     0     0
  F0     0     0     0     0     0     0     0     0
  F8     0     0     0     0     0     0     0     0
 100     0     0     0     0     0     0     0     0

ENTER COMMAND (HELP=<CR>):
```

The 16 in location >C0 is an integer, as is the 32 in location >C8.

Example 2 This example is for a VAX/VMS system or an MS/PC-DOS system running the TMS32020/C25 Simulator.

```
COMMAND:  ROMI <CR>

Enter start PROGRAM address (in Hex): 0 <CR>

    0    16     0     0     0     0     0     0     0
    8    32     0     0     0     0     0     0     0
   10     0     0     0     0     0     0     0     0
   18     0     0     0     0     0     0     0     0
   20     0     0     0     0     0     0     0     0
   28     0     0     0     0     0     0     0     0
   30     0     0     0     0     0     0     0     0
   38     0     0     0     0     0     0     0     0

COMMAND:
```

The 16 in location >0 is an integer number as is the 32 in location >8. Note that the addresses are displayed in hexadecimal. To view values beyond those displayed, re-enter the ROMI command. Respond to the prompt, "`Enter starting address (in Hex)`," with the new starting address and a carriage return. The new range will then be displayed.

RPTC Modify/Inspect Repeat Counter (TMS32020/C25)

Description RPTC displays and allows modification of the repeat instruction counter, which tabulates the number of times an instruction has been repeated. RPTC is helpful in streamlining instructions. RPTC is listed in the TMS32020 and TMS320C25 Simulator RH menu.

Example 1

```
ENTER COMMAND (HELP=<CR>):
RPTC <CR>

PRESENT VALUE OF REPEAT INSTRUCTION COUNTER
0
ENTER NEW VALUE (0 - >FF, IN HEX)
C <CR>

ENTER COMMAND (HELP=<CR>):
```

Reset Simulator RS

Description RS causes a reset. RS is located in the TMS320C10 Simulator DM menu. In the TMS32020/C25 Simulator, RS is in the EH menu. For the TMS320C10, TMS32020, and TMS320C25 Simulators, the reset:

- Loads the program counter with zero (PC = 0).
- Sets the interrupt mode register to 1 to disable interrupts (INTM = 1).
- Clears any pending interrupts (INTF = 0).

Also, the reset for the TMS32020 and TMS320C25 Simulators:

- Loads the CNF bit with 0, configuring all RAM as data memory.
- Clears RPTC.
- Sets status bit OV to zero and status bit XF to 1.
- Sets timer register TIM to >FFFF.

RS does not start the simulation. Use the R or the C command to continue simulation after a reset.

Example

```
ENTER COMMAND (HELP=<CR>):
RS <CR>


ENTER COMMAND (HELP=<CR>):
```

RSI — Reset Selected Input Port File

Description RSI resets an input-port file so that data is taken from the top of the file (i.e., the pointer is repositioned to the top of the file). RSI is located in the IOH menu.

Example This display appears for the TMS320C10.

```
ENTER COMMAND (HELP=<CR>):
RSI <CR>

ENTER THE INPUT PORT (0,...,7)
2 <CR>

INPUT PORT FILE  # 2 HAS BEEN RESET

ENTER COMMAND (HELP=<CR>):
```

Example This display appears for the TMS32020 and TMS320C25.

```
ENTER COMMAND (HELP=<CR>):
RSI <CR>

ENTER THE INPUT PORT (0,...,15)
2 <CR>

INPUT PORT FILE  # 2 HAS BEEN RESET

ENTER COMMAND (HELP=<CR>):
```

The file associated with port 2 has been reset. It is not necessary to use this command when the end of a file is reached as the simulator provides an autowrap feature. Once the end of a file is reached, any further attempt to read from the file automatically resets to the top of that file.

Description SGN uses the sine function to generate test data in the RAM on VAX/VMS systems. The simulator prompts for information to determine where the data table will begin in the RAM and how the data pattern will appear. The default values for SGN are shown in Table 3-9. SGN is in the TMS320C25 UTLH menu.

Table 3-9. Default Values for Generated Test Data in RAM

Data Point	Notation	Default
starting address	hexadecimal	0
period	decimal	128
cycles	decimal	1
stepsize	decimal	1
max—amplitude	decimal	8192

Example 1

```
COMMAND:  SGN <CR>

SIN(x) function generator
start address in hex (default = 0) :
period (default = 128) :
cycles (default = 1) :
step size (default = 1) :
max amplitude (default = 8192) :

COMMAND:
```

In this example, default values were accepted by entering carriage returns.

Note that:

1) The number of words generated is found by this formula:

   n = (period × cycles) - 1

2) For i = 1 to period:

   RAM(start—addr + i-1) = max—ampl[SIN (2π × (i-1)/period)]

Description

SI associates an existing file with an input port. If a file is already associated with the specified port, the new file overrides the old file. The associated input files can be displayed using the LF command on the TMS320C10 and the LI command on the TMS32020 and TMS320C25. The SI command can be found in the IOH menu.

Whenever an IN instruction is executed, data is read from the file associated with the port. If no file has been assigned, the simulation is suspended, and the simulator prompts for input. Once the input is supplied, simulation resumes.

Example

This display appears for the TMS320C10.

```
ENTER COMMAND (HELP=<CR>):
SI <CR>

ENTER THE INPUT PORT (0,...,7)
3 <CR>
ENTER FILE NAME FOR INPUT
NAME.INP <CR>

ENTER COMMAND (HELP=<CR>):
```

Example

This display appears for the TMS32020 and TMS320C25.

```
ENTER COMMAND (HELP=<CR>):
SI <CR>

ENTER THE INPUT PORT (0,...,15)
3 <CR>
ENTER FILE NAME FOR INPUT
NAME.INP <CR>

ENTER COMMAND (HELP=<CR>):
```

The file NAME.INP is now associated with input port 3. Any data input through port 3 is read from the file NAME.INP.

Simulator Mode Command (TMS32020/C25) SIM

Description SIM allows the simulator to be changed from the TMS320C25 simulator to the TMS32020 simulator and back again. SIM also allows the user to select either the microProcesser mode or the microComputer mode.

Example 1 The following prompt appears for the SIM command.

```
Command: SIM

Present simulation mode: 3 = TMS320C25 (MP/MC pin = mi-
croprocessor mode)

1)   TMS32020
2)   TMS320C25 (MP/MC = microComputer  = 0)
3)   TMS320C25 (MP/MC = microProcessor = 0)

Enter new Simulation Mode (1, 2, or 3) 1 <CR>
New Mode = TMS32020

Command:
```

Description

SK provides the means to modify and/or inspect all four levels of the stack. The stack is always displayed in sequential, top-to-bottom form.

As each level is shown, changes can be made at that level. If a change is desired, a hexadecimal value of up to three (TMS320C10) or four (TMS32020, TMS320C25) characters may be entered. If no change is desired, enter a carriage return to proceed to the next level.

To terminate the process before reaching the bottom of the stack, enter a hyphen (-). The simulator then returns to the MH menu. SK is listed in the RH menu.

Note that SK changes the stack contents but does not alter the stack pointer.

Example

This display appears for the TMS320C10.

```
ENTER COMMAND (HELP=<CR>):
SK <CR>

TOP OF STACK - 0
>548
ENTER NEW VALUE (IN HEX) OR "-" TO TERMINATE
<CR>

TOP OF STACK - 1
>52F
ENTER NEW VALUE (IN HEX) OR "-" TO TERMINATE
0 <CR>

TOP OF STACK - 2
>548
ENTER NEW VALUE (IN HEX) OR "-" TO TERMINATE

- <CR>

ENTER COMMAND (HELP=<CR>):
```

This example does not change the top of the stack but zeros the second level (location one) of the stack. At the third level, a hyphen (-) terminates the command.

Example This display appears for the TMS32020/C25.

```
COMMAND:  SK <CR>

          STACK     0      0
          STACK     1      0
          STACK     2      0
          STACK     3      0
          STACK     4      0
          STACK     5      0
          STACK     6      0
          STACK     7      0

Enter Stack level or "-" to terminate :  3 <CR>

Stack value (in Hex):  0 <CR>

Enter Stack level or "-" to terminate :  - <CR>

COMMAND:
```

SO — Select Output Port File

Description

SO associates an existing file with an output port. The associated output files can be displayed using the LF command on the TMS320C10 and the LO command on the TMS32020 and TMS320C25. The SO command is listed in the IOH menu.

If a file is already associated with the specified port, the new file overrides the old file. Therefore, execution of an OUT instruction produces a write to the file associated with the appropriate port. The output defaults to the screen if no file is associated with the appropriate port.

Example 1

This display appears for the TMS320C10.

```
ENTER COMMAND (HELP=<CR>):
SO <CR>

ENTER THE OUTPUT PORT (0,...,7)
5 <CR>
ENTER FILE NAME FOR OUTPUT
NAME.OUT <CR>

ENTER COMMAND (HELP=<CR>):
```

Example 2

This display appears for the TMS32020 and TMS320C25.

```
ENTER COMMAND (HELP=<CR>):
SO <CR>

ENTER THE OUTPUT PORT (0,...,15)
5 <CR>
ENTER FILE NAME FOR OUTPUT
NAME.OUT <CR>

ENTER COMMAND (HELP=<CR>):
```

These examples associate the file NAME.OUT with output port 5. Now when an OUT instruction is directed to port 5, the data will be written to the file NAME.OUT.

Store RAM Data to External File (TMS320C25) SRAM

Description

SRAM stores RAM data to an external file, making RAM initialization easier. The SRAM command prompts for a filename, the starting address, and the ending address of the data memory to be stored in a file. The default address address range is 0-65535.

You may create a file using the SRAM command, which stores the RAM data to an external file. The LRAM command loads the stored file into simulator memory. Note that the file you load using the LRAM command must have the same format as the file created using the SRAM command. SRAM is in the UTLH menu.

Example 1

```
COMMAND:  SRAM <CR>

ENTER FILE NAME: TEXT1.TXT <CR>
Enter START address (in Hex): 0 <CR>
Enter END address (in Hex)  : 7 <CR>

COMMAND:
```

Below is the format of the file created by SRAM.

```
         PSEG DATA
         DEF  XXXX
XXXX     EQU  $
  DATA 19728
  DATA 19731
  DATA 19733
  DATA 19747
  DATA 19781
  DATA 19781
  DATA 19796
     PEND
[EOB]
```

Description

SS executes a program one instruction at a time, suspending execution after each instruction. SS is located in the TMS320C10 Simulator DM menu. For the TMS32020/C25 Simulator, SS is in the EH menu.

The ST command is automatically implemented after each program instruction is executed, displaying the new status of the registers. To continue the single-step execution, enter a carriage return. To halt the execution, enter a hyphen (-).

Example

```
ENTER COMMAND (HELP=<CR>):
SS <CR>
```

After each step is completed, the simulator displays the current status of the registers (see the ST command for sample displays).

```
ENTER <CR> TO CONTINUE
      "-" TO TERMINATE
- <CR>

ENTER COMMAND (HELP=<CR>):
```

This example returns to the main simulator prompt after it executes one instruction. The hyphen (-) terminates the single-step session.

Description

ST simultaneously displays (in hexadecimal) the contents of the program counter, the opcode and mnemonic name of the current instruction, the previous value of the program counter, and all four stack location contents. On the TMS32020/C25, ST updates the already displayed fields.

ST displays the following registers and their values on the TMS320C10:

Auxiliary Register Pointer (ARP)
Auxiliary Registers (AR0, AR1)
T Register (TREG)
P Register (PREG)
Accumulator (ACC)
Clock Counter (CLK)
Data Memory Page Pointer (DP)
I/O Branch Control Pin (BIO)
Interrupt Flag Register (INTF)
Interrupt Mode Register (INTM)
Overflow Flag Register (OV)
Overflow Mode Register (OVM)

On the TMS32020/C25, the ST command updates the following registers and their values:

Program counter (PC)
Instruction executed in previous cycle (-1)
Instruction executed 2 cycles past (-2)
Instruction executed 3 cycles past (-3)
Auxiliary registers (AR0-AR7)
Stack Values (SK0-SK7)
Interrupt flag register (IFR)
Interrupt mask register (IMR)
Ram(0) serial port data receive register (DRR)
Ram(1) serial port data transmit register
RAM(2) timer register (TIM)
Period register value (PDD
Global memory allocation register value (GREG)
The T-register value (TREG)
The P-register value (PREG)
Accumulator (ACC)
Chip operation in Microprocessor/Microcomputer mode (MP/MC)
Status Register 0 value (ST0)
Auxiliary register buffer value (ARB)
Carry Flag (CRY)
Frame sychronized mode bit (FSM)
Overflow mode flag (OVM)
Test/Control flag (TC)
Output port (OUTP)
Repeat instruction counter (RPTC)
Status register 1 value (ST1)
Auxiliary register buffer pointer value (ARP)
Data memory page pointer (DP)
Interrupt mode (INTM)
Product register shift mode (PM)
Transmit mode bit (TXM)
Clock counter (CLK)
I/O branch control value (BIO)
On chip Ram B0 configuration (CNFD/CNFP)
Format mode bit (FO)
Overflow mode flag (OV)
Sign extension mode flag (SXM)
Status of external flag output pin (XF)
Prompt for command input (COMMAND)

Example This example shows a TMS320C10 display.

```
ENTER COMMAND (HELP=<CR>):

ST <CR>

>>PC=   7       OPCODE=    0     ADD     PREVIOUS PC=   6

               ARP    AR0    AR1    TREG    PREG    ACC    CLK
INTEGER          1     16      0     256       5     21      8
HEX              1     10      0     100       5     15      8

>>STK=    2    1     0     0     DP = 0  INTF= 0  OV = 0
                                 BIO= 1  INTM= 1  OVM= 0

ENTER COMMAND (HELP=<CR>):
```

Example This example shows a TMS32020/C25 display.

```
COMMAND: ST <CR>
```

```
PC :0000  ADD >0200        ┌─────────────C25 MP──────────────────────────────
-1 :0000  ADD >0200        │ IFR :000000   │ ST0:0604  ST1 :07F0  BIO: 1
-2 :0000  ADD >0200        │ IMR :000000   │ ARB:0     ARP :0     CNFD
-3 :0000  ADD >0200        ├───MMRS────────┤ CRY:0     DP  :04    FO :0
─────────────────┬───STACK─┤ DRR :0000     │ FSM:1     INTM:1     OV :0
  AR0:   0000    │ SK0: 0000│ DXR :0000     │ OVM:1     PM  :1     SXM:1
  AR1:   0000    │ SK1: 0000│ TIM :FFFF     │ TC  :0    TXM :0     XF :1
  AR2:   0000    │ SK2: 0000│ PRD :FFFF     │ OUTP:0000
  AR3:   0000    │ SK3: 0000│ GREG:0000     │ RPTC:    0   CLK:      0
  AR4:   0000    │ SK4: 0000├───────────────┴──────────────────────────────
  AR5:   0000    │ SK5: 0000│    TREG:       0000
  AR6:   0000    │ SK6: 0000│    PREG:   00000000
  AR7:   0000    │ SK7: 0000│    ACC :   00000000
─────────────────┴──────────┴──────────────────────────────────────────────
COMMAND:
```

The ST command has updated the simulator's machine-state display fields.

STH — Modify/Inspect Status Registers/Pins Help Menu (TMS32020/C25)

Description STH displays the commands used to display/modify parameters that appear in the status register. STH is located in the TMS32020 and TMS320C25 DM menu.

Example 1 This display appears for the TMS32020/C25.

