MODEL 5000

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.

Tel: (716) 473-3610
Fax: (716) 271-8380
E-mail: sales@dalanco.com
http://www.dalanco.com

Copyright (c) 1995-1998 Dalanco Spry

TABLE OF CONTENTS

1. INTRODUCTION

This manual is the reference to the Model 5000 Digital Signal Processor Board and its accompanying software. The reference for the Texas Instruments TMS320C5x Digital Signal Processor and its assembly language is the TMS320C5x User's Guide available from TI.

Partial List of References

1. TMS320C5x User's Guide , Reference no. SPRUO56B
2. Digital Signal Processing Applications with the TMS320 Family, Ref no. SPRA012

2. INSTALLATION

A. Make a backup working copy of the software.

B. Set the address jumpers for the IO mapping.

These jumpers determine the Base IO addressing at which the Model 5000 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, is an equivalent representation). 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 300h 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 zero and no jumper to a bit set to one.

The Base IO address is selectable on 8 byte boundaries.

Note that in systems with multiple Model 5000 boards, each Model 5000 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
```

Fig. 1

```
 A3               A9
```

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 5000 is fitted with 64K words of Program RAM. The Program RAM resides in the four ICs directly above the TMS320.
The RAM ICs are Toshiba TC55464P-15 or equivalent.

D. Data RAM Size

The Model 5000 Data RAM resides in the two IC locations on the leftmost side of the board. For 128K x 16 total RAM, these are Paradigm 40124P-20 or equivalent.

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 5000 is to use the D5000 debugger. 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 to 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 D5000:
Type D5000 at the MS-DOS prompt. A dash (-) appears as the new prompt.

Display memory:
To display the first 8Oh words of Program memory (0 to 7fh), type

```
-dp0,7f<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 D5000 awaits your entry. Type

b 7 (branch to address 7) **<cr>**

Note: If D5000 responds with ???, you made an entry error. Reenter the instruction. After successful assembly D5000 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 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 SETC INTM  <cr>
8 LDP #4 <cr>
9 LACL #80 <cr>
A TBLR 0 <cr>
B LACL #81 <cr>
C TBLW 0 <cr>
D B 0DH <cr>
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 .

<u>Disassemble the program</u>:
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 5000 Program memory space (64K words) to disk. Make sure that you have 128 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 D5000 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) ?

Before writing more programs, see Appendix D, Programming Guidelines for the Model 5000, for important information. See also the programming examples on the distribution diskette.

4. **HARDWARE FEATURES**

A. Onboard Timer

The onboard 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 5000 Timer diagram:

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

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

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 D5000 **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 5000 holds two banks of memory. The first is the Program memory, which contains the TMS320 program instructions. The second is the Data memory, which contains data.

From the TMS320 point of view, access to these memory banks is quite simple and is a natural part of the TMS320 instruction set.

From the PC point of view, memory access is a little more complex, and is explained below. This information does not have to be understood in detail by those programmers using the subroutines in the Link Package (see Chapter 7).

The first step is to set the address counter on the Model 5000 to the desired ’address value’. The format of the 16 bit ’address value’ is as follows:

a15 a14 a13 a12 a11 a10 a9 a8 a7 a6 a5 a4 a3 a2 BK12 DP

where

a15 - a2 are the 14 MSBs of the address to be accessed.

DP designates whether the access is to be for Program or Data memory
= 0 for Data
= 1 for Program

BK12 is the 64K segment designator for Data memory in the case where the Data memory is populated with 128K words RAM.
= 0 lower 64K words
= 1 upper 64K words

DP	BK12	Memory Access by Host
0	0	Data RAM Bank 0
0	1	Data RAM Bamk 1
1	0	Program RAM
1	1	Lockout Condition

The Lockout Condition is enabled by setting the DP and BK12 bits to 1. During this condition, the host denies itself access to the data RAM. The data RAM becomes the private domain of the TMS320 and is accessed by the DSP at the maximum rate allowed by the data RAM's access time. The host may subsequently turn off the Lockout Condition by writing the values of DP and BK12 to the 'address value' when RAM access is needed.

Remember to use 16 bit word (not 8 bit byte) IO instructions when accessing the address counter or memory on the Model 5000.

