# MODEL 250

# DATA ACQUISITION AND

# SIGNAL PROCESSING BOARD

# FOR THE IBM PC AT AND ISA BUS COMPATIBLES

Dalanco Spry
89 Winslow Avenue
Rochester, NY 14620
U.S.A.

(716) 473-3610

## TABLE OF CONTENTS

## Table of Contents

## 1. <u>INTRODUCTION</u>

This manual is the reference to the Model 250 Digital Signal Processor Board and its accompanying software. The reference for the Texas Instruments TMS320C25 Digital Signal Processor and its assembly language is the <u>Second Generation TMS320 User's Guide</u> available from TI.

<u>Partial List of References</u>

1. Second Generation TMS320 User's Guide , Reference no. SPRUO14A
2. TMS320 Development Support Reference Guide, Ref no. SPRU007A
3. Digital Signal Processing Applications with the TMS320 Family, Ref no. SPRA012

## 2. <u>INSTALLATION</u>

A. <u>Make a backup working copy of the software.</u>
Please see your DOS manual if you have any questions on the use of the COPY command.

B. <u>Set the address jumpers for the IO mapping.</u>

These jumpers determine the Base IO addressing at which the Model 250 resides in the host CPU's IO address space. (The host CPU is the IBM PC AT, etc.). This address should not conflict with existing functions or adapters in the PC system. (The h appended to a number indicates that the number is written in hexadecimal, or base 16 notation. The '0x' prefix, used in C programs, and the '>' prefix, used in the Dalanco assembler and debugger, are equivalent hexadecimal notations). The address setting is located at J7 on the lower right side of the board. The 7 jumpers are read right to left (most significant bit to least significant), and determine the 8 byte block of IO addresses used to control the board. Only bits 3 thru 9 of the address are selectable. The upper bits (bits 10-11) are forced to represent a 0 while bits 0-2 are also zero. Thus shorting the jumpers as in Fig. 1 yields 3OOh as the Base IO address. This is the default address to be used on the majority of IBM PC systems. A jumpered connection corresponds to a bit set to <u>zero</u> and no jumper to a bit set to <u>one</u>.

**Table of Contents**

The Base IO address is selectable on 8 byte boundaries.

Note that in systems with multiple Model 250 boards, each Model 250 must be addressed at its own unique Base IO address.

## Examples

```
For the Base IO address of 300h, bits 3-9 are  1100000.
                                              3 | 0
The correct jumper configuration is
```

```
 0  0  0  0  0  1  1
 _  _  _  _  _
|o||o||o||o||o| o  o
|o||o||o||o||o| o  o

 A3                A9
```

**Fig. 1**

Example for Base IO address 310h:

```
 0  1  0  0  0  1  1
 _     _  _  _
|o| o |o||o||o| o  o
|o| o |o||o||o| o  o
```

**Fig. 1a**

## OTHER SETTINGS

### C. Program RAM Size

The Model 250 may be fitted with either 4K or 64K words of Program RAM. The Program RAM resides in the four ICs directly above the TMS320.

Case 1:
4K words of RAM. The RAM ICs are Cypress CY7C169 or equivalent. They must be mounted as shown, occupying the rightmost 20 pins of the 24 pin sockets. Jumper J2 is set to the ’4’ position.

Case 2:
64K words of RAM. The RAM ICs are Hitachi 6708 or equivalent. Jumper J2 is set to the ’64’ position.

### D. EPROM DSP in Stand Alone Mode

Jumper J0 is set to the C (for computer) position when the Model 250 is equipped with a TMS320E25 preprogrammed to operate in standalone mode.
Otherwise set J0 to the default ’P’ (for processor) position, whether using the TMS320C25 or TMS320E25 DSPs.

### E. Interrupt Jumper Settings

Jumper JI1 is used to implement one of the following PC interrupts:

INT10, INT11, INT12, or INT15.

Only one connection should be made.

JI2 is jumpered if either INT3 or INT5 are desired. Jumpers JI1 and JI2 cannot both be jumpered. Only one of the connections may be made.

## **3.** OPERATION EXAMPLE

The best way to become familiar with the Model 250 is to use D325, the monitor program. We will use it to:

1) Display Program memory
2) Write a short program using the Assemble and Substitute commands.
3) Disassemble the program to make sure it is correct.
4) Save the program on disk.
5) Fill memory with a constant.
6) Read the program from disk.
7) Run the program.
8) Halt the program.
9) Display the result.

Note: In the discussion below, **<cr>** indicates a carriage return.

Invoke D325:
Type D325 at the MS-DOS prompt. A dash (-) appears as the new prompt.

Display memory:
To display the first 1OOh words of Program memory (0 to ffh), type

```
-dp0,ff<cr>
```

Note that the addressing points to 16 bit words, not 8 bit bytes.

Write a short program:
To assemble an instruction at location 0, type

```
-a0<cr>
```

The address 0 appears and D325 awaits your entry. Type

**b 7** (branch to address 7).

Note: If D325 responds with ???, you made an entry error. Reenter the instruction. After successful assembly D325 provides the next address (in this example, 2) and waits for your entry. If you don’t want to enter anything at that address, press **<cr>** to move to the following address. If you want to get out of Assemble mode enter, type

```
.<cr>
```

The dash (-) prompt reappears.

We will continue our program at address 7, so type

```
-a7<cr>
```

Then enter the rest of the program

```
7 DINT
8 LDPK 4
9 LACK 80
A TBLR 0
B LACK 81
C TBLW 0
D B >D
F . <cr>
```

The assembler displays the object code corresponding to each instruction.

Using the Substitute command, let’s enter data at location 50h.

```
-sp50<cr>
0050 ABCD-1234 .<cr>
```

Here we are entering the number 1234h  at location 5Oh .

Disassemble the program:
The following command disassembles the statement at location 0.

```
-u0
```

Continue to press <cr> to disassemble subsequent instructions. Correct any errors using the Assemble or Substitute commands.

Save the program on disk:
We will now write the Model 250 Program memory space (4K words) to disk. Make sure that you have 8 kbytes of free space on that disk.

```
-wb:myprog<cr>
```

In this example the program is written to the b: disk drive.

Fill Program memory with a constant:
Enter the Fill command and D325 responds with three questions. In the example below, we are filling all of memory with zeroes.

```
-fp<cr>
  Enter constant?0<cr>
  Starting Address?0<cr>
  Number of Locations to fill?4096<cr>
```

Read the program from disk:
Since this program is short we read in only the first 60h words.

```
-rb:myprog
  Starting Address? 0
  Ending Address (<=FFF)? 60
```

Run the program:
Enter the Go command to run your program.

```
-g<cr>
```

Halt the program:
Enter the Halt command to stop your program.

```
-h<cr>
```

Display the result:
Now use the Display command to see if the program performed as expected. Did the contents of RAM location 80 (50h) get copied into location 81 (51h) ?