```
COMMAND:  STH <CR>

MODIFY/INSPECT STATUS REGISTER COMMANDS ARE:

 ARB : AUX reg pointer buff ARP  : AUX Reg Pointer
 CNF : RAM CNF bit          DP   : Data memory page Pointer
 FO  : Format bit           INTM : INTerrupt flag Mask reg
 OV  : Ovflw flg reg        OVM  : Ovflw Mode reg
 PM  : Product shift mode   SXM  : Sign extension mode bit
 TC  : Test/Control flg bit TXM  : Transmit mode bit
 XF  : XF pin               CY   : Carry bit
 HM  : Hold mode bit        FSM  : Frame sync mode bit

COMMAND:
```

Any command may be entered after the COMMAND: prompt.

Save Trace Buffer STR

Description STR saves the contents of the trace buffer in a file for retrieval after the simulator session is ended. STR is located in the TMS320C10 Simulator DM menu and in the TMS32020/C25 Simulator TH menu.

Entering the STR command produces one of the two following series of prompts, as shown in the examples.

Example The display in the trace buffer file is nonexistent.

```
ENTER COMMAND (HELP=<CR>):
STR <CR>

ENTER TRACE BUFFER FILE NAME - NONE EXISTS
TRACE.DAT <CR>

ENTER COMMAND ((HELP=<CR>):
```

On a VAX/VMS system, if no file name is entered, the TMS320C10 Simulator uses the default file FOR088.DAT. All other simulators will not create a file at all.

Example This display is shown if STR has already been used in the present session.

```
ENTER COMMAND ((HELP=<CR>):
STR <CR>

TRACE BUFFER FILE ALREADY EXISTS
ENTER NEW NAME
TRACE1.DAT <CR>

ENTER COMMAND (HELP=<CR>):
```

If a new file name is not entered, the current trace buffer will be appended to the contents of the old file (the old contents will not be overwritten).

SW — Switch from Screen to File (TMS32020/C25)

Description

The SW command saves the simulation state in a disk file by switching the screen output device from the terminal to the specified file. Once the output is switched to the file, the results of the following commands are saved in a file:

RAM
RAMH
RAMI
ROM
ROMH
ROMI
PUSH
POP
VIEW

The SW command prompts for a log filename if one has not yet been set. Entering the SW command a second time causes future displays to be output on a standard output device. Entering the SW command a third time toggles the mode so that the output displays from the above commands are appended to the log file.

Example 1

```
COMMAND:  SW <CR>

ENTER LOG FILE NAME :  TEXT.TXT
switched to the log file

COMMAND:
```

Modify/Inspect Sign-Extension Mode Bit (TMS32020/C25) SXM

Description SXM allows you to inspect and change the sign-extension mode bit value. SXM is listed in the TMS32020 and TMS320C25 Simulator STH menu.

The SXM bit values are listed in Table 3-10.

Table 3-10. Sign-Extension Mode Bit Values

SXM	Result
0	Suppresses sign extension.
1	Sign extension as data is passed into the accumulator.

Example

```
ENTER COMMAND (HELP=<CR>):
SXM <CR>

PRESENT VALUE OF SIGN EXTENSION MODE BIT
0
ENTER NEW VALUE (0 OR 1)
1 <CR>

ENTER COMMAND (HELP=<CR>):
```

T — Modify/Inspect T Register

Description The T command displays and allows modification of the T register. T is located in the RH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
T <CR>

PRESENT VALUE OF THE T REGISTER
>     F89
ENTER NEW VALUE (IN HEX)
0 <CR>

ENTER COMMAND (HELP=<CR>):
```

Check the status of the T register with the ST command.

Modify/Inspect Test/Control Flag Bit (TMS32020/C25) TC

Description TC displays and allows modification of the test/control flag bit of the status register. TC is listed in the TMS32020 and TMS320C25 Simulator STH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
TC <CR>

PRESENT VALUE OF TEST/CONTROL FLAG BIT
0
ENTER NEW VALUE (0 OR 1)
1 <CR>

ENTER COMMAND (HELP=<CR>):
```

TH — Trace Help Menu (TMS320C25)

Description TH displays commands that control notation, and display and update the status registers. TH is located in the TMS320C25 DM menu.

Example This display appears for the TMS320C25.

```
COMMAND:  TH <CR>

   STR  : save the trace buffer
   TR   : toggle trace mode (on or off)
   JF   : select journal file
   DT   : display the trace buffer
   Z    : zero clock counter

COMMAND:
```

Specify Clock Tics until Next Interrupt (TMS320C10) TIC

Description

TIC generates interrupts every x number of clock tics.

After the simulator prompts for the interrupt interval, the simulator prompts for the interrupt counter. Both values are entered as hexadecimal numbers. In the following example, an interrupt is generated every >1000 clock tics until five interrupts have been generated. The simulator then proceeds without interrupts. The TIC command is listed in the TMS320C10 DM menu.

Example

```
ENTER COMMAND (HELP=<CR>):
TIC <CR>

ENTER THE NUMBER OF CLOCK TICS TILL INTERRUPT
1000 <CR>
ENTER THE NUMBER OF TIMES TO REPEAT THE INTERRUPT
CYCLE
5 <CR>

ENTER COMMAND (HELP=<CR>):
```

Note: TIC0-TIC2 commands are used for the TMS32020/C25 simulator.

TICO-TIC2 — Specify Clock Tics until Interrupt (TMS32020/C25)

Description The TICO, TIC1, and TIC2 commands allow interrupts to be generated every x number of clock tics. The TICO–TIC2 command is listed in the TMS32020 and TMS320C25 Simulator TICH menu.

After the simulator prompts for the interrupt interval, the simulator prompts for the interrupt counter. Both values are entered as decimal numbers. In the following example, interrupt #0 is generated every 1000 clock tics until five interrupts have been generated. The simulator then proceeds without interrupts.

Example

```
ENTER COMMAND (HELP=<CR>):
TICO <CR>

ENTER THE NUMBER OF CLOCK TICS TILL INTERRUPT #0
1000 <CR>
ENTER THE NUMBER OF TIMES TO REPEAT THE INTERRUPT
CYCLE
5 <CR>

ENTER COMMAND (HELP=<CR>):
```

Each TIC command (TICO, TIC1, and TIC2) produces the same series of prompts.

Description TICH lists commands that are available to display/modify timing and interrupt-related items. TICH is located in the TMS32020 and TMS320C25 Simulator DM menu.

Example This display appears for the TMS32020/C25.

```
COMMAND:  TICH <CR>

INTERRUPT/TIMING COMMANDS ARE:
   DWAIT     : # of wait cycles for Ext-Data memory response
   IOWAIT    : # of wait cycles for Ext-I/O memory response
   PWAIT     : # of wait cycles for Ext-program memory
             : response
   TIC0      : number of clock tics till interrupt #0
   TIC1      : number of clock tics till interrupt #1
   TIC2      : number of clock tics till interrupt #2
   ZTIC      : disables the tic commands
   XINTM     : modify XINTM bit
   RINTM     : modify RINTM bit

COMMAND:
```

Any command may be submitted after the COMMAND prompt.

TINT — Modify/Inspect Timer Interrupt Flag (TMS32020/C25)

Description TINT displays and allows modification of the value of the timer interrupt flag in the interrupt flag register. TINT is listed in the TMS32020 and TMS320C25 Simulator RH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
TINT <CR>

PRESENT VALUE OF TINT
0
ENTER NEW VALUE (0,1)
1 <CR>

ENTER COMMAND (HELP=<CR>):
```

Modify/Inspect Timer Interrupt Mask (TMS32020/C25) TINTM

Description TINTM displays and allows modification of the value of the timer-interrupt mask in the interrupt mask register. TINTM is listed in the TMS32020 and the TMS320C25 Simulator RH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
TINTM <CR>

PRESENT VALUE OF TINTM
0
ENTER NEW VALUE (0,1)
1 <CR>

ENTER COMMAND (HELP=<CR>):
```

TR — Toggle Trace Mode

Description

TR toggles the trace mode on or off. The trace is a circular buffer, 256 samples long, that traces the auxiliary registers, accumulator, and the program counter. With the trace on, the last 256 states of the simulation can be displayed. The default for the trace is off. TR is located in the TMS320C10 Simulator DM menu. In the TMS32020/C25 Simulator, TR is in the TH menu.

The DT command displays the trace buffer contents; the STR command saves the trace buffer.

Note that when the trace wraps, the previous entries are overwritten; the trace buffer always contains the *last* 256 entries. If execution is stopped and restarted (when the next R, C or SS command is executed), the trace buffer is *reinitialized*; the new entries are not appended to the previous entries.

Example

```
ENTER COMMAND (HELP=<CR>):
TR <CR>

TRACE MODE IS ON

ENTER COMMAND (HELP=<CR>):
```

The trace mode is toggled from off to on in this example.

Modify/Inspect Transmit Mode Bit (TMS32020/C25) TXM

Description TXM displays and allows modification of the transmit mode bit. TXM is listed in the TMS32020 and TMS320C25 Simulator STH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
TXM <CR>

PRESENT VALUE OF TRANSMIT MODE BIT
0
ENTER NEW VALUE (0 OR 1)
1 <CR>

ENTER COMMAND (HELP=<CR>):
```

UTLH — Utilities Help Menu (TMS32020/C25)

Description UTLH displays commands that control notation, and display and update the status registers. UTLH is located in the TMS320C25 DM menu.

Example This display appears for the TMS320C25.

```
COMMAND:  UTLH <CR>

   SW   :  SWitch output toggle [SCREEN/FILE]
   SRAM :  Store Ram data to  text file
   LRAM :  Load Ram data from text file
   DC   :  Display Control
   PUSH :  Save registers on save stack
   POP  :  Restore registers from save stack
   VIEW :  Display the save stack
   SGN  :  Save wave generator  on RAM

COMMAND:
```

Modify/Inspect XF Pin (TMS32020/C25) XF

Description XF allows you to modify and inspect the XF pin. XF is listed in the TMS32020 and TMS320C25 Simulator STH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
XF <CR>

PRESENT VALUE OF XF PIN
1
ENTER NEW VALUE (0 OR 1)
0 <CR>

ENTER COMMAND (HELP=<CR>):
```

Description The XINTM command allows you to inspect and change the XINTM bit in the interrupt mask register. XINTM is in the TICH menu.

Example

```
COMMAND:  XINTM <CR>

Present value of XINTM :0
Enter new value (0,1) :  1 <CR>

COMMAND:
```

Description

The XMT command assigns a serial port (XMT) channel to a specified file. The simulator prompts for the number of clock tics per sample (which is per word or per byte, depending on the mode in effect) read from the input file. The data is produced in hexadecimal ASCII format. XMT is in the IOH menu.

If the channel to the ports is not specified, no interrupt will occur in the simulator When no interrupt occurs, the simulation speed accelerates. If the channel to the ports is specified, the simulator displays the serial port register value in greater brightness than before. When the input stream data is exhausted, the file is automatically closed. The constant "FFFF" appears in normal brightness.

Example

```
COMMAND:  XMT <CR>

Enter XMT data file name:  TEXT2.TXT
Min XMT IRT cycle (x100) = 16 when FO=1
Min XMT IRT cycle (x100) = 32 when FO=0
Enter XMT IRT cycle (x100 nsec): 32


   serial port XMT channel assigned.

COMMAND:
```

XMTC Close the (XMT) Channel File (TMS320C25)

Description The XMTC command closes the XMT channel file. XMTC is in the IOH menu.

Example

```
COMMAND:  XMTC <CR>

   serial port XMT channel closed.

COMMAND:
```

Zero Clock Counter Z

Description Z clears the clock counter. The CC command displays the status of the clock counter (the number of clock cycles that have occurred since the simulator started). Z is listed in the TMS320C10 Simulator DM menu and in the TMS32020/C25 Simulator TH menu.

Example

```
ENTER COMMAND (HELP=<CR>):
Z <CR>

CLOCK COUNTER HAS BEEN ZEROED

ENTER COMMAND (HELP=<CR>):
```

Use the ST command to verify the clock counter status.

ZRAM Set RAM Contents to Zero (TMS32020/C25)

Description ZRAM sets RAM contents (addresses 6 through 65535) to zero. ZRAM is in the MH menu.

Example

```
COMMAND:  ZRAM <CR>

   RAM DATA CLEARED

COMMAND:
```

Disable TIC Commands ZTIC

Description ZTIC disables interrupts set by the TIC command. The ZTIC command is in the TMS320C10 Simulator DM menu and in the TMS32020 and TMS320C25 Simulator TICH menu.

Example This example is for the TMS320C10.

```
ENTER COMMAND (HELP=<CR>):
ZTIC <CR>

THE TIC COMMAND HAS BEEN DISABLED

ENTER COMMAND (HELP=<CR>):
```

Example This example is for the TMS32020 and TMS320C25.