Examples:

These assume a Base IO address of 300h. Therefore the port_value for address setting is 302h. Our pseudo-language IO instructions are of the form:

```
        outword(port_value, address_value);
```

```
outword(302h,C00h)  ; set addr. to location C00h in Data memory
                      Bank 0
outword(302h,C01h)  ; set addr. to location C00h in Program memory
outword(302h,C02h)  ; set addr. to loc. C00h in Data memory Bank 1
```

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 C00h in Data memory
   outword(300h,1234h) ; write 1234h to Data location C00h
2) Assume address ctr. has been set to C00h in Program memory
   data_value_at_C00h = inword(300h) ; read from Pgm location C00h

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 C00h in Data memory
        data_value_at_C00h = inword(300h) ; read Data loc. C00h
        data_value_at_C01h = inword(300h) ; read Data loc. C01h
        data_value_at_C02h = inword(300h) ; read Data loc. C02h
```

Note:

The above example shows how to reach an address value which is not modulo 4. To reach C03h, ’go to’ C00h and read 3 times.

The auto increment feature results in very fast data transfers across the PC AT bus, particularly when using the REP INSW and REP OUTSW assembly language instructions. See the sendio and recvio functions in the Link Package (Chapter 7).

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 C10h in Data memory:

```
                lar 1,#0C10h   ; set auxiliary register 1 to C10h
                mar *,1        ; activate auxiliary register 1
                sacl *         ; write to Data RAM loc. C10h
```

PC reads location C10h in Data memory (Model 5000 mapped at 300h):

```
                mov dx,302h    ; set address
                mov ax,0C10h   ;
                out dx,ax      ; output address value
                mov dx,300h    ;
                in  ax,dx      ; read data value from RAM
```

or, in Turbo C:

```
                outport(0x302,0xC10);
                data_value = inport(0x300);
```

C. Start TMS320C5x Operation

The host PC instruction to begin TMS320C5x execution is a byte input at (Base IO address + 6). Our example shows a system with Base IO address at 300h.

```
Assembly Language:      MOV DX,306h
                        IN AL,DX ; TMS320 /RS pin --> HIGH

BASIC:              INP(&H306)
Debug:              i306
Turbo C:            inportb(0x306);
```

D. Halt TMS320C5x Operation

The host PC instruction to halt TMS320C5x execution is a byte input at (Base IO address + 7). Our example shows a system with the Base IO address at 300h.

```
Assembly Language:      MOV DX,307h
                        IN AL,DX ; TMS320 /RS pin --> LOW

BASIC:              INP(&H307)
Debug:              i307
Turbo C:            inportb(0x307);
```

E. Interrupts

E.1 The host toggles the INT pin on the TMS320C5x

The host PC instruction to send an INT2 interrupt to the TMS320C5x (to set the INT2 pin on the TMS320C5x to LOW) is a byte input at (Base IO address + 5).

Our example shows a system with the Base IO address at 300h.

```
Assembly Language:      MOV DX,305h
                        IN AL,DX

BASIC:              INP(&H305)
Debug:              i305
Turbo C:            inportb(0x305);
```

E.2 Model 5000 Interrupt to the host PC’s CPU

The Host CPU may be interrupted by the Model 5000. In this way the TMS320 may tell the PC that it has completed a computation or that it has some data to pass to the PC.

From the TMS320 point of view, the operation is quite simple. An input instruction to 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 task.

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

3) The Interrupt Driver on the Model 5000 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).

TSR5_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 5000. The interrupt may be generated from within D5000 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 and takes action (or keeps on polling) according to the value that it receives.

Example TMS320C5x Code