## 4. **HARDWARE FEATURES**

### A. Timer

The timer consists of two 8 bit counters connected in cascade. Each of the two 8 bit counters is associated with its own divisor. The divisor determines the output frequency of the counter as a function of its input frequency. If, for example, the input frequency to a counter is 1 Mhz, and the counter is programmed with a divisor of 4, then the output frequency will be 250 Khz.

The Model 250 Timer diagram:

```
             ___________           __________
            | Counter 1 |         | Counter 2|
4 Mhz ----->| div1      |-------->| div2     |---> OUTPUT CLOCK
            |___________|         |__________|
```

OUTPUT CLOCK FREQUENCY = $\dfrac{4\ \text{Mhz}}{\text{div1} * \text{div2}}$

div1 and div2 may be in the range ( 0<  <255).

Once div1 and div2 are determined, subtract each of them from 255 (0FFh), and form a single 16 bit value.

```
     time1 = 255 - div1
     time2 = 255 - div2
     TheValue = time1time2
```

The TMS320 writes TheValue to Port 0 to set the Timer.

Example:
          We need a clock value of 100 Khz (divisor of 4 Mhz/100 Khz = 40). We can then choose div1 = 40, div2 = 1.
     Then

```
          time1 = 255 - 40 = 215 = 0D7h
          time2 = 255 - 1  = 254 = 0FEh
```

The TMS320 then writes 0D7FEh (or 0FED7h) to Port 0 to set the Timer output to 100 Khz.

We could also have chosen div1 = 20, div2 = 2 or another combination to achieve the same result.

## Programming the Timer

To program the timer, you may use the D325 **O** command, or write a subroutine as part of your TMS320 assembly language program. Assembly language examples may be found on the distribution diskette.

## B. Memory Access

The Model 250 holds two bank0(two)-nk0()]TJ±¼mory (Memory00(Model)firstMode

```
outword(302h,400h)  ; set addr. to location 400h in Data memory
outword(302h,401h)  ; set addr. to location 400h in Program
                      memory
outword(302h,402h)  ; set addr. to loc. 400h in Data memory Bank
                      2
outword(302h,403h)  ; set addr. to loc. 400h in Program memory
```

The next step is to access the actual data. The data is read from or written to address Base IO using 16 bit IO instructions.

These assume a Base IO address of 300h. Therefore the port_value is 300h. Our pseudo-language IO instructions are of the form:

```
     write:
          outword(port_value, data_value);
     read:
          data_value = inword(port_value);
```

Examples:

```
1)   Assume address ctr. has been set to 400h in Data memory
     outword(300h,1234h) ; write 1234h to Data location 400h
2)   Assume address ctr. has been set to 400h in Program memory
     data_value_at_400h = inword(300h) ; read from Pgm location
    400h
```

After each memory access the address counter is incremented. A string of data may thus be read or written with only one address ’setup’.

Example:

```
     Assume address ctr. has been set to 400h in Data memory
     data_value_at_400h = inword(300h) ; read Data loc. 400h
     data_value_at_401h = inword(300h) ; read Data loc. 401h
     data_value_at_402h = inword(300h) ; read Data loc. 402h
```

Note:

The above example shows how to reach an address value which is not modulo 4. To reach 403h, ’go to’ 400h and read 3 times.

The auto increment feature results in very fast data transfers across the PC/AT bus, particularly when using the 80286/80386 REP INSW and REP OUTSW assembly language instructions. See the sendio and recvio functions.

The PC may access the Program RAM only when the the TMS320 is in the RESET (not running) condition.

Both the PC and the TMS320 may access the Data RAM at any time. They may thus communicate or pass large strings of data through the Data RAM.

An Example:

TMS320 writes the contents of the lower accumulator half to location 410h in Data memory:

```
               lrlk 1,>410    ; set auxiliary register 1 to >410
               larp 1         ; activate auxiliary register 1
               sacl *         ; write to Data RAM loc. >410
```

PC reads location 410h in Data memory (Model 250 mapped at 300h):

```
               mov dx,302h    ; set address
               mov ax,410h    ;
               out dx,ax      ; output address value
               mov dx,300h    ;
               in  ax,dx      ; read data value from RAM
```

or, in Turbo C:

```
               outport(0x302,0x412);
               data_value = inport(0x300);
```

## C. Start TMS32OC25 Operation

The 80_86 instruction to begin TMS32OC25 execution is a byte input at (Base IO address + 6). Our example shows a system with Base IO address at 3OOh.

```
     Assembly Language:        MOV DX,3O6h
                               IN AL,DX

     BASIC:                    INP(&H3O6)
     Debug:                    i3O6
     Turbo C:                  inportb(0x306);
```

## D. Halt TMS32OC25 Operation

The 80_86 instruction to halt TMS32OC25 execution is a byte input at (Base IO address + 7). Our example shows a system with the Base IO address at 3OOh.

```
     Assembly Language:        MOV DX,3O7h
                               IN AL,DX

     BASIC:                    INP(&H3O7)
     Debug:                    i3O7
     Turbo C:                  inportb(0x307);
```

## E. Interrupts

### E.1 The host toggles the INT pin on the TMS32OC25

The 8O_86 instruction to send an INT2 interrupt to the TMS32OC25 (to set the INT2 pin on the TMS32OC25 to LOW) is a byte input at (Base IO address + 5).

Our example shows a system with the Base IO address at 3OOh.

| | |
|---|---|
| Assembly Language: | MOV DX,3O5h<br>IN AL,DX |
| BASIC: | INP(&H3O5) |
| Debug: | i305 |
| Turbo C: | inportb(0x305); |

### E.2 Model 250 Interrupt to the host PC’s CPU

The Host CPU may be interrupted by the Model 250. In this way the TMS32OC25 may tell the PC that it has completed a computation or that it has some data to pass to the PC.

From the TMS320C25 point of view, the operation is quite simple. An input instruction at port 1 will, depending on the setting of jumpers JI1 and JI2 , send one of the following hardware interrupts to the host PC.