```
ENTER COMMAND (HELP=<CR>):
ZTIC <CR>

TIC0, TIC1, AND TIC2 COMMANDS HAVE BEEN DISABLED

ENTER COMMAND (HELP=<CR>):
```

3.4 I/O Simulation

All first generation TMS320C10 digital signal processors contain the same internal CPU and can benefit from using a common software simulator. In this environment, data files can be used to simulate the I/O interface in a target system. The file should list one data value per line, written at the far left margin.

The TMS320C17 and TMS320E17 each have two additional I/O peripheral circuits: a dual-channel serial port and a coprocessor mode. These ports allow the device to interface directly to two combo codecs and a microcomputer. Although the software simulator does not directly implement these hardware ports, they can be simulated as parallel I/O. Here data is accessed using the parallel I/O port instructions IN and OUT. The sections below discuss simulation techniques for these peripheral interfaces.

3.4.1 Serial Port

The dual-channel serial port on the TMS320C17 device provides all of the necessary clocking and framing signals to directly interface to two combo codecs. The processor interprets all data as parallel 13-bit two's complement numbers. Therefore, the user must set up his input file with the serial port data represented in two's complement format rather than in its 8-bit PCM form.

If the TMS320C17 serial ports are configured to be interrupt driven, then the interrupt traps can be initiated via the simulator's interrupt counter. The count value loaded into the simulator is a function of the sample rate of the processor. The relationship is as follows:

COUNT = (CLKIN/4) / SAMPLE

where COUNT = count value used by the simulator
CLKIN = processor input frequency
SAMPLE = codec's sample rate

The following example illustrates how the number of instruction cycles is calculated before an interface interrupt occurs:

Assume CLKIN = 18.432 MHz
SAMPLE = 9.6 kHz
then COUNT = (18432/4) / 9.6 = 480 instruction cycles.

No synchronization is required if the serial port is polled rather than interrupt driven.

3.4.2 Coprocessor Port

The coprocessor port on the TMS320C17 device facilitates the interface to other microcomputers or microprocessors in the system. In a multiprocessing environment, the TMS320C17 can act as either master or slave. No synchronization is required when simulating the TMS320C17 as master. In a slave configuration, the synchronization can be handled using the cycle count method described above, or the user can "break" the executing code by the following keystrokes:

- Striking <CNTRL-C> on VAX/VMS systems.
- Striking any key on MS/PC-DOS systems running the TMS320C10 simulator.

When interfacing to 8-bit microcomputers in hardware, the TMS320C17 reads the eight LSBs on the data lines. In software, the simulator reads all data as 16-bit values. The data file can contain 4, 8 or 16-bit values.

Section 4

Sample Debugging Sessions

The following debugging sessions are presented to give the reader an understanding of how to find and correct errors in programs. There are two debugging sessions, one for the TMS320C10 simulator and one for the TMS32020 and TMS320C25 simulators.

A program must be assembled and linked using the TMS320C10, TMS32020, or TMS320C25 Macro Assembler/Link Editor programs before it can be tested and debugged. After a TMS320C10, TMS32020, or TMS320C25 object file has been developed by the assembler, it is necessary to test the program on the simulator in order to verify that it executes correctly.

The program to be debugged is a 5 tap FIR filter as presented in the book, *Digital Signal Processing Applications with the TMS320 Family.* The FIR filter is simply a finite length weighted sum of the present and previous inputs to the filter. The equation of the filter is written as

$$x(n-4)h(4) + x(n-3)h(3) + x(n-2)h(2) + x(n-1)h(1) + x(n)h(0) = y(n)$$

Figure 4-1 gives an example of a length-5 direct-form FIR filter.

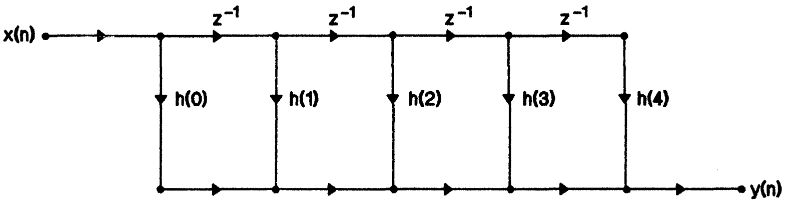


Figure 4-1. Length-5 Direct-Form FIR Filter

These debugging sessions follow this sequence of steps:

1) Test the program on the simulator for accuracy.
2) If an error is indicated, locate the error.
3) Correct the error.
4) Test the revised program for accuracy.

These debugging examples can be found on the following pages.

4.1 TMS320C10 Debugging Example

The TMS320C10 program in this debugging example attempts to perform a simple 5 tap FIR filter of the equation

$$x(n-4)h(4) + x(n-3)h(3) + x(n-2)h(2) + x(n-1)h(1) + x(n)h(0) = y(n)$$

However, there is a "bug" in the program. The simulator will be used to detect the "bug" and correct the code. The sequence of steps in a debugging session is followed for this TMS320C10 example.

In the program shown in Figure 4-2, the TMS320C10 will prompt the user for an input. The value 6 should be entered, and it will be stored at location XN0. A breakpoint will be set to determine the output of the first pass of the filter. According to the program the first data sample, 6, should be multiplied by the first filter coefficient, 3, giving an output of 12. Although it may be obvious where the "bug" is, the simulator will be used to discover it and correct the program.

```
       AORG   10       ;LOAD PROGRAM STARTING AT LOCATION
10.
 YN    EQU    50       ;OUTPUT ADDRESS.
 XN0   EQU    51       ;FIRST INPUT ADDRESSES.
 XN4   EQU    55       ;LAST INPUT ADDRESSES.
       LARP   0
 NXTIN IN     XN0,PA0 ;GET NEW INPUT VALUE XN FROM PORT
PA0.
       LARK   AR0,XN0 ;STORE LOCATION AT XN0.
       ZAC             ;ZERO ACCUMULATOR.
 *
       LTD    *+       ;3x(n);h(0)=2.
       MPYK   2
       LTD    *+       ;4x(n-1);h(1)=3.
       MPYK   3
       LTD    *+       ;3x(n-2);h(2)=4.
       MPYK   4
       LTD    *+       ;2x(n-3);h(3)=3.
       MPYK   3
       LT     *+       ;5x(n-4);h(4)=2.
       MPYK   2
 *
       APAC            ;ADD RESULT OF LAST MULTIPLY TO
 *                     ;ACCUMULATOR.
       SACL   YN       ;STORE RESULT IN YN.
       OUT    YN,PA1   ;OUTPUT THE RESPONSE TO PA1.
       B      NXTIN    ;GET THE NEXT INPUT VALUE.
       END
```

Figure 4-2. TMS320C10 Test Program

1) Test the program on the simulator for accuracy.

Figure 4-3, page 4-4 illustrates the following procedure used to test the program.

- Begin simulator execution.
- Choose either the microprocessor or microcomputer mode of operation.
- Load the program.
- Name the file in which the object file is stored.
- Enter the breakpoints.
- Run the program; enter <CR>.
- Enter the input values.
- View the processor status on the screen.
- Note the error (if indicated).

To begin testing the program, the simulator begins by entering the proper command (see Section 2 for execution verification). The simulator next prompts that either 0 or 1 be entered to place the simulator in the microprocessor or microcomputer mode, respectively. The microprocessor mode is selected, so 0 is entered. The default also places the simulator in microprocessor mode.

The L (load) command is next entered to load the program into the simulator. The simulator prompts for the file name in which the object file is stored. BUG10.MPO is entered since this is the file name for this program. The simulator executes all of the instructions it encounters until it is halted by either a <CR>, an interrupt, a branch to self, a set breakpoint, or an error. Since a breakpoint will be set at the OUT instruction, the simulator will stop further instruction execution.

When the simulator encounters a breakpoint, it displays the processor status. At this time, it can be seen if the program has executed as desired. This program did not execute correctly since the output is 84 instead of 12. At this point a RAMH command should be entered to check the data RAM. Since the value 6 is in all input value locations, we can conclude that the first input sample is being incorrectly multiplied by all filter coefficients.

```
                        SIMULATION OF THE TMS320C10
                               VERSION # 2.0

0 - MICROPROCESSOR MODE (ADDR 0-1535, OFF CHIP)
1 - MICROCOMPUTER  MODE (ADDR 0-1535, ON  CHIP)

ENTER VALUE TO SELECT MODE OF OPERATION
0 <CR>

YOU ARE IN THE MICROPROCESSOR MODE (ADDR 0-1535, OFF CHIP)

ENTER COMMAND (HELP=<CR>):
L <CR>

ENTER A NEW OBJECT FILE
BUG10.MPO <CR>
* * * * LOADING PROGRAM "NO$IDT  " * * * *

ENTER COMMAND (HELP=<CR>):
BIAQ <CR>

BREAK ON INSTRUCTION ACQUISITION
ENTER THE ADDRESS (IN HEX)
1A <CR>

ENTER COMMAND (HELP=<CR>):
RUN <CR>

>>PC=     B     OPCODE=4033     IN          PREVIOUS PC=     A

            ARP    ARO     AR1     TREG     PREG     ACC      CLK
INTEGER      0      0       0       0        0        0        2
HEX          0      0       0       0        0        0        2

>>STK=    0   0    0    0    DP = 0  INTF= 0  OV = 0
                             BIO= 1  INTM= 0  OVM= 0

ENTER INPUT VALUE (IN HEX) OR "-" TO RETURN TO MAIN
6 <CR>
* * * OUTPUT VALUE (IN HEX) IS 54

>>PC=     1A    OPCODE=4932     OUT         PREVIOUS PC=    19

            ARP    ARO     AR1     TREG     PREG     ACC      CLK
INTEGER      0      56      0       6        12       84       19
HEX          0      38      0       6        0C       54       13

>>STK=    0   0    0    0    DP = 0  INTF= 0  OV = 0
                             BIO= 1  INTM= 0  OVM= 0
>>> INSTRUCTION AQUISITION BREAK POINT # 1 <<<
```

Figure 4-3. Error Revealed in Tested TMS320C10 Program

```
ENTER COMMAND (HELP=<CR>):
RAMH <CR>
              0       1       2       3       4       5       6       7
              8       9       A       B       C       D       E       F

       0      0       0       0       0       0       0       0       0
       8      0       0       0       0       0       0       0       0
      10      0       0       0       0       0       0       0       0
      18      0       0       0       0       0       0       0       0
      20      0       0       0       0       0       0       0       0
      28      0       0       0       0       0       0       0       0
      30      0       0       5A      6       6       6       6       6
      38      0       0       0       0       0       0       0       0
      40      0       0       0       0       0       0       0       0
      48      0       0       0       0       0       0       0       0
      50      0       0       0       0       0       0       0       0
      58      0       0       0       0       0       0       0       0
      60      0       0       0       0       0       0       0       0
      68      0       0       0       0       0       0       0       0
      70      0       0       0       0       0       0       0       0
      78      0       0       0       0       0       0       0       0
      80      0       0       0       0       0       0       0       0
      88      0       0       0       0       0       0       0       0
```

Figure 4-3. Error Revealed in Tested TMS320C10 Program (Concluded)

2) Locate the error.

Figure 4-4, pages 4-6 to 4-8, illustrates the following procedure.

- Clear the P register.
- Clear the T register.
- Reset the simulator.
- Set an additional breakpoint.
- Single-step through part of the program, viewing the results of each instruction.
- When an error is located, inspect the memory locations for that instruction.
- Locate the error in memory.

Since the program did not result with the desired number for an output, the next step is to locate the error. To reset the program the P register and T register must be cleared. This is done by entering P <CR>. After being prompted for a new value, enter 0. Do the same for the T register. Next, the simulator is reset using the RS command, and an additional breakpoint is added at the ZAC instruction (address D). From this point the program will be single stepped through to determine the cause of the error. In this pass the value 5 should be entered for an input value so that it can be distinguished form the previous input value.

```
ENTER COMMAND (HELP=<CR>):
P <CR>

PRESENT VALUE OF THE P REGISTER
>  1E
ENTER NEW VALUE (IN HEX)
0 <CR>

ENTER COMMAND (HELP=<CR>):
T <CR>

PRESENT VALUE OF THE T REGISTER
>  6
ENTER NEW VALUE (IN HEX)
0 <CR>

ENTER COMMAND (HELP=<CR>):
RS  <CR>

ENTER COMMAND (HELP=<CR>):
BIAQ  <CR>

BREAK ON INSTRUCTION ACQUISITION
ENTER THE ADDRESS (IN HEX)
D <CR>

ENTER COMMAND (HELP=<CR>):
R <CR>

>>PC=     B     OPCODE=4033    IN             PREVIOUS PC=     A

          ARP      AR0      AR1     TREG      PREG      ACC      CLK
INTEGER    0        54       0        0         0        54       29
HEX        0        36       0        0         0        36       1D

>>STK=   0   0    0    0    DP  = 0  INTF= 0  OV  = 0
                              BIO= 1  INTM= 1  OVM= 0

ENTER INPUT VALUE (IN HEX) OR "-" TO RETURN TO MAIN
5 <CR>


>>PC=     D     OPCODE=7F89    ZAC            PREVIOUS PC=     C

          ARP      AR0      AR1     TREG      PREG      ACC      CLK
INTEGER    0        51       0        0         0        0        34
HEX        0        33       0        0         0        0        22

>>STK=   0   0    0    0    DP  = 0  INTF= 0  OV  = 0
                              BIO= 1  INTM= 1  OVM= 0

>>> INSTRUCTION ACQUISITION BREAK POINT # 2  <<<
```

Figure 4-4. Single-Stepping Through the TMS320C10 Program

Next the data memory is checked to insure that the first data sample has been written into the correct data RAM location. Using the RAMH command we see that the 5 was indeed written into location 33 (hex). Single stepping is continued. The next two instructions, LTD and MPYK appear correct. However, the next LTD is incorrect because a 6, not a 5, should be loaded into the T register.

```
ENTER COMMAND (HELP=<CR>):
RAMH <CR>

                 0      1      2      3      4      5      6      7
                 8      9      A      B      C      D      E      F

           0     0      0      0      0      0      0      0      0
           8     0      0      0      0      0      0      0      0
          10     0      0      0      0      0      0      0      0
          18     0      0      0      0      0      0      0      0
          20     0      0      0      0      0      0      0      0
          28     0      0      0      0      0      0      0      0
          30     0      0      5A     5      6      6      6      6
          38     0      0      0      0      0      0      0      0
          40     0      0      0      0      0      0      0      0
          48     0      0      0      0      0      0      0      0
          50     0      0      0      0      0      0      0      0
          58     0      0      0      0      0      0      0      0
          60     0      0      0      0      0      0      0      0
          68     0      0      0      0      0      0      0      0
          70     0      0      0      0      0      0      0      0
          78     0      0      0      0      0      0      0      0
          80     0      0      0      0      0      0      0      0
          88     0      0      0      0      0      0      0      0
          90     0      0      0      0      0      0      0      0

ENTER COMMAND (HELP=<CR>):
SS <CR>

>>PC=    E     OPCODE=6BA8    LTD            PREVIOUS PC=     D

          ARP     ARO     AR1     TREG     PREG     ACC     CLK
INTEGER    0       52      0        5        0       0       35
HEX        0       34      0        5        0       0       23

>>STK=   0   0    0    0    DP = 0   INTF= 0   OV = 0
                            BIO= 1   INTM= 1   OVM= 0

ENTER <CR> TO CONTINUE
      "-" TO RETURN TO MAIN
<CR>

>>PC=    F     OPCODE=8003    MPYK           PREVIOUS PC=     E

          ARP     ARO     AR1     TREG     PREG     ACC     CLK
INTEGER    0       52      0        5       10       0       36
HEX        0       34      0        5        A       0       24

>>STK=   0   0    0    0    DP = 0   INTF= 0   OV = 0
                            BIO= 1   INTM= 1   OVM= 0

ENTER <CR> TO CONTINUE
      "-" TO RETURN TO MAIN
```

Figure 4-4. Single-Stepping Through the TMS320C10 Program (Continued)

```
<CR>

>>PC=   10      OPCODE=6BA8     LTD              PREVIOUS PC=       F

            ARP       AR0       AR1      TREG       PREG      ACC      CLK
INTEGER      0         53        0         5         10        10       37
HEX          0         35        0         5          A         A       25

>>STK=   0   0    0    0     DP = 0   INTF= 0   OV = 0
                              BIO= 1   INTM= 1   OVM= 0

ENTER <CR> TO CONTINUE
      "-" TO RETURN TO MAIN
```

Figure 4-4. Single-Stepping Through the TMS320C10 Program (Concluded)

It is now necessary to inspect the memory locations from which the LTD instruction fetches its data. The input values stored in locations >33 to >36 are expected to be 5,6,6,6, respectively, since the old data values were never cleared out of RAM. The RAMH command is entered and when these locations are displayed, a 5 is in memory locations >33 through >35. By carefully examining the program, it becomes clear that the value 5 is writing over the next higher address before it gets multiplied in the next filter tap.