```
mar     *,1
lar     1,#0C00h
lacl    #73      ; send a '73' to location
sacl    *        ; C00h in Data RAM
    .
    .
DO FFT HERE
    .
mar     *,1
lacl    #88      ; send an '88' to location
sacl    *        ; C00h in Data RAM
```

In this case the host would poll location C00H in Data RAM 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	+5V
-BSTRB	o	o	+5V
-EXBIO	o	o	RATE
-BIS	o	o	-PCRESET

(O) means that the signal is an OUTPUT from the Model 5000.
(I) means that the signal travels from the external module INTO the Model 5000.
(IO) means that the signal is bi-directional.

D0-D15 (IO) is the TMS320's buffered 16 bit data bus
A0-A3 (O) are the TMS320 buffered address lines 0-3
-EXINT (I) is the active low INT0 interrupt to the Model 5000
-EXBIO (I) is the active low BIO interrupt to the Model 5000
RATE (O) is the output of the Model 5000's Onboard Timer.
-BRW (O) is buffered and inverted TMS320 RW Control Signal. Read active low. Write active high.
-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. Active high. Driven low by the external device when ready in order to end the IO cycle.
-PCRESET (I) The PC Host Reset Signal. Low during PC Power Up, then high.

G. The Latch

The TMS320 controls the following Model 5000 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.

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)[64K-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 D5000:

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

H. Serial Ports

The serial port brought out to the external 37 pin connector is configured in the following manner when populated with **PALSER1** at U80:

CLKR	- Input
CLKX	- Input
FSX	- Input
FSR	- Input

The TDM Port pins are located to the right of the TMS320 at J0. Utmost care must be employed when using these pins. They are not buffered.

TFSR	o	o	TDR
TCLKX	o	o	TCLKR
TFSX	o	o	TDX

5. D5000 DEBUGGER

D5000, the debugger, is the basic tool for ’manually’ controlling the Model 5000 board. The commands are explained below:

A - Assemble
The A command places TMS320C5x instructions into Program memory one line at a time. As each line is assembled, its corresponding object code is displayed on the screen.

```
-Ab90 <cr>
B90  b 7b4h  <cr>  ; hex - Program memory address
B92  lacl #17 <cr> ; decimal - Constant
B93  sacl 22  <cr> ; decimal - Data memory address
```

Press <cr> to assemble the next instruction. D5000 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>.

E - Extended Instructions
The E command provides debugging features such as trace and single step. The program under study must conform to the following standard format:
1. The statement at address 0 must be a branch to the beginning of the program, which itself must reside at address 30h or above.
2. Certain Data RAM areas are reserved for use by D5000. Not to be used by the user program.
3. Certain Program RAM areas are reserved for use by D5000. Not to be used by the user program.
4. E commands make use of the TMS320 TRAP interrupt. This interrupt should not be used by the user program.

See FIR5000.ASM on the distribution diskette for an example of a user program compatible with the D5000 extended commands.

Commands
ESC - Exit Extended Instruction Mode. Return to main menu.
T - Trace Instruction. Enter number of instructions to trace through.
G - Execute Program without breakpoints.
N - Set and go to Next Breakpoint.
D - Display External Data RAM. Sets display windows of Data RAM.
S - Substitute values in Data RAM.
A - Assemble in Program RAM. To enter Data values in Program RAM, use the ’data’ instruction statement.
U - Unassemble command.
F2 - Function Key 2 is the Single Step key.

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 TMS320 set/reset pin, thus setting the processor into operation. The TMS320 HOLD pin is deactivated.
a) No Breakpoint Operation.

```
-g<cr>
```

The TMS320 is set into operation and continues to run until the Halt command is issued.

b) Breakpoint

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

The TMS320 is set into operation and continues to run until the breakpoint address is reached or a timeout occurs. The TMS320 is then reset (halted) and the program counter is set back to zero. Then D5000 will display the TMS320 status and registers, and optionally, one of the TMS320 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 D5000, that is, not used by the applications program under examination.

H - Halt
The Halt command resets the TMS320 set/reset pin. This halts TMS320 operation. The HOLD pin on the TMS320 is activated.

```
-h<cr>
```

I - Input
The Input command causes the TMS320 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 5000 boards from within a single session of the D5000 program.

M - Move
The Move command moves data from one block of program memory (the source) to another (the destination). D5000 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 TMS320 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 D5000 program.

R - Read from Disk
This command reads the contents of a binary file into the TMS320 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 disassembles program memory contents into their TMS320C5x instruction codes one line at a time. Enter <cr> to disassemble the next memory location. Exit by entering a dot (.) followed by <cr>.