INT10
INT11
INT12
INT15
INT3
INT5

The host must, however, be prepared to receive the interrupt. The following steps must be taken:

1) The address of the Interrupt Service Routine (ISR) (the routine that you write to tell the processor what to do in case of an interrupt) must be placed in the Interrupt Routine Table. This table is located at the very beginning of memory in segment 0. The DOS manual suggests the use of DOS function call 25h to accomplish this end.

2) The Interrupt Mask Register (IMR) must be modified to accept the hardware INT signal. To enable an interrupt, its corresponding bit is set to zero .

3) The Interrupt Driver on the Model 250 must be removed from the Tri-State condition. This is done with a BYTE Write to Port (Base IO + 4). It can be returned to the Tri-State condition with a BYTE write to Port (Base IO + 5).

TSR250-10.C on the distribution diskette illustrates the setup and use of hardware interrupt 10. The host PC will beep upon receipt of an interrupt 10 from the Model 250. The interrupt may be generated from within D325 with the i1 (Input from Port 1) command. Make sure that JI1 is jumpered in the ’INT10’ position and that JI2 is NOT jumpered.

## E.3 Polling

In this practice, the host PC polls the memory location (see 4.B.1 or 4.B.3) and takes action (or keeps on polling) according to the value that it receives.

Example TMS320C25 Code

```
     larp      1
     lrlk      1,>400
     lack      73        ; send a ’73’ to location
     sacl      *; 400h in Data RAM
          .
          .
     DO FFT HERE
          .
     larp      1
     lack      88        ; send an ’88’ to location
     sacl      *         ; 400h in Data RAM
```

In this case the host would poll location 400H in Data RAM as shown in 4.B.1 and would know by the value received (either a 73 or an 88) that an FFT calculation was either in progress or had been completed.

F. <u>The IO Connectors</u>

The external 37 PIN connector is configured as follows:

```
                    GND       o
                                   o  INPUT CHANNEL 5
                    GND       o
                                   o  INPUT CHANNEL 4
                    GND       o
                                   o  INPUT CHANNEL 6
                    GND       o
                                   o  INPUT CHANNEL 7
                    GND       o
                                   o  INPUT CHANNEL 3
                    GND       o
                                   o  INPUT CHANNEL 2
                    GND       o
                                   o  INPUT CHANNEL 1
                    GND       o
                                   o  INPUT CHANNEL 0
                    GND       o
                                   o
                              o
                                   o
                              o
                                   o
                    GND       o
                                   o  OUTPUT CHANNEL 0
                    GND       o
                                   o  OUTPUT CHANNEL 1
                    GND       o
                                   o
                              o
                                   o
                              o
                                   o SERIAL PORT FSR
          SERIAL PORT DR      o
                                   o SERIAL PORT CLKR
          SERIAL PORT CLKX    o
                                   o SERIAL PORT DX
          SERIAL PORT FSX     o
```

J3 is the digital logic connector.

J3 Connector

| | | |
|---|---|---|
| D15 | o o | GND |
| D14 | o o | GND |
| D13 | o o | GND |
| D12 | o o | GND |
| D11 | o o | GND |
| D10 | o o | GND |
| D9 | o o | GND |
| D8 | o o | GND |
| A2 | o o | GND |
| A3 | o o | GND |
| A0 | o o | GND |
| A1 | o o | GND |
| D7 | o o | GND |
| D6 | o o | GND |
| D5 | o o | GND |
| D4 | o o | GND |
| D3 | o o | GND |
| D2 | o o | GND |
| D1 | o o | GND |
| D0 | o o | GND |
| WAIT | o o | -BRW |
| EXINT | o o | |
| -BSTRB | o o | |
| EXBIO | o o | RATE |
| -BIS | o o | RATE |

(O) means that the signal is an OUTPUT from the Model 250.
(I) means that the signal travels from the external module INTO the Model 250.
(IO) means that the signal is bi-directional.

**D0-D15** (IO) is the TMS320C25's buffered 16 bit data bus
**A0-A3** (O) are the TMS320C25 buffered address lines 0-3
**EXINT** (I) is the active low INT0 interrupt to the Model 250 **EXBIO** (I) is the active low BIO interrupt to the Model 250
**RATE** (O) is the output of the Model 250's programmable Timer
**-BRW** (O) is buffered TMS320 -RW Control Signal
**-BIS** (O) is buffered TMS320 -IS Control Signal
**-BSTRB** (O) is buffered TMS320 -STRB Control Signal
**WAIT** (I) used to insert Wait states in the TMS320 instruction cycle.

## G. TMS320E25 Operation

The TMS320E25 is similar to the TMS320C25, with the exception that it provides for 4K words of EPROM on chip. The programmed code may be secured, and secured code may be later erased but not read. The 4K words is used as Program space and is mapped at Program RAM locations 0 to FFFh . The TMS320E25 is programmed with a standard EPROM programmer capable of programming the 27C64 EPROM (21 Volt Programming voltage). An EPROM to TMS320E25 adapter is required. A TMS320E25 equipped Model 250 may be operated in 3 different ways.

1) Microprocessor Mode

Set J0 to the ’P’ (processor) position.
Operates just like a TMS320C25 and accesses external RAM in the 0-FFFh range upon being SET. The TMS320E25 may toggle itself to access its onboard EPROM by toggling the XF pin and then jumping to the desired location in the 0-FFFh Program space. Of course, the EPROM will have to have been previously programmed.

2) Computer Mode 1

Set J0 to the ’C’ (computer) position.
When the PC is turned on and after the Model 250 is SET, the TMS320E25 will begin execution at location 0 in its EPROM space. Locations 0-FFFh in external Program RAM are not used.

3) Computer Mode 2

Set J0 to the ’C’ (computer) position.
When power is turned on, the TMS320E25 will begin execution at location 0 in its EPROM space. Locations 0-FFFh in external Program RAM are not used. This mode enables the Model 250 to be used in standalone mode on backplane systems supplying only power and the PC Reset signal. This mode requires a PAL change at U14.

## H. The Latch

The TMS320 controls the following Model 250 parameters by writing to a latch mapped at port 1 in its IO space.

a. The next A/D input channel to be sampled.
b. The next D/A channel to be written to.
c. The 64K Data RAM segment to be accessed when the total Data RAM space is 128K words.

Format

| bit4 | bit3 | bit2 | bit1 | bit0 |
|---|---|---|---|---|
| Bank Select | D/A Select | muxa2 | muxa1 | muxa0 |

where

| | | |
|---|---|---|
| Bank Select | = 0 | for Data RAM Bank 0 (0-ffffh)[0-64K words] |
| | = 1 | for Data RAM Bank 1 (10000h-1ffffh)[64-128K] |
| D/A Select | = 0 | D/A Channel 0 |
| | = 1 | D/A Channel 1 |

Bits 0-2 select the Input Channel value, which ranges from 0 to 7.