```
ENTER COMMAND (HELP=<CR>):
RAMH <CR>

        0      1      2      3      4      5      6      7
        8      9      A      B      C      D      E      F

  0     0      0      0      0      0      0      0      0
  8     0      0      0      0      0      0      0      0
 10     0      0      0      0      0      0      0      0
 18     0      0      0      0      0      0      0      0
 20     0      0      0      0      0      0      0      0
 28     0      0      0      0      0      0      0      0
 30     0      0      5A     5      5      5      6      6
 38     0      0      0      0      0      0      0      0
 40     0      0      0      0      0      0      0      0
 48     0      0      0      0      0      0      0      0
 50     0      0      0      0      0      0      0      0
 58     0      0      0      0      0      0      0      0
 60     0      0      0      0      0      0      0      0
 68     0      0      0      0      0      0      0      0
 70     0      0      0      0      0      0      0      0
 78     0      0      0      0      0      0      0      0
 80     0      0      0      0      0      0      0      0
 88     0      0      0      0      0      0      0      0

ENTER COMMAND (HELP=<CR>):
Q <CR>
```

Figure 4-5. Display of TMS320C10 Memory Locations

3) Correct the error.

Since the error is now located, the code can be modified by changing all of the LTD *+ instructions to LTD *- instructions. This is because the multiplications must start with the oldest data sample in the filter since the next higher address is written over before the next multiplication. The LARK AR0,XN0 must also be changed to LARK AR0,XN4 to begin at the last address for the filter taps. Finally, the LT instruction is changed to LTD, and the first LTD instruction is changed to LT.

```
      AORG  10         ;LOAD PROGRAM STARTING AT LOCATION 10.
YN    EQU   50         ;OUTPUT ADDRESS.
XN0   EQU   51         ;FIRST INPUT ADDRESSES.
XN4   EQU   55         ;LAST INPUT ADDRESSES.
      LARP  0
NXTIN IN    XN0,PA0    ;GET NEW INPUT VALUE XN FROM PORT PA0.
      LARK  AR0,XN4    ;STORE LOCATION AT XN0.
      ZAC              ;ZERO ACCUMULATOR.
*
      LT    *-         ;3x(n-4);h(0)=2.
      MPYK  2
      LTD   *-         ;4x(n-3);h(1)=3.
      MPYK  3
      LTD   *-         ;3x(n-2);h(2)=4.
      MPYK  4
      LTD   *-         ;2x(n-1);h(3)=3.
      MPYK  3
      LTD   *-         ;5x(n);h(4)=2.
      MPYK  2
*
      APAC             ;ADD RESULT OF LAST MULTIPLY TO
*                      ;ACCUMULATOR.
      SACL  YN         ;STORE RESULT IN YN.
      OUT   YN,PA1     ;OUTPUT THE RESPONSE TO PA1.
      B     NXTIN      ;GET THE NEXT INPUT VALUE.
      END
```

Figure 4-6. Revised TMS320C10 Program

4) Test the revised program for accuracy.

Now that code has been revised, the program is tested again using the same procedure sequence as outlined for Step 1. Simulator execution begins and the assembled revised code is loaded into the simulator. The RUN command is entered for the instructions to be executed. Upon entering a 6 when prompted for an input value, the output value is a 30, which is the input value multiplied by the first filter coefficient. After examining the RAM and finding the value 6 in memory locations >33 and >34 as it should be to prepare for the next filter input, a C (continue) can be entered to examine the results of more inputs. In this example 5, 3, and 8 are entered with the results of 37, 43, and 85, respectively.

```
                    SIMULATION OF THE TMS320C10
                           VERSION # 2.0

0 - MICROPROCESSOR MODE (ADDR 0-1535, OFF CHIP)
1 - MICROCOMPUTER  MODE (ADDR 0-1535, ON  CHIP)

ENTER VALUE TO SELECT MODE OF OPERATION
0 <CR>

YOU ARE IN THE MICROPROCESSOR MODE (ADDR 0-1535, OFF CHIP)

ENTER COMMAND (HELP=<CR>):
L <CR>

ENTER A NEW OBJECT FILE
BUG10.MPO <CR>
* * * * LOADING PROGRAM "NO$IDT  " * * * *

ENTER COMMAND (HELP=<CR>):
BIAQ <CR>

BREAK ON INSTRUCTION ACQUISITION
ENTER THE ADDRESS (IN HEX)
1A <CR>

ENTER COMMAND (HELP=<CR>):
RUN <CR>

>>PC=    B     OPCODE=4033     IN          PREVIOUS PC=    A

          ARP    ARO    AR1    TREG     PREG     ACC      CLK
INTEGER    0      0      0      0        0        0        2
HEX        0      0      0      0        0        0        2

>>STK=   0   0    0    0    DP = 0  INTF= 0  OV = 0
                            BIO= 1  INTM= 0  OVM= 0

ENTER INPUT VALUE (IN HEX) OR "-" TO RETURN TO MAIN
6 <CR>
* * * OUTPUT VALUE (IN HEX) IS C

>>PC=     1A   OPCODE=4932     OUT         PREVIOUS PC=   19

          ARP    ARO    AR1    TREG     PREG     ACC      CLK
INTEGER    0      50     0      6        12       12       19
HEX        0      32     0      6        C        C        13

>>STK=   0   0    0    0    DP = 0  INTF= 0  OV = 0
                            BIO= 1  INTM= 0  OVM= 0

>>> INSTRUCTION AQUISITION BREAK POINT # 1 <<<
```

Figure 4-7. Testing the Revised TMS320C10 Program

```
ENTER COMMAND (HELP=<CR>):
RAMH <CR>

                   0      1      2      3      4      5      6      7
                   8      9      A      B      C      D      E      F

            0      0      0      0      0      0      0      0      0
            8      0      0      0      0      0      0      0      0
           10      0      0      0      0      0      0      0      0
           18      0      0      0      0      0      0      0      0
           20      0      0      0      0      0      0      0      0
           28      0      0      0      0      0      0      0      0
           30      0      0      C      6      6      0      0      0
           38      0      0      0      0      0      0      0      0
           40      0      0      0      0      0      0      0      0
           48      0      0      0      0      0      0      0      0
           50      0      0      0      0      0      0      0      0
           58      0      0      0      0      0      0      0      0
           60      0      0      0      0      0      0      0      0
           68      0      0      0      0      0      0      0      0
           70      0      0      0      0      0      0      0      0
           78      0      0      0      0      0      0      0      0
           80      0      0      0      0      0      0      0      0
           88      0      0      0      0      0      0      0      0

ENTER COMMAND (HELP=<CR>):
C <CR>

>>PC=     B     OPCODE=4033     IN           PREVIOUS PC=    1B

         ARP    ARO     AR1    TREG      PREG     ACC      CLK
INTEGER   0      50      0        6        12      12       22
HEX       0      32      0        6         C       C       16

>>STK=   0   0    0    0    DP = 0  INTF= 0  OV = 0
                            BIO= 1  INTM= 0  OVM= 0

ENTER INPUT VALUE (IN HEX) OR "-" TO RETURN TO MAIN
5 <CR>
* * * OUTPUT VALUE (IN HEX) IS 1C

>>PC=     1A    OPCODE=4932     OUT          PREVIOUS PC=    19

         ARP    ARO     AR1    TREG      PREG     ACC      CLK
INTEGER   0      50      0        5        10      28       39
HEX       0      32      0        5         A      1C       27

>>STK=   0   0    0    0    DP = 0  INTF= 0  OV = 0
                            BIO= 1  INTM= 0  OVM= 0

>>> INSTRUCTION AQUISITION BREAK POINT # 1 <<<
```

Figure 4-7. Testing the Revised TMS320C10 Program (Continued)

```
ENTER COMMAND (HELP=<CR>):
RAMH <CR>

                  0      1      2      3      4      5      6      7
                  8      9      A      B      C      D      E      F

             0    0      0      0      0      0      0      0      0
             8    0      0      0      0      0      0      0      0
            10    0      0      0      0      0      0      0      0
            18    0      0      0      0      0      0      0      0
            20    0      0      0      0      0      0      0      0
            28    0      0      0      0      0      0      0      0
            30    0      0      1C     5      5      6      0      0
            38    0      0      0      0      0      0      0      0
            40    0      0      0      0      0      0      0      0
            48    0      0      0      0      0      0      0      0
            50    0      0      0      0      0      0      0      0
            58    0      0      0      0      0      0      0      0
            60    0      0      0      0      0      0      0      0
            68    0      0      0      0      0      0      0      0
            70    0      0      0      0      0      0      0      0
            78    0      0      0      0      0      0      0      0
            80    0      0      0      0      0      0      0      0
            88    0      0      0      0      0      0      0      0

ENTER COMMAND (HELP=<CR>):
C <CR>

>>PC=     B     OPCODE=4033     IN           PREVIOUS PC=     1B

           ARP    AR0     AR1    TREG     PREG     ACC      CLK
INTEGER     0      50      0       5       10       28       42
HEX         0      32      0       5        A       1C       2A

>>STK=   0   0    0    0    DP = 0  INTF= 0  OV = 0
                            BIO= 1  INTM= 0  OVM= 0

ENTER INPUT VALUE (IN HEX) OR "-" TO RETURN TO MAIN
3 <CR>
* * * OUTPUT VALUE (IN HEX) IS 2D

>>PC=     1A    OPCODE=4932     OUT          PREVIOUS PC=     19

           ARP    AR0     AR1    TREG     PREG     ACC      CLK
INTEGER     0      50      0       3        6       45       59
HEX         0      32      0       3        6       2D       3B

>>STK=   0   0    0    0    DP = 0  INTF= 0  OV = 0
                            BIO= 1  INTM= 0  OVM= 0

>>> INSTRUCTION AQUISITION BREAK POINT # 1 <<<
```

Figure 4-7. Testing the Revised TMS320C10 Program (Continued)

```
ENTER COMMAND (HELP=<CR>):
RAMH <CR>

                 0      1      2      3      4      5      6      7
                 8      9      A      B      C      D      E      F

          0      0      0      0      0      0      0      0      0
          8      0      0      0      0      0      0      0      0
         10      0      0      0      0      0      0      0      0
         18      0      0      0      0      0      0      0      0
         20      0      0      0      0      0      0      0      0
         28      0      0      0      0      0      0      0      0
         30      0      0      2D     3      3      5      6      0
         38      0      0      0      0      0      0      0      0
         40      0      0      0      0      0      0      0      0
         48      0      0      0      0      0      0      0      0
         50      0      0      0      0      0      0      0      0
         58      0      0      0      0      0      0      0      0
         60      0      0      0      0      0      0      0      0
         68      0      0      0      0      0      0      0      0
         70      0      0      0      0      0      0      0      0
         78      0      0      0      0      0      0      0      0
         80      0      0      0      0      0      0      0      0
         88      0      0      0      0      0      0      0      0

ENTER COMMAND (HELP=<CR>):
C <CR>

>>PC=    B      OPCODE=4033      IN           PREVIOUS PC=     1B

           ARP    ARO      AR1     TREG      PREG      ACC       CLK
INTEGER     0     50        0         3         6       45        62
HEX         0     32        0         3         6       2D        3E

>>STK=   0   0    0    0     DP = 0  INTF= 0  OV = 0
                             BIO= 1  INTM= 0  OVM= 0

ENTER INPUT VALUE (IN HEX) OR "-" TO RETURN TO MAIN
8 <CR>
* * * OUTPUT VALUE (IN HEX) IS 3F

>>PC=     1A    OPCODE=4932      OUT          PREVIOUS PC=     19

           ARP    ARO      AR1     TREG      PREG      ACC       CLK
INTEGER     0     50        0         8        16       63        79
HEX         0     32        0         8        10       3F        4F

>>STK=   0   0    0    0     DP = 0  INTF= 0  OV = 0
                             BIO= 1  INTM= 0  OVM= 0

>>> INSTRUCTION AQUISITION BREAK POINT # 1 <<<
```

Figure 4-7. Testing the Revised TMS320C10 Program (Continued)

```
ENTER COMMAND (HELP=<CR>):
RAMH <CR>

                 0     1     2     3     4     5     6     7
                 8     9     A     B     C     D     E     F

            0    0     0     0     0     0     0     0     0
            8    0     0     0     0     0     0     0     0
           10    0     0     0     0     0     0     0     0
           18    0     0     0     0     0     0     0     0
           20    0     0     0     0     0     0     0     0
           28    0     0     0     0     0     0     0     0
           30    0     0     3F    8     8     3     5     6
           38    0     0     0     0     0     0     0     0
           40    0     0     0     0     0     0     0     0
           48    0     0     0     0     0     0     0     0
           50    0     0     0     0     0     0     0     0
           58    0     0     0     0     0     0     0     0
           60    0     0     0     0     0     0     0     0
           68    0     0     0     0     0     0     0     0
           70    0     0     0     0     0     0     0     0
           78    0     0     0     0     0     0     0     0
           80    0     0     0     0     0     0     0     0
           88    0     0     0     0     0     0     0     0

ENTER COMMAND (HELP=<CR>):
Q <CR>
```

Figure 4-7. Testing the Revised TMS320C10 Program (Concluded)

4.2 TMS32020/C25 Debugging Example

The TMS32020/C25 program in this debugging example attempts to perform a simple 5 tap FIR filter of the equation

x(n-4)h(4) + x(n-3)h(3) + x(n-2)h(2) + x(n-1)h(1) + x(n)h(0) = y(n)

However, there is a "bug" in the program. The simulator will be used to detect the "bug" and correct the code. The sequence of steps in a debugging session is followed for this TMS32020/C25 example.

In the program shown in Figure 4-8, the TMS32020/C25 will prompt the user for an input. The value 6 should be entered, and it will be stored at location XN. A breakpoint will be set to determine the output of the first pass of the filter. According to the program the first data sample, 6, should be multiplied by the first filter coefficient, 2, giving an output of 12. Although it may be obvious where the "bug" is, the simulator will be used to discover it and correct the program.