```
-u5<cr>
 5 LACL #0    <cr>
 6 B 0F00h    <cr>
 8 NOP              . <cr>
-
```

V - View
The View command continuously scans the desired range of Data RAM. In this way one may examine ares of Data RAM during Model 5000 operation. Input syntax is similar to **Display**.

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

W - Write to Disk
This command writes the contents of TMS320 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 TMS320 Program Memory. This allows programs written on the Texas Instruments Assembler to be run directly on the Model 5000.

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

6. A5000 ASSEMBLER

A5000 is the assembler for the Model 5000.

The Command line

```
a5000 {-<output mode>} infile
```

The output mode can be:

```
     d  direct  -- program is loaded to MODEL 5000, no file
                   produced.
     b  binary -- this is compatible with D5000 R command
     a  ascii   -- this is compatible with the C language array
                   syntax
```

The default output mode is d.

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

Instructions

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

Directives

aorg <position pointer>

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

Example:

```
aorg 10h
```

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  0,324h,646h,964h,0C7Ch,0F8Dh,01294h,1590h
```

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
```

iobase <Base IO value>

This specifies the base IO address to be used. This number is divisible by 8 and is between 100h and 3FFh. The choice of Base IO address must agree with the IO jumper settings on the Model 5000. The default value is 300h.

Example:

```
iobase      310h
```

****

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 5000 Equates
PDWSR           equ 028h
IOWSR           equ 029h
CWSR            equ 02Ah
CWSR_VAL        equ 4         ; IO port 0-7
```

```
IOWSR_VAL       equ 0Ch     ; 7 wait states for port 2-3
PDWSR_VAL       equ   0     ; 0 s/w wait states for memory access
IMR             equ   4     ; Interrupt Mask Location
ADPort          equ   2     ; A/D Port
DAPort          equ   3     ; D/A Port
TimerPt         equ   0     ; Timer Port
LatchPt         equ   1     ; Latch Port
IntMask         equ   2     ; = 2 for INT2
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   0f5d7h        ; Timer Value for 10 Khz
T100KHZ         equ   0f5fbh        ;  "      "    " 100 Khz
T200KHZ         equ   0f5fdh        ;  "      "    " 200 Khz

;       OnChip Data Memory Locations
                dseg
                aorg  0
Sample          bss   1
Num             bss   1
LatchVal0       bss   1
LatchVal1       bss   1
TimerVal        bss   1
                dend


;       Start Program
        b           Start
        aorg        4
        b           HaveSamp
        aorg        20h
Start:
        setc        INTM            ; disable interrupts
        ldp         #0
        lacl        #IntMask        ; Interrupt Mask
        sacl        IMR
        lacc        #CWSR_VAL       ; set wait states
```

```
                sacl            CWSR
                lacc            #PDWSR_VAL
                sacl            PDWSR
                lacc            #IOWSR_VAL
                sacl            IOWSR
                ldp             #5
                lacc            #T200KHZ            ; program onboard
                sacl            TimerVal            ; timer
                out             TimerVal,TimerPt
                lacl            #CHAN5              ; select A/D channel
                or              #DAC0               ; select D/A channel
                sacl            LatchVal0
                out             LatchVal0,LatchPt
                in              Sample,ADPort       ; Clear A/D interrupt
                clrc            INTM                ; enable interrupts

GetSamp:                                            ; wait here
                b               GetSamp

HaveSamp:                                           ; ISR
                in              Sample,ADPort       ; read from A/D
                out             Sample,DAPort       ; write to D/A
                rete                                ; ret and reenable interrupts

                go                                  ; start program

                end                                 ; end of code
```

7. <u>**LINK PACKAGE**</u>

This section describes the utility functions for controlling the Model 5000 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 5000. The necessary Model 5000 information is in Chapter 4 of this manual. These programs are in the files M5000IO.OBJ and TCL5000.OBJ.

go320(BASEIO)

BASEIO - Base IO Address of Model 5000

Sets the Model 5000 into operation. The TMS320 RS pin is brought high.

hlt320(BASEIO)

BASEIO - Base IO Address of Model 5000

Resets the Model 5000. The TMS320 RS pin is brought low.