Example Using D325:

```
          -o17,1    ; select A/D channel 7, Data RAM Bank 1,
                     D/A channel 0
```

# 5. D325 DEBUGGER

D325, the debugger, is the basic tool for 'manually' controlling the Model 250 board. The commands are explained below:

**A - Assemble**
The A command places TMS320C25 instructions into Program memory one line at a time. As each line is assembled, its corresponding object code is displayed on the screen. Hexadecimal numbers are preceded by a '>' and are often used for program memory references.

```
        -Ab90 <cr>
        B90  b >7b4  <cr>  ; hex - Program memory address
        B92  lack 17  <cr> ; decimal - Constant
        B93  sacl 22  <cr> ; decimal - Data memory address
```

Press <cr> to assemble the next instruction. D325 displays the assembled object code. Exit by entering a dot (.) followed by <cr>.

**B - Bank Select**
The B command selects the Data RAM bank (0 for the first 64K words or 1 for the second 64K words) for use in subsequent commands involving Data RAM. Used only with boards populated with 128K words of Data RAM.

**C - Convert**
The C command converts numbers in decimal notation to hexadecimal, and vice versa.

```
        -CH60<cr> converts the dec. number 60 to its
                  hex equivalent.
        -CD5b<cr> converts the hex number 5B to its
                  decimal equivalent.
```

**D - Display**
The D command displays Program or Data memory in hexadecimal notation. Syntax: D{p|d}address1,address2 where address2 is greater than address1.

```
        -DD54,110<cr>
```

displays all Data memory locations between 54h and 110h.

```
        -DP12a,170<cr>
```

displays all Program memory locations between 12Ah and 170h.

To pause the display, enter <Cntrl-s>.
To terminate the display prematurely, enter <ESC>.

**F - Fill**
The F command fills a portion of Program or Data memory with a 16 bit constant. The user is in turn prompted for the constant in hexadecimal notation, the first address at which the constant is to be placed, and the number of memory locations to be filled.

```
-Fd Enter constant in hex ? 200 <cr>
    Starting Address in hex ? BCD <cr>
    Number of Locations to Fill ? 300 <cr>
```

This example fills the 300 locations in Data memory starting at BCDh with 200h.

**G - Go**
The G command sets the TMS320C25 set/reset pin, thus setting the processor into operation.
a) No Breakpoint Operation.

```
-g<cr>
```

The TMS320C25 is set into operation and continues to run until the Halt command is issued.

b) Breakpoint

```
-g,breakpoint address<cr>
```

The TMS320C25 is set into operation and continues to run until the breakpoint address is reached or a timeout occurs. The TMS320C25 is then reset (halted) and the program counter is set back to zero. Then D325 will display the TMS320C25 status and registers, and optionally, one of the TMS320C25 Data Memory pages. The page to be displayed will have been set with the P command.

Note: To use the breakpoint option,
Program Memory between F00h and FFFh be available for use by D325, that is, not used by the applications program under examination.

**H - Halt**
The Halt command resets the TMS320C25 set/reset pin. This halts TMS320C25 operation.

```
-h<cr>
```

**I - Input**
The Input command causes the TMS320C25 to input a 16 bit word from the designated port (0-15) and then displays the word on the screen.

```
       -i5<cr>
       received A325h from port 5
```

**L - Log**
The L (log) command is used to log in boards which are not addressed at the default setting. Otherwise the default setting is assumed and the L command is not needed. All subsequent commands such as G (go) and H (halt) then refer to the logged on board. This permits the controlling of several Model 250 boards from within a single session of the D325 program.

**M - Move**
The Move command moves data from one block of program memory (the source) to another (the destination). D325 prompts the user for the start and end addresses of the source block and the beginning address of the destination block.

**O - Output**
The Output command causes the TMS320C25 to output a 16 bit word to the designated port (0-15).

```
       -oA531,7<cr>    outputs A531h to port 7
```

**P - Page**
Selects the page in Data Memory to be displayed upon reaching the breakpoint. See the G command description.

```
       -p4     to display page 4
```

**Q - Quit**
Used to exit the D325 program.

**R - Read from Disk**
This command reads the contents of a binary file into the TMS320C25 Program Memory. The user is prompted for the start and end addresses. In this way data and procedures from different files may be 'spliced' together in Program Memory.

```
       -rb:crypto.bin<cr>
        Start Address (0 <=  < FFF) ? 0 <cr>
        End Address (Start <    <=FFF) ? FFF <cr>
```

reads the file b:crypto.bin into Program Memory.

**S - Substitute**
The Substitute command allows the replacement of the 16 bit word at each Program or Data memory location.

```
        -sp556<cr>
         556 A321-8000<cr>
         557 A098- .<cr>
```

In the above example the user has replaced the word at address 556h in Program memory with 8000h. Enter <cr> to modify the next memory location. Exit by entering a dot (.) followed by <cr>.

**U - Unassemble**
This command dissassembles program memory contents into their TMS320C25 instruction codes one line at a time. Enter <cr> to dissassemble the next memory location. Exit by entering a dot (.) followed by <cr>.

```
        -u5<cr>
         5 ZAC          <cr>
         6 B F00        <cr>
         8 LARP 0       . <cr>
        -
```

**V - View**
The View command continously scans the desired range of Data RAM. In this way one may examine ares of Data RAM during Model 250 operation. Input syntax is similar to **Display**.

```
        -v400,41f<cr>
```

**W - Write to Disk**
This command writes the contents of TMS320C25 Program Memory to disk.

```
        -wa:fft.bin<cr>
```

writes Program Memory contents to a:fft.bin

**X - Read TI hexfile from disk**
This command reads the contents of a Texas Instruments format hex file into the TMS320C25 Program Memory. This allows programs written on the Texas Instruments Assembler to be run directly on the Model 250.

```
        -xb:fft.hex<cr>
```

# 6. A320 ASSEMBLER

A320 is the assembler for all Dalanco TI based DSP products. It features support for the TMS32010, TMS32020, and TMS320C25, and will automatically load binary files into the Model 250. It is also very fast. The assembly rate is approximately 1000 lines/second on AT class computers.

## The Command line

```
a320/<output mode> infile
```

The output mode can be:

```
     d  direct -- program is loaded to MODEL 250, no file
          produced
     b  binary -- this is compatible with D325 R command
     h  hex    -- Tektronics hex format
```

The default output mode is d.