```
      AORG  >32        ;LOAD PROGRAM STARTING AT LOCATION 32.
YN    EQU   5          ;OUTPUT ADDRESS.
XN    EQU   0          ;FIRST INPUT ADDRESSES.
XNM4  EQU   4          ;LAST INPUT ADDRESSES.
      LDPK  4          ;POINT TO DATA PAGE 4.
*
HX    DATA  2,3,4,3,2
*
      LARP  AR0        ;USE AR0 FOR INDIRECT ADDRESSING.
      LRLK  AR0,>200   ;POINT TO BLOCK B0.
      RPTK  XNM4       ;N-1 NUMBER OF COEFFICIENTS.
      BLKP  HX,*+
*
      CNFP             ;USE BLOCK B0 AS PROGRAM AREA.
      LDPK  6          ;POINT TO DATA PAGE 6.
NXTIN IN    XN,PA0     ;GET NEW SAMPLE.
      LRLK  AR1,>300   ;POINT TO THE STARTING ADDRESS.
      LARP  AR1        ;OF DATA SAMPLES IN BLOCK B1.
*
      MPYK  0          ;SET P REG TO 0.
      ZAC              ;CLEAR ACCUMULATOR.
      RPTK  XNM4       ;REPEAT N-1 TIMES.
      MACD  >FF00,*+   ;MULTIPLY/ACCUMULATE.
      APAC             ;ADD RESULT OF LAST MULTIPLY TO
*                      ;ACCUMULATOR.
      SACL  YN         ;STORE RESULT IN YN.
      OUT   YN,PA1     ;OUTPUT THE RESPONSE TO PA1.
      B     NXTIN      ;GET THE NEXT FILTER COEFFICIENT.
      END
```

Figure 4-8. TMS32020/C25 Test Program

1) Test the program on the simulator for accuracy.

Figure 4-9, page 4-17, illustrates the following procedure, used to test the program.

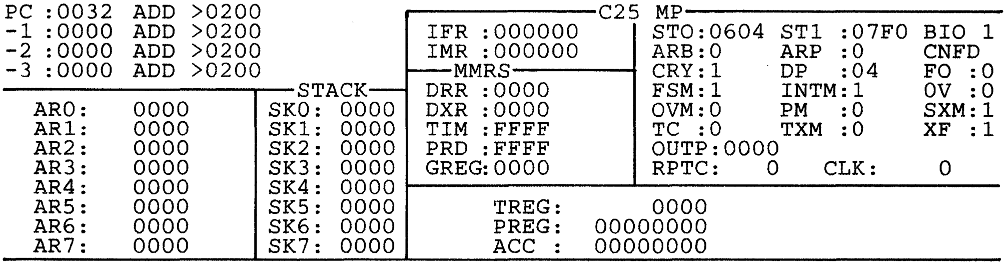
a) Begin simulator execution.
b) Configure block B0.
c) Load the program.
d) Name the file in which the object file is stored.
e) Set program counter.
f) Set breakpoints.
g) Run the program; enter <CR>.
h) Enter input value.
i) View the processor status on the screen.
j) Note the error (if indicated).

To begin testing the program, initiate simulator execution by entering the proper command (see Section 2 for execution verification). The simulator next prompts that either a 0 or a 1 be entered to configure the first 4K words as internal or external program ROM. Since this program performs the correct configuration when desired, a <CR> is entered to default. The default configures the first 4K words as internal program ROM.

Next, the L (Load) command is entered to load the program into the simulator (see Figure 4-9). The simulator prompts for the filename in which the object file is stored. BUGC25.MPO is entered since this is the filename for this program. The simulator executes all of the instructions it encounters until it is halted by either a <CR>, an interrupt, a branch to self, a set breakpoint, or an error. Since a breakpoint will be set at the OUT instruction, the simulator will stop further instruction execution.

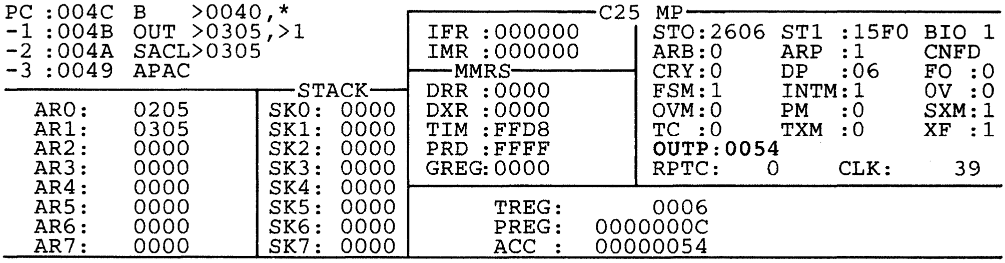
When the simulator encounters a breakpoint, it displays the processor status. At this time, it can be seen if the program has executed as desired. This program did not execute correctly since the output is 84 instead of 12. At this point a RAMH command should be entered to check the data RAM. Since the value 6 is in all input value locations, we can conclude that the first input sample is being incorrectly multiplied by all filter coefficients.

```
                  (C)   COPYRIGHT  TEXAS INSTRUMENTS
                          INCORPORATED 1986
                         TMS320C25 SIMULATOR
                         RELEASE 1.0  86.194

0 : First 4k words INTERNAL program ROM
1 : First 4k words EXTERNAL program ROM

Enter the memory configuration (0 or 1):  <CR>

First 4k WORDS are mapped on Internal ROM

COMMAND:  L <CR>

Enter a new object file  :  BUGC25.MPO <CR>
**** LOADING PROGRAM "NO$IDT   " ****

COMMAND:  PC <CR>

Present value for program counter :   0
Enter a new value for the PC (in Hex) :  32 <CR>

COMMAND:  BIAQ <CR>
BREAK ON INSTRUCTION ACQUISITION
ENTER THE ADDRESS (IN HEX): 4B <CR>

COMMAND:  RUN <CR>
```



```
     PC :0032 ADD >0200        ┌───────────C25 MP─────────────────────────
     -1 :0000 ADD >0200        │ IFR :000000  │ STO:0604 ST1  :07F0 BIO 1
     -2 :0000 ADD >0200        │ IMR :000000  │ ARB:0    ARP  :0    CNFD
     -3 :0000 ADD >0200        ├────MMRS──────┤ CRY:1    DP   :04   FO :0
     ───────────────────┬──STACK──┤ DRR :0000    │ FSM:1    INTM:1     OV :0
      AR0:    0000      │SK0: 0000│ DXR :0000    │ OVM:0    PM   :0    SXM:1
      AR1:    0000      │SK1: 0000│ TIM :FFFF    │ TC  :0   TXM  :0    XF :1
      AR2:    0000      │SK2: 0000│ PRD :FFFF    │ OUTP:0000
      AR3:    0000      │SK3: 0000│ GREG:0000    │ RPTC:    0    CLK:     0
      AR4:    0000      │SK4: 0000├──────────────────────────────────────────
      AR5:    0000      │SK5: 0000│     TREG:       0000
      AR6:    0000      │SK6: 0000│     PREG:   00000000
      AR7:    0000      │SK7: 0000│     ACC :   00000000
     ───────────────────┴─────────┴──────────────────────────────────────────
Enter input value (in HEX) or "-" to return to main: 6 <CR>
```

After entering the value of 6, the screen displays the following:



```
     PC :004C B   >0040,*      ┌───────────C25 MP─────────────────────────
     -1 :004B OUT >0305,>1     │ IFR :000000  │ STO:2606 ST1  :15F0 BIO 1
     -2 :004A SACL>0305        │ IMR :000000  │ ARB:0    ARP  :1    CNFD
     -3 :0049 APAC             ├────MMRS──────┤ CRY:0    DP   :06   FO :0
     ───────────────────┬──STACK──┤ DRR :0000    │ FSM:1    INTM:1     OV :0
      AR0:    0205      │SK0: 0000│ DXR :0000    │ OVM:0    PM   :0    SXM:1
      AR1:    0305      │SK1: 0000│ TIM :FFD8    │ TC  :0   TXM  :0    XF :1
      AR2:    0000      │SK2: 0000│ PRD :FFFF    │ OUTP:0054
      AR3:    0000      │SK3: 0000│ GREG:0000    │ RPTC:    0    CLK:    39
      AR4:    0000      │SK4: 0000├──────────────────────────────────────────
      AR5:    0000      │SK5: 0000│     TREG:       0006
      AR6:    0000      │SK6: 0000│     PREG:   0000000C
      AR7:    0000      │SK7: 0000│     ACC :   00000054
     ───────────────────┴─────────┴──────────────────────────────────────────
>> instruction acquisition breakpoint # 1 <<
```

Figure 4-9. Error Revealed in Tested TMS32020/C25 Program

```
COMMAND:  RAMH <CR>

Enter start DATA address (in Hex):  300 <CR>

         300  0006  0006  0006  0006  0006  0054  0000  0000
         308  0000  0000  0000  0000  0000  0000  0000  0000
         310  0000  0000  0000  0000  0000  0000  0000  0000
         318  0000  0000  0000  0000  0000  0000  0000  0000
         320  0000  0000  0000  0000  0000  0000  0000  0000
         328  0000  0000  0000  0000  0000  0000  0000  0000
         330  0000  0000  0000  0000  0000  0000  0000  0000
         338  0000  0000  0000  0000  0000  0000  0000  0000

COMMAND:
```

Figure 4-9. Error Revealed in Tested TMS32020/C25 Program (Concluded)

2) Locate the error.

Figure 4-10, pages 4-18 through 4-20, illustrates the following procedure.

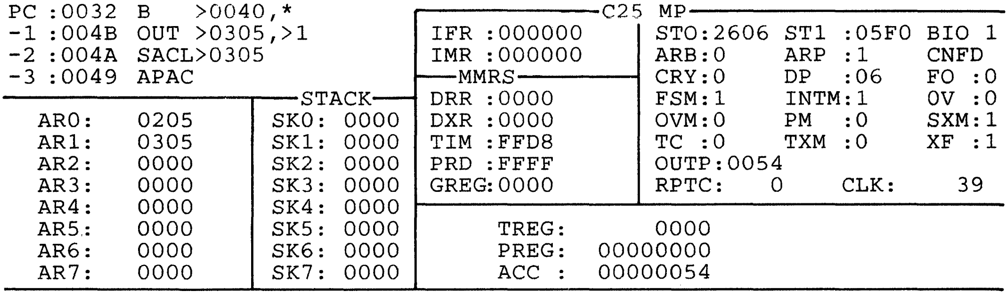
- Enter a new PC value.
- Single step through the program, viewing the results of each instruction.
- When an error is located, inspect the memory locations for that instruction.
- Locate the error in memory.

Since the program did not result with the desired number for an output, the next step is to locate the error. To reset the program the P register and T register must be cleared. This is done by entering P <CR>. After being prompted for a new value, enter 0. Do the same for the T register. Next, the simulator is reset using the RS command, and an additional breakpoint is added at the LACK instruction (address 45). From this point the program will be single stepped through to determine the cause of the error. In this pass the value 5 should be entered for an input value so that it can be distinguished form the previous input value.

```
COMMAND:  PC <CR>

Present value for program counter :  004C
Enter a new value for the PC (in Hex) :  32 <CR>

COMMAND:  CNF <CR>

Present value of ram configuration control bit :0001
 0 - Onchip memory block B0 is Data      RAM
 1 - Onchip memory block B0 is Program   ROM
Enter new value :  0 <CR>

COMMAND:  P <CR>
Present value of the P register: 000C
Enter new value (in HEX): 0 <CR>

COMMAND: T <CR>
Present value of the T register: 0006
Enter new value (in HEX): 0 <CR>
```

Figure 4-10. Single-Stepping Through the TMS32020/C25 Program

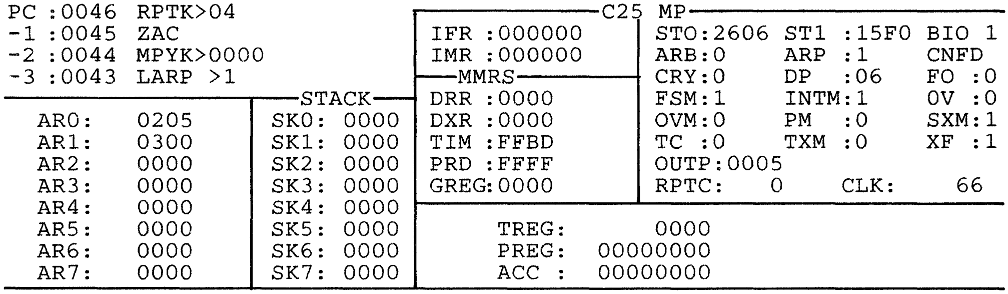
```
COMMAND: BIAQ <CR>
Break on instruction acquisition
Enter the address (in HEX): 45 <CR>

COMMAND: RUN <CR>
```



```
PC :0032  B    >0040,*
-1 :004B  OUT >0305,>1          ---------------C25 MP-------------------------------
-2 :004A  SACL>0305              IFR :000000       | STO:2606  ST1  :05FO  BIO 1
-3 :0049  APAC                   IMR :000000       | ARB:0     ARP  :1     CNFD
                                 ----MMRS----------| CRY:0     DP   :06    FO :0
------------------------ STACK-- DRR :0000         | FSM:1     INTM:1      OV :0
  ARO:     0205   | SKO:  0000 | DXR :0000         | OVM:0     PM   :0     SXM:1
  AR1:     0305   | SK1:  0000 | TIM :FFD8         | TC  :0    TXM  :0     XF :1
  AR2:     0000   | SK2:  0000 | PRD :FFFF         | OUTP:0054
  AR3:     0000   | SK3:  0000 | GREG: 0000        | RPTC:    0     CLK:      39
  AR4:     0000   | SK4:  0000 |---------------------------------------------------
  AR5.:    0000   | SK5:  0000 |      TREG:      0000
  AR6:     0000   | SK6:  0000 |      PREG:  00000000
  AR7:     0000   | SK7:  0000 |      ACC :  00000054
```

Enter input value (in HEX) or "-" to return to main: 5 <CR>

After entering the value 5, the screen displays the following:



```
PC :0046  RPTK>04
-1 :0045  ZAC                   ---------------C25 MP-------------------------------
-2 :0044  MPYK>0000              IFR :000000       | STO:2606  ST1  :15FO  BIO 1
-3 :0043  LARP >1                IMR :000000       | ARB:0     ARP  :1     CNFD
                                 ----MMRS----------| CRY:0     DP   :06    FO :0
------------------------ STACK-- DRR :0000         | FSM:1     INTM:1      OV :0
  ARO:     0205   | SKO:  0000 | DXR :0000         | OVM:0     PM   :0     SXM:1
  AR1:     0300   | SK1:  0000 | TIM :FFBD         | TC  :0    TXM  :0     XF :1
  AR2:     0000   | SK2:  0000 | PRD :FFFF         | OUTP:0005
  AR3:     0000   | SK3:  0000 | GREG: 0000        | RPTC:    0     CLK:      66
  AR4:     0000   | SK4:  0000 |---------------------------------------------------
  AR5:     0000   | SK5:  0000 |      TREG:      0000
  AR6:     0000   | SK6:  0000 |      PREG:  00000000
  AR7:     0000   | SK7:  0000 |      ACC :  00000000
```

```
>> instruction acquisition breakpoint # 2 <<
```

Figure 4-10. Single-Stepping Through the TMS32020/C25 Program (Continued)

At this point all of the registers are correct, so a RAMH command is used to check the data RAM. The input, 5, is in the first location as expected, so the program is assumed to be correct up to this point. Next the SS (single-step) command is used. The next instruction (RPTK) appears correct. However, the MACD instruction is incorrect because a 6, not a 5, should be loaded into the T register.