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 Model 5000 memory
BASEIO - Base IO Address of Model 5000
    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 the array X into Model 5000 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 Model 5000 memory
BASEIO - Base IO Address of Model 5000
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 5000 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  */
```

lockout(BASEIO)

BASEIO - Base IO Address of Model 5000

Sets the Lockout condition, allowing maximum rate data transfers between the TMS320 and Data RAM. The lockout condition is cleared by a PC access to Data RAM, as when calling sendio or recvio.

NOTE: See the FFTP500.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 and Multiplexer

Data

A/D	: Maxim 120 or equivalent
Multiplexer	: Harris 518 or equivalent
Maximum Rate	: 500 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 TMS320'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 is generated by the output of the onboard Timer when Jumper J7 is set to the default 'B' position. When Jumper J7 is set to the 'A' position, the strobe is generated by a TMS320 IN instruction at Port 2. See Section 4.A.

Step 2: The ADC announces that it has completed the conversion by generating an INT1 interrupt to the TMS320. Upon its reception the TMS320 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

Due to the architecture of the A/D converter, the multiplexer channel must be selected one sample in advance of the conversion. MUXTEST.ASM is an example program on your distribution diskette which continually scans all eight input channels.

B. D/A Converter

Data

Type	: AD DAC8222 or equivalent
Rate	: 250 Khz +
Output Voltage Range	: +/- 5 Volts

Operation

The D/A Converter is mapped as Write Port 3 in the TMS320'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 conversion strobe. The conversion strobe is the clock strobe from the programmable Onboard Timer when Jumper J5 is in the default 'D' position. When Jumper J5 is in the 'C' position, the conversion strobe comes from a Write to Port 2. 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 conversion strobe.

Desired Output Voltage	Value	D5000 Instruction
- 5 Volts	0	o0,3
.	.	.
0 Volts	800	o800,3
.	.	.
4.9976 Volts	FFF	oFFF,3

Unless otherwise noted, all Dalanco applications software uses Jumpers J5 and J7 shunted in the default positions.

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

9. FFT ROUTINES

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.

Example: a 'manual' FFT in D325.

```
-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 TMS320
-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 TMS320 is doing.

Example: FFT in a user's program.

If you program in C, see the FFTP500.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. <u>DIS5000 PROGRAM</u> - 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 500 Khz. This program is only useful with Model 5000 boards equipped with the on-board A/D converter.

1) Type **DIS5000 <cr>**
2) DIS5000 will ask you for the IO addressing of the Model 5000 board.

   <u>Example</u>

   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 DIS5000 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 oscilloscope.

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

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

To exit DIS5000, enter **Q**.

11. DATA TRANSFER

A. RECORD - Model 5000 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 can exceed 200 Khz on most IDE and SCSI based hard drive systems.
The RECORD program prompts the user for the following data:

1. IO addressing of the Model 5000.
2. Sample Rate
3. Disk file name (for writing)

B. PLAYBACK - PC to Model 5000 Transfer

This program transfers data from the PC's hard disk to the Model 5000 for output to the D/A converter. The program prompts for the following data:

1. IO addressing of the Model 5000.
2. Sample Rate
3. Disk file name (for reading)

You may connect the Model 5000 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 5000. The output from the D/A Channel 0 on the Model 5000 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. <u>EDIT50 Program</u>

This program enables the user to view captured data files on the Graphics computer screen. EDIT50 can scroll through a file, and create and import file sections. Additional features include file editing, cut and paste, and mixing functions.
Enter <?> for the command list.

D. <u>DELAY Program</u>

The DELAY program demonstrates how a digital delay may be implemented using the Model 5000. 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. <u>DACQ DATA ACQUISITION MANAGER</u>

The Data Acquisition Manager program is a multi-channel data acquisition manager employing a graphical user interface for ease of use. The Record, Playback, and Response features are incorporated and expanded upon in a single program. Various triggering options are available. Requires graphics card and mouse.

13. DIGITAL FILTER IMPLEMENTATION

One popular use for the TMS320 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.

The listing in the Applications book must be modified slightly to run on the Model 5000. The assembly language instructions must be changed from the TMS320C25 to the TMS320C5x format. The further 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.

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

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 5000 is 2. The output port (the D/A Converter) value is 3.

The modified program FIR5000.ASM is on your disk.

Examine FIR5000.ASM with your text editor, and run the program by typing

```
A5000 FIR5000<cr>
```

at the DOS prompt. You will need to modify the listing FIR5000.ASM if you are not using the default Base IO value.

GEN5000 CODE GENERATOR

GEN5000 is a digital filter code generator which produces FIR filter code for the Model 5000. In this way, filter responses may be viewed and studied minutes after their design.