If the output mode is b, a .BIN file is generated.
If the output mode is h, a .HEX file is generated.

## Instructions

```
Label                   ;comment
Label:                  ;comment
          Opcode    Operand         ;comment
Label     Opcode    Operand         ;comment
Label:    Opcode    Operand         ;comment
```

Opcodes MUST be in lower case. Labels can be upper and lower case. They are case-sensitive.

## Directives

**aorg** <position pointer>

Changes the current (program or data) position pointer to the one specified.

Example:

```
    aorg >10
```

**bss** <number of words>

Advances the current position pointer by the specified number of words.

Example:

```
One  bss  2
```

**data** <data word>,<data word>, etc...

Stores the data word. Note that this only works in program memory. It is useful for storing things like sine tables, etc., that will be read with TBLR instructions.

Example:

```
SinTab   data  >0,>324,>646,>964,>C7C,>F8D,>1294,>1590
```

**dseg**

Starts data segment. This is primarily useful for bss declarations of variables used. Use dend to return to program memory. Labels defined within dseg cannot be used as program labels, and program memory labels cannot be used for data memory. You may consider using equ $ for labels that can be used for more applications.

Example:

```
     dseg
One  bss  1
     dend
```

**end**

Ends assembly, no further instructions are recognized. This directive is not necessary if the file ends in a blank line.

Example:

```
     end
```

**equ**

Equates a label to an expression. The expression can be $, which indicates the current position pointer

Example:

```
Here equ  $
```

**go**

If the output mode is direct to DSP, go causes the DSP to start the program.

Example:

```
     go
```

**tms**    { 32010 | 32020 | 320c25 }

This specifies the instruction set to be used. The instruction set code may be abbreviated: 10, 20 or 25. The default is 320c25.

Example:

```
     tms   25
```

**iobase**    <Base IO value>

This specifies the base IO address to be used. This number is divisible by 8 and is between >100 and >3ff. The choice of Base IO address must agree with the IO jumper settings on the Model 250. The default value is >300.

Example:

```
     iobase    >310
```

**************************************************************

**Example program**

```
;    Sample Program  -  A/D to D/A Pass Thru
;    This Program reads from the A/D converter and then
;    writes the value to the D/A converter at the programmed
;    Sample Rate.
;
;    Model 250 Equates
ADPort    equ  2    ; M250 A/D Port
DAPort    equ  3    ; M250 D/A Port
TimerPt   equ  0    ; M250 Timer Port
LatchPt   equ  1    ; M250 Latch Port
IntMask   equ  2    ; = 2 for INT1
AckPt     equ  0    ; M250 Int. Acknowledge Port
DAC0      equ  0    ; DAC 0 latch value {Bit 3 = 0}
DAC1      equ  >8   ; DAC 1 latch value {Bit 3 = 1}
CHAN0     equ  0    ; A/D channel 0
CHAN1     equ  1    ; A/D Channel 1
CHAN2     equ  2    ; A/D channel 2
CHAN3     equ  3    ; A/D Channel 3
CHAN4     equ  4    ; A/D channel 4
CHAN5     equ  5    ; A/D Channel 5
CHAN6     equ  6    ; A/D channel 6
CHAN7     equ  7    ; A/D Channel 7
T10KHZ    equ  >f5d7     ; Timer Value for 10 Khz Sampling Rate
T100KHZ   equ  >f5fb     ;  "       "    " 100 Khz Sampling Rate
T200KHZ   equ  >f5fd     ;  "       "    " 200 Khz Sampling Rate

          dseg      ; Data Segment Locations
          aorg >0
Zero      bss  1
One       bss  1
Sample    bss  1
Num       bss  1
LatchVal  bss  1
TimerVal  bss  1
          dend
          aorg      0              ; Beginning of Program Memory
          b         Start

          aorg      >4
          b         HaveSamp       ; ISR for INT 1

          aorg      >10
Start:                             ; Initialize Hardware
          dint
          ldpk      0              ; Set up
          lack      IntMask        ; Interrupt Mask
          sacl      4
          ldpk      4
          zac
          sacl      Zero
          lalk      T200KHZ        ; Set Sampling Rate
          sacl      TimerVal
          out       TimerVal,TimerPt
          lack      1
          sacl      One
          lack      CHAN5     ; select next A/D Channel
          ork       DAC0      ; select next DAC Channel
          sacl      LatchVal
          out       LatchVal,LatchPt ; Output selections to Latch
          eint                       ; Enable Interrupt

GetSamp:                 ; Wait for Interrupt
          b         GetSamp

HaveSamp:                ; Interrupt Service Routine
          in        Sample,ADPort
          out       Sample,DAPort
          eint
          ret

          go             ; Start this program

          end            ; End of Code
```

# 7. LINK PACKAGE

This section describes the utility functions for controlling the Model 250 that are linkable to the user’s software. It should be noted that the user can also write his own functions for controlling the Model 250. The necessary Model 250 information is in Chapter 4 of this manual.

**int320(BASEIO)**

```
         BASEIO - Base IO Address of Model 250
```

Sends an INT2 interrupt to the TMS320C25 on the Model 250.

**go320(BASEIO)**

```
          BASEIO - Base IO Address of Model 250
```

Sets the TMS320C25  into operation.

**hlt320(BASEIO)**

```
          BASEIO - Base IO Address of Model 250
```

Resets the TMS320C25, halting its operation.

**sendio(X, LENGTH, START, BASEIO, PROGRAM_OR_DATA, DATA_BANK)**

```
          X      - Array of 16 bit words
          LENGTH - Number of words in array X to peek
          START  - Source start address in TMS320C25 memory
          BASEIO - Base IO Address of Model 250
          PROGRAM_OR_DATA = 0 for DATA MAEMORY access
                          = 1 for PROGRAM MEMORY access
          DATA_BANK - Bank of Data Memory being accessed
                    = 0 for Bank 0
                    = 1 for Bank 1
```

Copies the array X into Model 250 memory starting at memory location START.

```
          Example
          C Language:
               sendio(pgm1, 0x88, 0, baseio,1,0);
               /* copy 88h words from array pgm1 into Program RAM
                 starting at location 0 */
```

**recvio(X, LENGTH, START, BASEIO, PROGRAM_OR_DATA, DATA_BANK)**

```
X      - Array of 16 bit words
LENGTH - Number of words in array X to peek
START  - Source start address in TMS320C25 memory
BASEIO - Base IO Address of Model 250
PROGRAM_OR_DATA = 0 for DATA MEMORY access
                = 1 for PROGRAM MEMORY access
DATA_BANK - Bank of Data Memory being accessed
          = 0 for Bank 0
          = 1 for Bank 1
```

Copies LENGTH words of Model 250 memory starting at START into array X.

```
Example
C Language:
     recvio(&x,0x300,0x1000,baseio,0,0);
     /* copy 300h words from Data Ram Location 1000h
    into array x  */
```

**NOTE: See the FFTP250.C and other listings on the diskette for examples of the use of the Link Package functions.**

## 8. **A/D CONVERTER and D/A CONVERTER**

### A. A/D Converter

Data

|  |  |
|---|---|
| Type | : Maxim 162 or equivalent |
| Maximum Rate | : 250 Khz |
| Input Voltage Range | : +/- 5 Volts |

Operation

The Analog to Digital conversion is a two step process from the software or high level point of view. The ADC resides at Port 2 in the TMS320C25's IO space.