```
COMMAND:  RAMH <CR>

Enter start DATA address (in Hex):  300 <CR>

        300  0005  0006  0006  0006  0006  0054  0000  0000
        308  0000  0000  0000  0000  0000  0000  0000  0000
        310  0000  0000  0000  0000  0000  0000  0000  0000
        318  0000  0000  0000  0000  0000  0000  0000  0000
        320  0000  0000  0000  0000  0000  0000  0000  0000
        328  0000  0000  0000  0000  0000  0000  0000  0000
        330  0000  0000  0000  0000  0000  0000  0000  0000
        338  0000  0000  0000  0000  0000  0000  0000  0000
```

Now, Single Step (SS) the simulator to locate the program "bug".

```
COMMAND:  SS <CR>
```

Execute RPTK instruction

```
 PC :0047  MACD>FF00,*+         ------------------C25 MP----------------------------
 -1 :0046  RPTK>04              | IFR :000000    | STO:2606  ST1 :15F0  BIO 1
 -2 :004A  ZAC                  | IMR :000000    | ARB:0     ARP :1     CNFD
 -3 :0049  MPYK>0000            |----MMRS--------| CRY:0     DP  :06    FO :0
--------------------------STACK-| DRR :0000      | FSM:1     INTM:1     OV :0
   AR0:    0205    |  SK0: 0000 | DXR :0000      | OVM:0     PM  :0     SXM:1
   AR1:    0300    |  SK1: 0000 | TIM :FFBC      | TC :0     TXM :0     XF :1
   AR2:    0000    |  SK2: 0000 | PRD :FFFF      | OUTP:0005
   AR3:    0000    |  SK3: 0000 | GREG:0000      | RPTC:   4      CLK:     67
   AR4:    0000    |  SK4: 0000 |---------------------------------------------------
   AR5:    0000    |  SK5: 0000 |       TREG:       0000
   AR6:    0000    |  SK6: 0000 |       PREG:   00000000
   AR7:    0000    |  SK7: 0000 |       ACC :   00000000
------------------------------------------------------------------------------------
<CR>:continue <->:return to main <D>:display mode

<CR>
```

Execute MACD instruction

```
 PC :0049  APAC                 ------------------C25 MP----------------------------
 -1 :0047  MACD>FF00,*+         | IFR :000000    | STO:2606  ST1 :15F0  BIO 1
 -2 :004A  RPTK>04              | IMR :000000    | ARB:0     ARP :1     CNFD
 -3 :0049  ZAC                  |----MMRS--------| CRY:0     DP  :06    FO :0
--------------------------STACK-| DRR :0000      | FSM:1     INTM:1     OV :0
   AR0:    0205    |  SK0: 0000 | DXR :0000      | OVM:0     PM  :0     SXM:1
   AR1:    0305    |  SK1: 0000 | TIM :FFD8      | TC :0     TXM :0     XF :1
   AR2:    0000    |  SK2: 0000 | PRD :FFFF      | OUTP:0005
   AR3:    0000    |  SK3: 0000 | GREG:0000      | RPTC:   0      CLK:     75
   AR4:    0000    |  SK4: 0000 |---------------------------------------------------
   AR5:    0000    |  SK5: 0000 |       TREG:       0005
   AR6:    0000    |  SK6: 0000 |       PREG:   0000000A
   AR7:    0000    |  SK7: 0000 |       ACC :   0000003C
------------------------------------------------------------------------------------
<CR>:continue <->:return to main <D>: display mode
- <CR>
```

Figure 4-10. Single-Stepping Through the TMS32020/C25 Program (Concluded)

It is now necessary to inspect the memory locations from which the MACD instruction fetches its data. The input values stored in locations >300 to >305 are expected to be 5,6,6,6 respectively since the old data values were never cleared out of RAM. The RAMH command is entered, and when these locations are displayed, a 5 is in memory locations >300 through >305. By carefully examining the program, it becomes clear that the value 5 is writing over the next higher address before it gets multiplied in the next filter tap.

```
      COMMAND:  RAMH <CR>

      Enter start DATA address (in Hex):  300 <CR>

300   0005   0005   0005   0005   0005   0005   0000   0000
308   0000   0000   0000   0000   0000   0000   0000   0000
310   0000   0000   0000   0000   0000   0000   0000   0000
318   0000   0000   0000   0000   0000   0000   0000   0000
320   0000   0000   0000   0000   0000   0000   0000   0000
328   0000   0000   0000   0000   0000   0000   0000   0000
330   0000   0000   0000   0000   0000   0000   0000   0000
338   0000   0000   0000   0000   0000   0000   0000   0000
```

Figure 4-11. TMS32025 Memory Locations in Block B1

3) Correct the error.

Since the error is now located, the code can be modified by changing the *+ to *- in the MACD instruction. The multiplications must start with the oldest samples in the filter since the next higher address is written over before the next multiplication. The LRLK AR1,>300 must also be changed to LRLK AR1,>304 to begin at the last address for the filter taps.

```
       AORG  >32        ;LOAD PROGRAM STARTING AT LOCATION 32.
YN     EQU   5          ;OUTPUT ADDRESS.
XN     EQU   0          ;FIRST INPUT ADDRESSES.
XNM4   EQU   4          ;LAST INPUT ADDRESSES.
       LDPK  4          ;POINT TO DATA PAGE 4.
*
HX     DATA  2,3,4,3,2
*
       LARP  AR0        ;USE AR0 FOR INDIRECT ADDRESSING.
       LRLK  AR0,>200   ;POINT TO BLOCK B0.
       RPTK  XNM4       ;N-1 NUMBER OF COEFFICIENTS.
       BLKP  HX,*+
*
       CNFP             ;USE BLOCK B0 AS PROGRAM AREA.
       LDPK  6          ;POINT TO DATA PAGE 6.
NXTIN  IN    XN,PA0     ;GET NEW SAMPLE.
       LRLK  AR1,>304   ;POINT TO THE STARTING ADDRESS.
       LARP  AR1        ;OF DATA SAMPLES IN BLOCK B1.
*
       MPYK  0          ;SET P REG TO 0.
       ZAC              ;CLEAR ACCUMULATOR.
       RPTK  XNM4       ;REPEAT N-1 TIMES.
       MACD  >FF00,*-   ;MULTIPLY/ACCUMULATE.
       APAC             ;ADD RESULT OF LAST MULTIPLY TO
*                       ;ACCUMULATOR.
       SACL  YN         ;STORE RESULT IN YN.
       OUT   YN,PA1     ;OUTPUT THE RESPONSE TO PA1.
       B     NXTIN      ;GET THE NEXT FILTER COEFFICIENT.
       END
```

Figure 4-12. Revised TMS32020/C25 Program

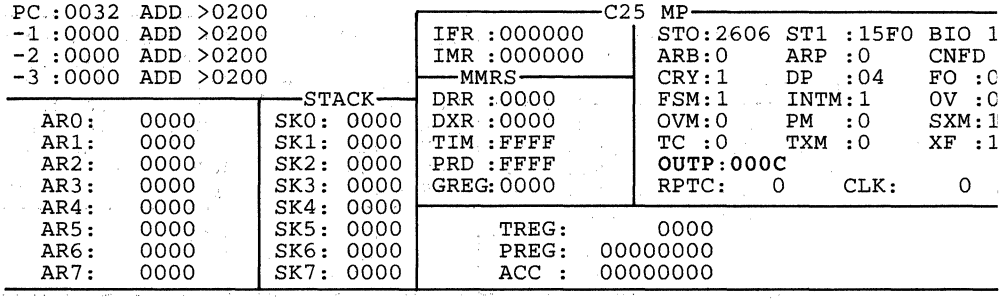
4) Test the revised program for accuracy.

Now that the code has been revised, the program is tested again using the same procedure outlined for Step 1. Simulator execution begins and the assembled revised code is loaded into the simulator. The RUN command is entered for the instructions to be executed. Upon entering a 6 when prompted for an input value, the output value is 12 which is correct. Continuing, we enter the values 5 and 4 giving the outputs 28 and 47, respectively. The code has been debugged and is now accurate.

```
                        (C)   COPYRIGHT  TEXAS INSTRUMENTS
                                INCORPORATED 1986
                               TMS320C25 SIMULATOR
                               RELEASE 1.0  86.194

0 : First 4k words INTERNAL program ROM
1 : First 4k words EXTERNAL program ROM

Enter the memory configuration (0 or 1):  <CR>

First 4k WORDS are mapped on Internal ROM

COMMAND:  L <CR>

Enter a new object file  :  BUGC25.MPO <CR>
**** LOADING PROGRAM "NO$IDT   " ****

COMMAND:  PC <CR>

Present value for program counter  :   0
Enter a new value for the PC (IN HEX) :  32 <CR>

COMMAND:  RUN <CR>

     PC :0032  ADD >0200          ------------C25 MP--------------------------
     -1 :0000  ADD >0200          | IFR :000000   | STO:0604  ST1 :07F0  BIO 1
     -2 :0000  ADD >0200          | IMR :000000   | ARB:0     ARP :0     CNFD
     -3 :0000  ADD >0200          |---MMRS--------| CRY:1     DP  :04    FO :0
     -------------------------STACK---| DRR :0000     | FSM:1     INTM:1     OV :0
       AR0:    0000  | SK0: 0000 | DXR :0000     | OVM:0     PM  :0     SXM:1
       AR1:    0000  | SK1: 0000 | TIM :FFFF     | TC :0     TXM :0     XF :1
       AR2:    0000  | SK2: 0000 | PRD :FFFF     | OUTP:0000
       AR3:    0000  | SK3: 0000 | GREG:0000     | RPTC:    0     CLK:     0
       AR4:    0000  | SK4: 0000 |-------------------------------------------
       AR5:    0000  | SK5: 0000 |     TREG:        0000
       AR6:    0000  | SK6: 0000 |     PREG:    00000000
       AR7:    0000  | SK7: 0000 |     ACC :    00000000
     ------------------------------------------------------------------------

Enter input value (in HEX) or "-" to return to main: 6 <CR>
```

Figure 4-13. Testing the Revised TMS320C0/C25 Program

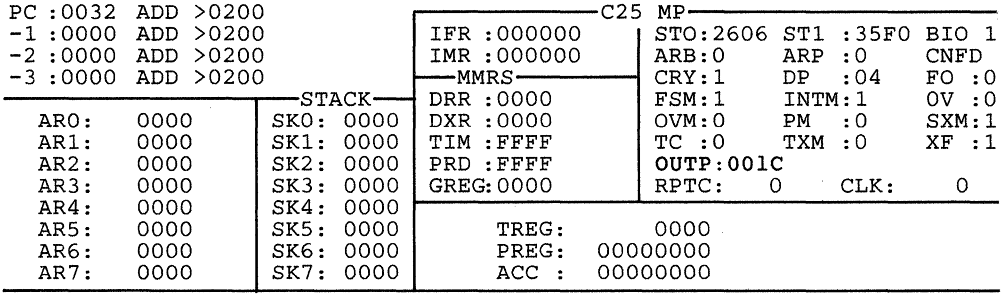
After entering the value of 6, the output (OUTP) changes to 12.



```
 PC :0032  ADD >0200                  ---------------C25 MP----------------------
 -1 :0000  ADD >0200                   IFR :000000   STO:2606  ST1 :15F0  BIO 1
 -2 :0000  ADD >0200                   IMR :000000   ARB:0     ARP :0     CNFD
 -3 :0000  ADD >0200                  ----MMRS-------CRY:1     DP  :04    FO :C
-------------------------------STACK---DRR :0000     FSM:1     INTM:1     OV :C
    AR0:    0000     SK0: 0000   DXR :0000     OVM:0     PM  :0     SXM:1
    AR1:    0000     SK1: 0000   TIM :FFFF     TC :0     TXM :0     XF :1
    AR2:    0000     SK2: 0000   PRD :FFFF     OUTP:000C
    AR3:    0000     SK3: 0000   GREG:0000     RPTC:   0    CLK:    0
    AR4:    0000     SK4: 0000  ---------------------------------------------
    AR5:    0000     SK5: 0000        TREG:       0000
    AR6:    0000     SK6: 0000        PREG:   00000000
    AR7:    0000     SK7: 0000        ACC :   00000000
```

```
Enter input value (in HEX) or "-" to return to main: 5 <CR>
```

After entering the value of 5, the output (OUTP) changes to 28.

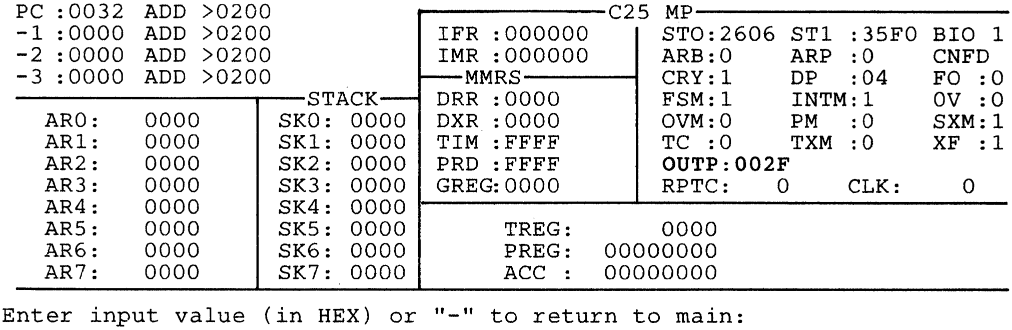


```
 PC :0032  ADD >0200                  ---------------C25 MP----------------------
 -1 :0000  ADD >0200                   IFR :000000   STO:2606  ST1 :35F0  BIO 1
 -2 :0000  ADD >0200                   IMR :000000   ARB:0     ARP :0     CNFD
 -3 :0000  ADD >0200                  ----MMRS-------CRY:1     DP  :04    FO :0
-------------------------------STACK---DRR :0000     FSM:1     INTM:1     OV :0
    AR0:    0000     SK0: 0000   DXR :0000     OVM:0     PM  :0     SXM:1
    AR1:    0000     SK1: 0000   TIM :FFFF     TC :0     TXM :0     XF :1
    AR2:    0000     SK2: 0000   PRD :FFFF     OUTP:001C
    AR3:    0000     SK3: 0000   GREG:0000     RPTC:   0    CLK:    0
    AR4:    0000     SK4: 0000  ---------------------------------------------
    AR5:    0000     SK5: 0000        TREG:       0000
    AR6:    0000     SK6: 0000        PREG:   00000000
    AR7:    0000     SK7: 0000        ACC :   00000000
```

```
Enter input value (in HEX) or "-" to return to main: 4 <CR>
```

Figure 4-13. Testing the Revised TMS320C0/C25 Program (Continued)

After entering the value of 4, the output (OUTP) changes to 47.



```
PC :0032  ADD >0200            ----------------------C25  MP--------------------------
-1 :0000  ADD >0200            IFR :000000     | STO:2606  ST1  :35F0  BIO 1
-2 :0000  ADD >0200            IMR :000000     | ARB:0     ARP  :0     CNFD
-3 :0000  ADD >0200            ---MMRS---------| CRY:1     DP   :04    FO :0
--------------------------STACK--- DRR :0000   | FSM:1     INTM:1      OV :0
   ARO:    0000   | SKO:  0000 | DXR :0000     | OVM:0     PM   :0     SXM:1
   AR1:    0000   | SK1:  0000 | TIM :FFFF     | TC :0     TXM  :0     XF :1
   AR2:    0000   | SK2:  0000 | PRD :FFFF     | OUTP:002F
   AR3:    0000   | SK3:  0000 | GREG:0000     | RPTC:    0      CLK:      0
   AR4:    0000   | SK4:  0000 |-------------------------------------------------
   AR5:    0000   | SK5:  0000 |      TREG:       0000
   AR6:    0000   | SK6:  0000 |      PREG:   00000000
   AR7:    0000   | SK7:  0000 |      ACC :   00000000
-----------------------------------------------------------------------------------
Enter input value (in HEX) or "-" to return to main:
```

Figure 4-13. Testing the Revised TMS320C0/C25 Program (Concluded)

Appendix A

Simulator Stop Codes

Table A-1 lists the TMS320 Simulator run-time stop codes that may occur during program execution. One of these stop codes is displayed each time program execution is suspended. Those stop codes that appear only when a particular device is used are indicated by the device number enclosed in parentheses following the stop-code definition.

The following stop codes (2600, 2780, 3505, 4190, 8683, and 9105) are illegal trap codes, and indicate the existence of states that do not occur in a properly functioning simulator.

Table A-1. Simulator Stop Codes

Stop Code	Definition
2600	Illegal trap
2695	Break on data read
2780	Illegal trap
2795	Break on output write
3505	Illegal trap
3665	Break on table read
4055	Break on table write
4065	Break on table write
4190	Illegal trap
5000	<CNTRL-C>
6000	Negative operand not allowed for SUBC instruction (TMS32020, TMS320C25).
6001	Data memory address must be in the range of 65280 to 65535 inclusive (TMS32020, TMS320C25).
6002	Data memory address must be on-chip (TMS32020, TMS320C25).
6003	CNF must equal 1 for MAC and MACD instructions (TMS32020, TMS320C25).
7601	Illegal opcode
8405	Break on instruction acquisition
8662	Illegal indirect addressing structure (bits 1,2, and 6 not zero)
8670	Illegal indirect addressing structure (bits 4 and 5 are both on).
8680	Break on data memory read (during development of indirect addressing)
8683	Illegal trap
9011	Branch to self
9020	Break on instruction acquisition
9105	Illegal trap
9950	Accumulator was used the first cycle after SUBC (TMS32010).
10000	"Steps" expired
10100	Addressed beyond end of 65536-word program ROM
10144	Addressed beyond end of 144-word data RAM (TMS32010)
10400	Error breakpoint (over/underflow, etc.)
10496	Addressed beyond end of 4096-word program ROM (TMS32010)
11000+N	Instruction acquisition breakpoint #N
12000+N	Program ROM breakpoint #N
13000+N	Data RAM breakpoint #N

Index

TI Sales Offices

ALABAMA: Huntsville (205) 837-7530.

ARIZONA: Phoenix (602) 995-1007;
Tucson (602) 624-3276.

CALIFORNIA: Irvine (714) 660-1200;
Sacramento (916) 929-0197;
San Diego (619) 278-9600;
Santa Clara (408) 980-9000;
Torrance (213) 217-7000;
Woodland Hills (818) 704-7759.

COLORADO: Aurora (303) 368-8000.

CONNECTICUT: Wallingford (203) 269-0074.

FLORIDA: Altamonte Springs (305) 260-2116;
Ft. Lauderdale (305) 973-8502;
Tampa (813) 286-0420.

GEORGIA: Norcross (404) 662-7900.

ILLINOIS: Arlington Heights (312) 640-3000.

INDIANA: Carmel (317) 573-6400;
Ft. Wayne (219) 424-5174.

IOWA: Cedar Rapids (319) 395-9550.

KANSAS: Overland Park (913) 451-4511.

MARYLAND: Baltimore (301) 944-8600.

MASSACHUSETTS: Waltham (617) 895-9100.

MICHIGAN: Farmington Hills (313) 553-1500;
Grand Rapids (616) 957-4200.

MINNESOTA: Eden Prairie (612) 828-9300.

MISSOURI: St. Louis (314) 569-7600.

NEW JERSEY: Iselin (201) 750-1050.

NEW MEXICO: Albuquerque (505) 345-2555.

NEW YORK: East Syracuse (315) 463-9291;
Melville (516) 454-6600; **Pittsford** (716) 385-6770;
Poughkeepsie (914) 473-2900.

NORTH CAROLINA: Charlotte (704) 527-0930;
Raleigh (919) 876-2725.

OHIO: Beachwood (216) 464-6100;
Dayton (513) 258-3877.

OREGON: Beaverton (503) 643-6758.

PENNSYLVANIA: Blue Bell (215) 825-9500.

PUERTO RICO: Hato Rey (809) 753-8700.

TENNESSEE: Johnson City (615) 461-2192.

TEXAS: Austin (512) 250-6769;
Houston (713) 778-6592; **Richardson** (214) 680-5082;
San Antonio (512) 496-1779.

UTAH: Murray (801) 266-8972.

VIRGINIA: Fairfax (703) 849-1400.

WASHINGTON: Redmond (206) 881-3080.

WISCONSIN: Brookfield (414) 782-2899.

CANADA: Nepean, Ontario (613) 726-1970;
Richmond Hill, Ontario (416) 884-9181;
St. Laurent, Quebec (514) 336-1860.

TI Regional Technology Centers

CALIFORNIA: Irvine (714) 660-8140;
Santa Clara (408) 748-2220;
Torrance (213) 217-7019.

COLORADO: Aurora (303) 368-8000.

GEORGIA: Norcross (404) 662-7945.

ILLINOIS Arlington Heights (313) 640-2909.

MASSACHUSETTS: Waltham (617) 895-9196.

TEXAS: Richardson (214) 680-5066.

CANADA: Nepean, Ontario (613) 726-1970.

TI Distributors

TI AUTHORIZED DISTRIBUTORS
Arrow/Kierulff Electronics Group
Arrow Canada (Canada)
Future Electronics (Canada)
GRS Electronics Co., Inc.
Hall-Mark Electronics
Marshall Industries
Newark Electronics
Schweber Electronics
Time Electronics
Wyle Laboratories
Zeus Components

– OBSOLETE PRODUCT ONLY –
Rochester Electronics, Inc.
Newburyport, Massachusetts
(617) 462-9332

ALABAMA: Arrow/Kierulff (205) 837-6955;
Hall-Mark (205) 837-8700; Marshall (205) 881-9235;
Schweber (205) 895-0480.

ARIZONA: Arrow/Kierulff (602) 437-0750;
Hall-Mark (602) 437-1200; Marshall (602) 496-0290;
Schweber (602) 997-4874; Wyle (602) 866-2888.

CALIFORNIA: Los Angeles/Orange County:
Arrow/Kierulff (818) 701-7500, (714) 838-5422;
Hall-Mark (818) 716-7300, (714) 669-4100,
(213) 217-8400; Marshall (818) 407-0101, (818) 459-5500,
(714) 458-5395; Schweber (818) 999-4702;
(714) 863-0200, (213) 320-8090; Wyle (213) 322-9953,
(818) 880-9000, (714) 863-9953; Zeus (714) 921-9000;
Sacramento: Hall-Mark (916) 722-8600;
Marshall (916) 635-9700; Schweber (916) 929-9732;
Wyle (916) 638-5282;
San Diego: Arrow/Kierulff (619) 565-4800;
Hall-Mark (619) 268-1201; Marshall (619) 578-9600;
Schweber (619) 450-0454; Wyle (619) 565-9171;
San Francisco Bay Area: Arrow/Kierulff (408) 745-6600,
Hall-Mark (408) 432-0900; Marshall (408) 942-4600;
Schweber (408) 432-7171; Wyle (408) 727-2500;
Zeus (408) 998-5121.

COLORADO: Arrow/Kierulff (303) 790-4444;
Hall-Mark (303) 790-1662; Marshall (303) 451-8383;
Schweber (303) 799-0258; Wyle (303) 457-9953.

CONNETICUT: Arrow/Kierulff (203) 265-7741;
Hall-Mark (203) 269-0100; Marshall (203) 265-3822;
Schweber (203) 748-7080.

FLORIDA: Ft. Lauderdale:
Arrow/Kierulff (305) 429-8200; Hall-Mark (305) 971-9280;
Marshall (305) 977-4880; Schweber (305) 977-7511;
Orlando: Arrow/Kierulff (305) 725-1480, (305) 682-6923;
Hall-Mark (305) 855-4020; Marshall (305) 767-8585;
Schweber (305) 331-7555; Zeus (305) 365-3000;
Tampa: Hall-Mark (813) 530-4543;
Marshall (813) 576-1399.

GEORGIA: Arrow/Kierulff (404) 449-8252;
Hall-Mark (404) 447-8000; Marshall (404) 923-5750;
Schweber (404) 449-9170.

ILLINOIS: Arrow/Kierulff (312) 250-0500;
Hall-Mark (312) 860-3800; Marshall (312) 490-0155;
Newark (312) 784-5100; Schweber (312) 364-3750.

INDIANA: Indianapolis: Arrow/Kierulff (317) 243-9353;
Hall-Mark (317) 872-8875; Marshall (317) 297-0483.

IOWA: Arrow/Kierulff (319) 395-7230;
Schweber (319) 373-1417.

KANSAS: Kansas City: Arrow/Kierulff (913) 541-9542;
Hall-Mark (913) 888-4747; Marshall (913) 492-3121;
Schweber (913) 492-2922.

MARYLAND: Arrow/Kierulff (301) 995-6002;
Hall-Mark (301) 988-9800; Marshall (301) 840-9450;
Schweber (301) 840-5900; Zeus (301) 997-1118.

MASSACHUSETTS Arrow/Kierulff (617) 935-5134;
Hall-Mark (617) 667-0902; Marshall (617) 658-0810;
Schweber (617) 275-5100, (617) 657-0760;
Time (617) 532-6200; Zeus (617) 863-8800.

MICHIGAN: Detroit: Arrow/Kierulff (313) 971-8220;
Marshall (313) 525-5850; Newark (313) 967-0600;
Schweber (313) 525-8100;
Grand Rapids: Arrow/Kierulff (616) 243-0912.

MINNESOTA: Arrow/Kierulff (612) 830-1800;
Hall-Mark (612) 941-2600; Marshall (612) 559-2211;
Schweber (612) 941-5280.

MISSOURI: St. Louis: Arrow/Kierulff (314) 567-6888;
Hall-Mark (314) 291-5350; Marshall (314) 291-4650;
Schweber (314) 739-0526.

NEW HAMPSHIRE: Arrow/Kierulff (603) 668-6968;
Schweber (603) 625-2250.

NEW JERSEY: Arrow/Kierulff (201) 538-0900,
(609) 596-8000; GRS Electronics (609) 964-8560;
Hall-Mark (201) 575-4415, (609) 235-1900;
Marshall (201) 882-0320, (609) 234-9100;
Schweber (201) 227-7880.

NEW MEXICO: Arrow/Kierulff (505) 243-4566.

NEW YORK: Long Island:
Arrow/Kierulff (516) 231-1000; Hall-Mark (516) 737-0600;
Marshall (516) 273-2424; Schweber (516) 334-7555;
Zeus (914) 937-7400;
Rochester: Arrow/Kierulff (716) 427-0300;
Hall-Mark (716) 244-9290; Marshall (716) 235-7620;
Schweber (716) 424-2222;
Syracuse: Marshall (607) 798-1611.

NORTH CAROLINA: Arrow/Kierulff (919) 876-3132,
(919) 725-8711; Hall-Mark (919) 872-0712;
Marshall (919) 878-9882; Schweber (919) 876-0000.

OHIO: Cleveland: Arrow/Kierulff (216) 248-3990;
Hall-Mark (216) 349-4632; Marshall (216) 248-1788;
Schweber (216) 464-2970;
Columbus: Arrow/Kierulff (614) 436-0928;
Hall-Mark (614) 888-3313;
Dayton: Arrow/Kierulff (513) 435-5563;
Marshall (513) 898-4480; Schweber (513) 439-1800.

OKLAHOMA: Arrow/Kierulff (918) 252-7537;
Schweber (918) 622-8003.

OREGON: Arrow/Kierulff (503) 645-6456;
Marshall (503) 644-5050; Wyle (503) 640-6000.

PENNSYLVANIA: Arrow/Kierulff (412) 856-7000,
(215) 928-1800; GRS Electronics (215) 922-7037;
Schweber (215) 441-0600, (412) 963-6804.

TEXAS: Austin: Arrow/Kierulff (512) 835-4180;
Hall-Mark (512) 258-8848; Marshall (512) 837-1991;
Schweber (512) 339-0088; Wyle (512) 834-9957;
Dallas: Arrow/Kierulff (214) 380-6464;
Hall-Mark (214) 553-4300; Marshall (214) 233-5200;
Schweber (214) 661-5010; Wyle (214) 235-9953;
Zeus (214) 783-7010;
Houston: Arrow/Kierulff (713) 530-4700;