First, **STEP 1**, the ADC must be told to begin the conversion. This is done by bringing the R/C pin ( C for convert) low. This strobe can be generated by the output of the onboard Timer. See Section 4.A.

**Step 2**: The ADC announces that it has completed the conversion by generating an INT1 interrupt to the TMS320C25. Upon its reception the TMS320C25 branches to a section of code where it can now input the result, eg.

```
in 6,2
```

inputs the result to Data Memory location 6

## B. D/A Converter

Data

| | |
|---|---|
| Type | : PMI 8222 or equivalent |
| Rate | : 250 Khz + |
| Output Voltage Range | : +/- 5 Volts |

Operation

The D/A Converter is mapped as Write Port 3 in the TMS320C25's IO space. Output of a value to Port 3 will result in the D/A converter producing the corresponding voltage at its output at the next arrival of the clock strobe from the programmable timer. Since there are two D/A channels, the desired channel is selected by first writing the appropriate value to bit 3 of the Latch Port (Port 1). Both channel outputs will be simultaneously updated by the Timer strobe. Make sure that the Timer is running.

| Desired Output Voltage | Value | D325 Instruction |
|---|---|---|
| - 5 Volts | 0 | o0,3 |
| . | . | . |
| 0 Volts | 800 | o800,3 |
| . | . | . |
| 4.9976 Volts | FFF | oFFF,3 |

See the distribution diskette for programming examples involving the A/D and D/A converters.

## 9. <u>**FFT ROUTINES**</u>

The FFT routines are labelled:

| | |
|---|---|
| FFT024.B25 | : 1024 point complex |
| FFT512.B25 | : 512 point complex |
| FFT256.B25 | : 256 point complex |

and are all identical with the exception of length dependent items such as the sine table and one constant.

In each case the **real** values are loaded in at **200H** and the **imaginary** values are loaded in at **600H**. Thus locations 200H thru 9FFH are reserved for input.

The results of the FFT replace the input.

<u>Example: a 'manual' FFT in D325.</u>

```
-rfft024.b25<cr>              ; load program
 Start Address ? 0 <cr>
 End Address ? FFF <cr>
- .......................... ; input data using
                              ; f,s,m commands
                              ; or load in from a
                           ;file with r command
-g <cr>                   ; Start TMS320C25
-h                        ; Stop
-dp200,9ff <cr>            ; look at results
```

The FFT routine outputs STARTING and COMPLETED words to the Register. (See the example in the section which describes **Polling**). This is to inform the host PC as to what the TMS320C25 is doing.

<u>Example: FFT in a user's program.</u>

If you program in C, see the FFTP250.C file on the distribution diskette. This program is also on your disk in executable form.
A Color, EGA, or VGA Adapter is required.

## 10. DISPLAY PROGRAM - Time and Frequency Domain Display

Displays input signals on the Color Graphics, Hercules, or EGA/VGA Graphics Adapters. This program enables the PC screen to function as a signal display and as a Frequency Spectrum domain display with sample rates ranging to 200 Khz. This program is only useful with Model 250 boards equipped with the on-board A/D converter.

1) Type **DISPLAY <cr>**
2) DISPLAY will ask you for the IO addressing of the Model 250 board.

Example
IO Address Base ? **300 <cr>**
Note that the answers are hexadecimal numbers.

3) Connect a signal source to the INPUT connector. Make sure the **signal does not exceed +/- 5 Volts with respect to the ground level of the PC.**
4) The signal should appear on the screen.

The Numeric Keypad on the right hand side of the PC keyboard controls the operation of the DISPLAY program.

**<--** and **-->** decrease and increase the sample rate, respectively.
**CNTRL <--** and **CNTRL -->** halve and double the sample rate, respectively.
**UP ARROW** and **DOWN ARROW** control the 'gain' of the display.
**P** activates the display Trigger feature, much like the 'Trigger Level' control on an oscillosope.

The **Function Keys** [F1-F8] are used for input channel selection.

To display the frequency spectrum domain, enter **SPACE BAR**.

To exit DISPLAY, enter **Q**.

## 11. **DATA TRANSFER**

### A. RECORD - Model 250 to PC Transfer

This program transfers continuous data from the A/D Converter to the host PC's hard disk. The maximum amount of data which may be recorded is dictated by the remaining free space on the disk. The maximum direct-to-disk sampling rate depends upon the computer and type of hard disk used. Recording rates of 50 Khz are typical on AT-class computers.
The RECORD program prompts the user for the following data:

1. IO addressing of the Model 250.
2. Sample Rate
3. Disk file name (for writing)

### B. PLAYBACK - PC to Model 250 Transfer

This program transfers data from the PC's hard disk to the Model 250 for output to the D/A converter. The program prompts for the following data:

1. IO addressing of the Model 250.
2. Sample Rate
3. Disk file name (for reading)

You may connect the Model 250 to a stereo by connecting the preamp's Tape Out output (which would ordinarily go to a tape deck) to the top input connector (Channel 5 on the A/D converter) on the Model 250. The output from the D/A 0 on the Model 250 can be connected to the Tape or Tuner input on the preamp. You will probably need BNC to Phono jack adapters (available in most electronic/stereo stores).

## C. EDIT25 Program

This program enables the user to view captured data files on the Graphics computer screen. EDIT25 can scroll through a file, play and loop segments through the D/A converter, and create and import file sections. Additional features include file editing, cut and paste, and mixing functions.
  Enter <?> for the command list.

## D. DELAY Program

The Delay program demonstrates how a digital delay may be implemented using the Model 250. This early version of Delay requires that a RAM DISK be configured on the host computer. The DOS manual explains how this is done. The size of the RAM DISK should be at least

4 * (Number of seconds of desired delay * sampling rate) bytes.