Hall-Mark (713) 781-6100; Marshall (713) 895-9200;
Schweber (713) 784-3600; Wyle (713) 879-9953.

UTAH: Arrow/Kierulff (801) 973-6913;
Hall-Mark (801) 972-1008; Marshall (801) 485-1551;
Wyle (801) 974-9953.

WASHINGTON: Arrow/Kierulff (206) 575-4420;
Marshall (206) 747-9100; Wyle (206) 453-8300.

WISCONSIN: Arrow/Kierulff (414) 792-0150;
Hall-Mark (414) 797-7844; Marshall (414) 797-8400;
Schweber (414) 784-9020.

CANADA: Calgary: Future (403) 235-5325;
Edmonton: Future (403) 438-2858;
Montreal: Arrow Canada (514) 735-5511;
Future (514) 694-7710;
Ottawa: Arrow Canada (613) 226-6903;
Future (613) 820-8313;
Quebec City: Arrow Canada (418) 687-4231;
Toronto: Arrow Canada (416) 672-7769;
Future (416) 638-4771;
Vancouver: Future (604) 294-1166;
Winnipeg: Future (204) 339-0554.

Customer Response Center

TOLL FREE: (800) 232-3200

OUTSIDE USA: (214) 995-6611
(8:00 a.m. – 5:00 p.m. CST)

TEXAS INSTRUMENTS

BU

TI Worldwide Sales Offices

ALABAMA: Huntsville: 500 Wynn Drive, Suite 514, Huntsville, AL 35805, (205) 837-7530.

ARIZONA: Phoenix: 8825 N. 23rd Ave., Phoenix, AZ 85021, (602) 995-1007.

CALIFORNIA: Irvine: 17891 Cartwright Rd., Irvine, CA 92714, (714) 660-8187; **Sacramento:** 1900 Point West Way, Suite 171, Sacramento, CA 95815, (916) 929-1521; **San Diego:** 4333 View Ridge Ave., Suite B., San Diego, CA 92123, (619) 278-9601; **Santa Clara:** 5353 Betsy Ross Dr., Santa Clara, CA 95054, (408) 980-9000; **Torrance:** 690 Knox St., Torrance, CA 90502, (213) 217-7010; **Woodland Hills:** 21220 Erwin St., Woodland Hills, CA 91367, (818) 704-7759.

COLORADO: Aurora: 1400 S. Potomac Ave., Suite 101, Aurora, CO 80012, (303) 368-8000.

CONNECTICUT: Wallingford: 9 Barnes Industrial Park Rd., Barnes Industrial Park, Wallingford, CT 06492, (203) 269-0074.

FLORIDA: Ft. Lauderdale: 2765 N.W. 62nd St., Ft. Lauderdale, FL 33309, (305) 973-8502; **Maitland:** 2601 Maitland Center Parkway, Maitland, FL 32751, (305) 660-4600; **Tampa:** 5010 W. Kennedy Blvd., Suite 101, Tampa, FL 33609, (813) 870-6420.

GEORGIA: Norcross: 5515 Spalding Drive, Norcross, GA 30092, (404) 662-7900

ILLINOIS: Arlington Heights: 515 W. Algonquin, Arlington Heights, IL 60005, (312) 640-2925.

INDIANA: Ft. Wayne: 2020 Inwood Dr., Ft. Wayne, IN 46815, (219) 424-5174; **Indianapolis:** 2346 S. Lynhurst, Suite J-400, Indianapolis, IN 46241, (317) 248-8555.

IOWA: Cedar Rapids: 373 Collins Rd. NE, Suite 200, Cedar Rapids, IA 52402, (319) 395-9550.

MARYLAND: Baltimore: 1 Rutherford Pl., 7133 Rutherford Rd., Baltimore, MD 21207, (301) 944-8600.

MASSACHUSETTS: Waltham: 504 Totten Pond Rd., Waltham, MA 02154, (617) 895-9100.

MICHIGAN: Farmington Hills: 33737 W. 12 Mile Rd., Farmington Hills, MI 48018, (313) 553-1500.

MINNESOTA: Eden Prairie: 11000 W. 78th St., Eden Prairie, MN 55344 (612) 828-9300.

MISSOURI: Kansas City: 8080 Ward Pkwy., Kansas City, MO 64114, (816) 523-2500; **St. Louis:** 11816 Borman Drive, St. Louis, MO 63146, (314) 569-7600.

NEW JERSEY: Iselin: 485E U.S. Route 1 South, Parkway Towers, Iselin, NJ 08830 (201) 750-1050

NEW MEXICO: Albuquerque: 2820-D Broadbent Pkwy NE, Albuquerque, NM 87107, (505) 345-2555.

NEW YORK: East Syracuse: 6365 Collamer Dr., East Syracuse, NY 13057, (315) 463-9291; **Endicott:** 112 Nanticoke Ave., P.O. Box 618, Endicott, NY 13760, (607) 754-3900; **Melville:** 1 Huntington Quadrangle, Suite 3C10, P.O. Box 2936, Melville, NY 11747, (516) 454-6600; **Pittsford:** 2851 Clover St., Pittsford, NY 14534, (716) 385-6770; **Poughkeepsie:** 385 South Rd., Poughkeepsie, NY 12601, (914) 473-2900.

NORTH CAROLINA: Charlotte: 8 Woodlawn Green, Woodlawn Rd., Charlotte, NC 28210, (704) 527-0930; **Raleigh:** 2809 Highwoods Blvd., Suite 100, Raleigh, NC 27625, (919) 876-2725.

OHIO: Beachwood: 23408 Commerce Park Rd., Beachwood, OH 44122, (216) 464-6100; **Dayton:** Kingsley Bldg., 4124 Linden Ave., Dayton, OH 45432, (513) 258-3877.

OREGON: Beaverton: 6700 SW 105th St., Suite 110, Beaverton, OR 97005, (503) 643-6758.

PENNSYLVANIA: Ft. Washington: 260 New York Dr., Ft. Washington, PA 19034, (215) 643-6450; **Coraopolis:** 420 Rouser Rd., 3 Airport Office Park, Coraopolis, PA 15108, (412) 771-8550.

PUERTO RICO: Hato Rey: Mercantil Plaza Bldg., Suite 505, Hato Rey, PR 00919, (809) 753-8700.

TEXAS: Austin: P.O. Box 2909, Austin, TX 78769, (512) 250-7655; **Richardson:** 1001 E. Campbell Rd., Richardson, TX 75080, (214) 680-5082; **Houston:** 9100 Southwest Frwy., Suite 237, Houston, TX 77036, (713) 778-6592; **San Antonio:** 1000 Central Parkway South, San Antonio, TX 78232, (512) 496-1779.

UTAH: Murray: 5201 South Green SE, Suite 200, Murray, UT 84107, (801) 266-8972.

VIRGINIA: Fairfax: 2750 Prosperity, Fairfax, VA 22031, (703) 849-1400.

WASHINGTON: Redmond: 5010 148th NE, Bldg B, Suite 107, Redmond, WA 98052, (206) 881-3080.

WISCONSIN: Brookfield: 450 N. Sunny Slope, Suite 150, Brookfield, WI 53005, (414) 785-7140.

CANADA: Nepean: 301 Moodie Drive, Mallorn Center, Nepean, Ontario, Canada, K2H9C4, (613) 726-1970. **Richmond Hill:** 280 Centre St. E., Richmond Hill L4C1B1, Ontario, Canada (416) 884-9181; **St. Laurent:** Ville St. Laurent Quebec, 9460 Trans Canada Hwy., St. Laurent, Quebec, Canada H4S1R7, (514) 335-8392.

ARGENTINA: Texas Instruments Argentina S.A.I.C.F.: Esmeralda 130, 15th Floor, 1035 Buenos Aires, Argentina, 1+394-3008.

AUSTRALIA (& NEW ZEALAND): Texas Instruments Australia Ltd.: 6-10 Talavera Rd., North Ryde (Sydney), New South Wales, Australia 2113, 2 + 887-1122; 5th Floor, 418 St. Kilda Road, Melbourne, Victoria, Australia 3004, 3 + 267-4677; 171 Philip Highway, Elizabeth, South Australia 5112, 8 + 255-2066.

AUSTRIA: Texas Instruments Ges.m.b.H.: Industriestrabe B/16, A-2345 Brunn/Gebirge, 2236-846210.

BELGIUM: Texas Instruments N.V. Belgium S.A.: Mercure Centre, Raketstraat 100, Rue de la Fusee, 1130 Brussels, Belgium, 2/720.80.00.

BRAZIL: Texas Instruments Electronicos do Brasil Ltda.: Rua Paes Leme, 524-7 Andar Pinheiros, 05424 Sao Paulo, Brazil, 0815-6166.

DENMARK: Texas Instruments A/S, Mairelundvej 46E, DK-2730 Herlev, Denmark, 2 - 91 74 00.

FINLAND: Texas Instruments Finland OY: Teollisuuskatu 19D 00511 Helsinki 51, Finland, (90) 701-3133.

FRANCE: Texas Instruments France: Headquarters and Prod. Plant, BP 05, 06270 Villeneuve-Loubet, (93) 20-01-01; Paris Office, BP 67 8-10 Avenue Morane-Saulnier, 78141 Velizy-Villacoublay, (3) 946-97-12; Lyon Sales Office, L'Oree D'Ecully, Batiment B, Chemin de la Forestiere, 69130 Ecully, (7) 833-04-40; Strasbourg Sales Office, Le Sebastopol 3, Quai Kleber, 67055 Strasbourg Cedex, (88) 22-12-66; Rennes, 23-25 Rue du Puits Mauger, 35100 Rennes, (99) 31-54-86; Toulouse Sales Office, Le Peripole—2, Chemin du Pigeonnier de la Cepiere, 31100 Toulouse, (61) 44-18-19; Marseille Sales Office, Noilly Paradis—146 Rue Paradis, 13006 Marseille, (91) 37-25-30.

TEXAS INSTRUMENTS

GERMANY (Fed. Republic of Germany): Texas Instruments Deutschland GmbH: Haggertystrasse 1, D-8050 Freising, 8161+80-4591; Kurfuerstendamm 195/196, D-1000 Berlin 15, 30+882-7365; III, Hagen 43/Kibbelstrasse, .19, D-4300 Essen, 201-24250; Frankfurter Allee 6-8, D-6236 Eschborm 1, 06196+8070; Hamburgerstrasse 11, D-2000 Hamburg 76, 040+220-1154, Kirchhorsterstrasse 2, D-3000 Hannover 51, 511+648021; Maybachstrabe 11, D-7302 Ostfildern 2-Nelingen, 711+547001; Mixikoring 19, D-2000 Hamburg 60, 40+637+0061; Postfach 1309, Roonstrasse 16, D-5400 Koblenz, 261+35044.

HONG KONG (+ PEOPLES REPUBLIC OF CHINA): Texas Instruments Asia Ltd., 8th Floor, World Shipping Ctr., Harbour City, 7 Canton Rd., Kowloon, Hong Kong, 3 + 722-1223.

IRELAND: Texas Instruments (Ireland) Limited: Brewery Rd., Stillorgan, County Dublin, Eire, 1 831311.

ITALY: Texas Instruments Semiconduttori Italia Spa: Viale Delle Scienze, 1, 02015 Cittaducale (Rieti), Italy, 746 694.1; Via Salaria KM 24 (Palazzo Cosma), Monterotondo Scalo (Rome), Italy, 6+9003241; Viale Europa, 38-44, 20093 Cologno Monzese (Milano), 2 2532541; Corso Svizzera, 185, 10100 Torino, Italy, 11 774545; Via J. Barozzi 6, 40100 Bologna, Italy, 51 355851.

JAPAN: Texas Instruments Asia Ltd.: 4F Aoyama Fuji Bldg., 6-12, Kita Aoyama 3-Chome, Minato-ku, Tokyo, Japan 107, 3-498-2111; Osaka Branch, 5F, Nissho Iwai Bldg., 30 Imabashi 3- Chome, Higashi-ku, Osaka, Japan 541, 06-204-1881; Nagoya Branch, 7F Daini Toyota West Bldg., 10-27, Meieki 4-Chome, Nakamura-ku Nagoya, Japan 450, 52-583-8691.

KOREA: Texas Instruments Supply Co.: 3rd Floor, Samon Bldg., Yuksam-Dong, Gangnam-ku, 135 Seoul, Korea, 2+462-8001.

MEXICO: Texas Instruments de Mexico S.A.: Mexico City, AV Reforma No. 450 — 10th Floor, Mexico, D.F., 06600, 5+514-3003.

MIDDLE EAST: Texas Instruments: No. 13, 1st Floor Mannai Bldg., Diplomatic Area, P.O. Box 26335, Manama Bahrain, Arabian Gulf, 973+274681.

NETHERLANDS: Texas Instruments Holland B.V., P.O. Box 12995, (Bullewijk) 1100 CB Amsterdam, Zuid-Oost, Holland 20+5602911.

NORWAY: Texas Instruments Norway A/S: PB106, Refstad 131, Oslo 1, Norway, (2) 155090.

PHILIPPINES: Texas Instruments Asia Ltd.: 14th Floor, Ba- Lepanto Bldg., 8747 Paseo de Roxas, Makati, Metro Manila, Philippines, 2+8188987.

PORTUGAL: Texas Instruments Equipamento Electronico (Portugal), Lda.: Rua Eng. Frederico Ulrich, 2650 Moreira Da Maia, 4470 Maia, Portugal, 2-948-1003.

SINGAPORE (+ INDIA, INDONESIA, MALAYSIA, THAILAND): Texas Instruments Asia Ltd.: 12 Lorong Bakar Batu, Unit 01-02, Kolam Ayer Industrial Estate, Republic of Singapore, 747-2255.

SPAIN: Texas Instruments Espana, S.A.: C/Jose Lazaro Galdiano No. 6, Madrid 16, 1/458.14.58.

SWEDEN: Texas Instruments International Trade Corporation (Sverigefilialen): Box 39103, 10054 Stockholm, Sweden, 8 - 235480.

SWITZERLAND: Texas Instruments, Inc., Reidstrasse 6, CH-8953 Dietikon (Zuerich) Switzerland, 1-740 2220.

TAIWAN: Texas Instruments Supply Co.: Room 903, 205 Tun Hwan Rd., 71 Sung-Kiang Road, Taipei, Taiwan, Republic of China, 2 + 521-9321.

UNITED KINGDOM: Texas Instruments Limited: Manton Lane, Bedford, MK41 7PA, England, 0234 67466; St. James House, Wellington Road North, Stockport, SK4 2RT, England, 61+442-7162.

BM

TEXAS INSTRUMENTS

Printed in U.S.A., November 1988
1604904-9704