For example, a 10 Khz sampling rate with a 2 second delay would require a RAM DISK with at least 80K bytes of free space.

## 12. DIGITAL FILTER IMPLEMENTATION

One popular use for the TMS320C25 is in the implementation of digital filters. The TI publication Digital Signal Processing Applications with the TMS320 Family is a good reference on the subject.

Refer to the listings of the **Length-80 Linear-Phase Passband FIR Filter** (Appendix A) in Chapter 3 of the Applications book. This is a 1 Khz to 4 Khz bandpass filter.
You will note that there are two different listings of the same program. The one listing is designed for the TMS32010, and can be run, with only slight modification, on second generation TMS320 DSPs as well. The other listing is for the TMS32020 and TMS320C25 only, as it makes use of the larger Data Memory space and expanded instruction capability of these devices.

The listings in the Applications book must be modified slightly to run on the Model 250. The steps taken to modify these programs have to do with the conversion of data from the format of the A/D and D/A converters to the twos complement format which is used in the actual filter calculations.

1) Modify the statements which convert the data from the hardware's format to two's complement. These statements immediately follow the IN instruction.

```
     Replace:
             lac XN       with         lac XN,4
             xor MASK1                 sub ONE,15
```

2) Modify the statements which convert the data from two's complement to the hardware's format. These statements immediately precede the OUT instruction.

```
      Replace:
               lac MASK2      with      lac YN,12
               xor YN                   addh MASK
                                     where MASK equu 800h
```

3) The port numbers in the IN and OUT instructions are of course hardware specific. The input port value (the A/D Converter) on the Model 250 is 2. The output port (the D/A Converter) value is 3.

The modified program (FIR250.ASM) is on your disk.

Examine FIR250.ASM with your text editor, and run the program by typing

**A320 FIR250<cr>**

at the DOS prompt. You will need to modify the listing FIR250.ASM if you are not using the default Base IO value.

## GEN250 CODE GENERATOR

GEN250 is a digital filter code generator which produces FIR filter code for the Model 250. In this way, filter responses may be viewed and studied minutes after their design.

GEN250 takes its input filter specifications from a file or interactively from the user console. The input filter specifications format consists of headings, each followed by its relevant data (if applicable).

The heading categories are:

| | |
|---|---|
| coefficients | : followed by the filter coefficients, each |
| | : appearing on a new line. The maximum number |
| | : of coefficients is 120. |
| rate | : followed by the sampling rate, in Hz. |
| quantize_C | : followed by the number of bits for |
| | : quantization (usually 16). |
| normalize_ON | : one of these is selected. The default is |
| normalize_OFF | : OFF. |
| end | : end of filter specification. |

**Normalize_ON** would be used in the case, for example,in a smoothing formula where you wanted GEN250 to normalize the coefficient values. Thus, after specifying:

```
                    -3
                    12
                    17
                    12
                    -3
```
as your coefficient values, GEN250 would divide these numbers by their sum (35).

In most cases, including those where the coefficients have been set by a filter design program, use **normalize_OFF**.

**quantize_C** indicates the number of bits used to represent the filter coefficient values. Currently the only allowed option is 16 bits.

The Output of GEN250 is a TMS320C25 source code file which may be assembled and run with A320. An iobase statement will have to be added to the source file in cases where a base IO address other than the default value of 300h is used.

<u>Example 1 </u>: Interactive Session

```
     >GEN250 OutCodeFile.asm
      rate
      10000.
      coefficients
      -3/35
      12/35
      17/35
      12/35
      -3/35
      quantize_C
      16
      normalize_OFF
      end
```

If a heading is entered incorrectly, GEN250 will take the entered string to be a variable and will display a number or ’syntax error’. In this case, reenter the heading name correctly.

Example 2 : Using an Input Specification File

```
>GEN250 <InputFilterSpec OutCodeFile.asm
```

This is useful in cases where there are many coefficients or where the coefficients have been produced by a filter design program.

Example of an Input Filter Specification File:

```
rate
2000
coefficients
     .
     .



     .
quantize_C
16
normalize_OFF
end
```

An example of such a file is the INSPEC.SPC file on the distribution diskette. Its coefficients were created with the MACPARK program (see below).

In many filter design programs the coefficient list is produced as an output file. This list (it should be a list of floating point numbers in the range [-1.0 to 1.0]) may be inserted under the **coefficients** heading in your Input Specification File.

One very useful FIR design program is EQFIR, which is described and listed in Reference 1, Section 5.1. For those who don't want to type the entire listing into their computers, the program is available on DOS disk (for a small fee) as MACPARK (Disk #2090 from Sector Systems (617) 639-2625). Reference 1 is required for an understanding of the program, and is a worthwhile investment for the many other DSP programs it contains.

Most, if not all, of the commercial filter design programs, which feature a more intuitive user interface than EQFIR/MACPARK, also output coefficients in a form which can be readily used by GEN250.

## Reference

1. Programs for Digital Signal Processing, IEEE Press.

## APPENDIX A

### MODEL 250 IO PORT ASSIGNMENTS

| Port | Read | Write |
|---|---|---|
| 0 | Interrupt Ack | Program Timer |
| 1 | Interrupt PC | Latch Port |
| 2 | A/D Converter | |
| 3 | | D/A Converter |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |

### HOST PC IO ASSIGNMENTS

Port value = IO Base + Offset

| Offset | Read | Write |
|---|---|---|
| 0 | RAM access (word) | RAM access (word) |
| 1 | | |
| 2 | | RAM address (w) |
| 3 | | |
| 4 | Int. Acknowledge (b) | Allow Int. (b) |
| 5 | Interrupt TMS320 (b) | Disallow Int. (b) |
| 6 | Set TMS320 (b) | |
| 7 | Reset TMS320 (b) | |

## APPENDIX B

### TMS320C25 Assembly Language

The detailed reference on this subject is the TMS320C25 User’s Guide. Please refer to this publication for detailed programming information.

**TMS32010 Compatible Instructions**

1. Accumulator Instructions

| | |
|---|---|
| ABS | - Absolute value of accumulator |
| ADD | - Add to accumulator with shift |
| ADDH | - Add to upper word in accumulator |
| ADDS | - Add to accumulator no sign extension |
| AND | - AND with accumulator |
| LAC | - Load accumulator with shift |
| LACK | - Load acc. immediate |
| OR | - OR with acc. |
| SACH | - Store upper word acc. with shift |
| SACL | - Store lower word acc. |
| SUB | - Subtract from acc. with shift |
| SUBC | - Conditional subtract for division |
| SUBH | - Subtract from upper word acc. |
| SUBS | - Subtract from acc. with no sign extension |
| XOR | - XOR with acc. |
| ZAC | - Zero acc. |
| ZALH | - Zero acc. and load upper word |
| ZALS | - Zero acc. and load lower word w. no sign extension. |

2. Auxiliary Register and Data Page Instructions

| | |
|---|---|
| LAR | - Load auxiliary register |
| LARK | - Load auxiliary register immediate |
| LARP | - Load auxiliary reg. pointer immediate |
| LDP | - Load Data memory page pointer |
| LDPK | - Load Data memory page ptr immediate |
| MAR | - Modify aux. register and pointer |
| SAR | - Store auxiliary register |

3. Branch Instructions

| | |
|---|---|
| B | - Branch unconditionally |
| BANZ | - Branch on aux. register not zero |
| BGEZ | - Branch if acc. .GE. zero |
| BGZ | - Branch if acc. .GT. zero |
| BIOZ | - Branch on BIO interrupt |
| BLEZ | - Branch if acc. .LE. zero |
| BLZ | - Branch if acc. .LT. zero |
| BNZ | - Branch if acc. .NE. zero |
| BV | - Branch on overflow |
| BZ | - Branch if acc. .EQ. zero |
| CALA | - Call Subroutine from accumulator |
| CALL | - Call Subroutine immediate |
| RET | - Return from Subroutine or Interrupt Handler |

3. T Register, P Register, Multiply Instructions.

| | |
|---|---|
| APAC | - Add P register to acc. |
| LT | - Load T reister |
| LTA | - LT and APAC in one instruction |
| LTD | - LT, APAC, DMOV in one instruction |
| MPY | - Multiply T reg, result in P register |
| MPYK | - Multiply T reg immediate, result in P reg. |
| PAC | - Load accumulator from P register |
| SPAC | - Subtract P register from accumulator |

4. Control Instructions

| | |
|---|---|
| DINT | - Disable interrupt |
| EINT | - Enable interrupt |
| LST | - Load status register |
| NOP | - No op |
| POP | - Pop stack to accumulator |
| PUSH | - Push stack from accumulator |
| ROVM | - Reset overflow mode |
| SOVM | - Set overflow mode |
| SST | - Store status register |

5. I/O and Data Memory Instructions

| | |
|---|---|
| DMOV | - Contents of Dmem[i] --> Dmem[i+1] |
| IN | - Input Data from Port |
| OUT | - Output Data to Port |
| TBLR | - Table read from Program to Data ram |
| TBLW | - Table write from Data to Program ram |

**Additional, Non-TMS32010 Instructions**

6. Accumulator Memory Reference Instructions

```
ADDC     - Add to accumulator with carry
ADDK - Add to accumulator short immediate
ADDT - Add to acc. w. shift specified by T Register
ADLK - Add to accumulator long immediate
ANDK - And immediate w. accumulator with shift
CMPL - Complement accumulator
LACT - Load acc w. shift specified by T Register
LALK - Load accumulator long immediate w. shift
NEG  - Negate accumulator
NORM - Normalize accumulator
ORK  - Or immediate with accumulator with shift
ROL  - Rotate accumulator left
ROR  - Rotate accumulator right
SBLK - Subtract from acc. long immed. w. shift
SFL  - Shift accumulator left
SFR  - Shift accumulator right
SUBB - Subtract from accumulator w. borrow
SUBT - Subtract from acc. shift spec. by T register
SUBK - Subtract from acc. short immediate
XORK - Exclusive Or immed. w. acc. w. shift
ZALR - Zero low acc., load hi acc w. rounding
```

7. Auxiliary Registers and Data Pointer Instructions

```
ADRK - Add to auxiliary register short immed.
CMPR - Compare auxiliary register with AR0
LRLK - Load auxiliary register long immediate
SBRK - Subtract from aux. register short immed.
```

8. T Register, P Register, and Multiply Instructions

```
LPH  - Load high P register
LTP  - Load T register and store P register in acc.
LTS  - Load T register and subtract previous product
MAC  - Multiply and accumulate
MACD - Multiply and accumulate with data move
MPYA - Multiply and accumulate previous product
MPYS - Multiply and subtract previous product
MPYU - Multiply unsigned
SPH  - Store high P register
SPL  - Store low P register
SPM  - Set P register output shift mode
SQRA - Square and accumulate
SQRS - Square and subtract previous product
```

9. <u>Branch and Call Instructions</u>

| | | |
|---|---|---|
| BACC | - | Branch to address specified by accumulator |
| BBNZ | - | Branch if TC bit .NE. 0 |
| BBZ | - | Branch if TC bit .EQ. 0 |
| BC | - | Branch on carry |
| BNC | - | Branch on no carry |
| BNV | - | Branch if no overflow |

10. <u>IO and Data Memory Operations</u>

| | | |
|---|---|---|
| BLKD | - | Block move data memory to data memory |
| BLKP | - | Block move program memory to data memory |
| FORT | - | Format serial port registers |
| RFSM | - | Reset serial port frame synchronization mode |
| RTXM | - | Reset serial port transmit mode |
| RXF | - | Reset external flag |
| SFSM | - | Set serial port synchronization mode |
| STXM | - | Set serial port transmit mode |
| SXF | - | Set external flag |

11. <u>Control Instructions</u>

| | | |
|---|---|---|
| BIT | - | Test bit |
| BITT | - | Test bit specified by T register |
| CNFD | - | Configure block as data memory |
| CNFP | - | Configure block as program memory |
| IDLE | - | Idle until interrupt |
| LST1 | - | Load status ST1 |
| POPD | - | Pop top of stack to data memory |
| PSHD | - | Push data memory value to stack |
| RC | - | Reset carry bit |
| RPT | - | Repeat instruction as spec. by data mem value |
| RPTK | - | Repeat instruction as spec. by immed. value |
| RSXM | - | Reset sign extension mode |
| RTC | - | Reset test/control flag |
| SC | - | Set carry bit |
| SHM | - | Set hold mode |
| SST1 | - | Store status ST1 |
| SSXM | - | Set sign extension mode |
| STC | - | Set test/control flag |
| TRAP | - | Software interrupt |

## REFERENCES

1.TMS320C25 User’s Guide , Reference no. SPRUOO12A
2.TMS320 Development Support Reference Guide, Ref no. SPRU007A
3.Digital Signal Processing Applications with the TMS320 Family,
Ref no. SPRA012