GEN5000 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 GEN5000 to normalize the coefficient values. Thus, after specifying:

```
-3
12
17
12
-3
```

as your coefficient values, GEN5000 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 GEN5000 is a TMS320C5x source code file which may
be assembled and run with A5000. 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.

Example 1 : Interactive Session

```
>GEN5000 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, GEN5000 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

```
>GEN5000 <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. 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 GEN5000.

Reference

1. Programs for Digital Signal Processing, IEEE Press.

APPENDIX A

MODEL 5000 IO PORT ASSIGNMENTS

Port	Read	Write
0	Interrupt Ack	Program Timer
1	Interrupt PC	Latch Port
2	A/D Converter	D/A 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

EISA Configuration

To configure your EISA bus computer for use with the Model 5000, you may need to invoke the System Configuration Program provided with your computer.

After reading the information given in the program select the option which permits you to add or remove boards. Select the empty slot into which you will insert the Model 5000. Then select the board type, in this case "Generic ISA Adapter Definition" or "Board with no CFG file".

Select the option which permits you to View or Edit Details. You will find the "Generic ISA Adapter Definition" in the previously selected slot position.

Now provide the following information:

Under "ISA I/O Port Resource Allocation", specify "Range of 8 Port Addresses Required". The specify the actual port addresses, for example 300h-307h.

Under "IRQ Resource Allocation", specify "One IRQ Required" if you will be using this resource. Then specify the actual IRQ in use with the Model 5000, eg: "INT 10 Trigger Edge".

The Model 5000 requires no DMA or MEMORY resources on the part of the computer.

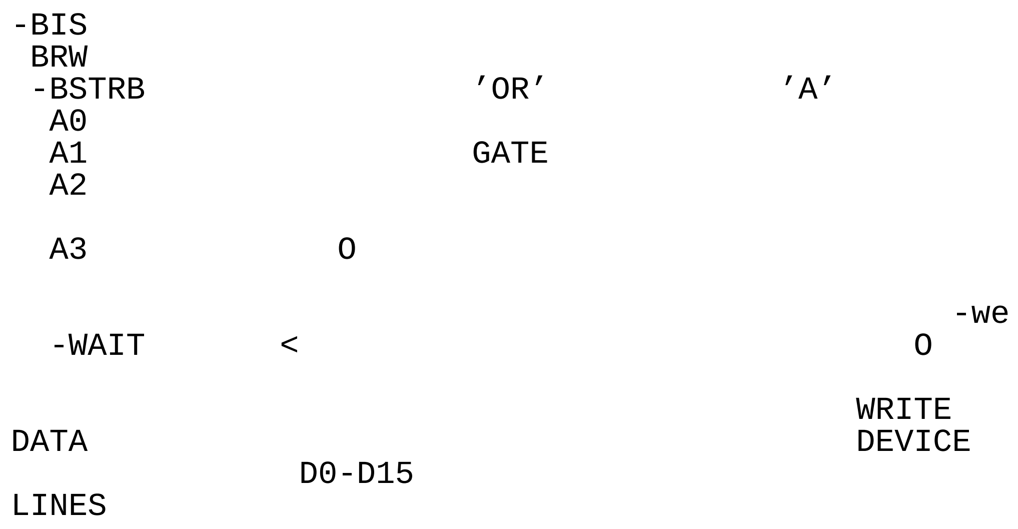
Save your configuration and exit.

APPENDIX C

Digital I/O Connector Usage

The digital I/O connector may be used to connect other devices to the Model 5000’s I/O bus. These devices typically reside at I/O port addresses 8-15.

This example demonstrates a fast write device residing at Port 8 in the TMS320’s I/O space.



If the device is a fast device device such as a latch, the -WAIT pin is wired directly to point ’A’. For slow devices, a wait state generator is inserted between the -WAIT pin and point ’A’.

In the case of multiple devices connected to the I/O connector multiple WAIT lines must be logically or’ed to form a single wait line to the -WAIT pin.

<u>APPENDIX D</u>

<u>Programming Guidelines for the Model 5000</u>

Correct operation of the Model 5000 requires that the software Wait State Registers be programmed.

Model 5000 resources require the following number of wait states:

Memory (Program and Data)	0 wait states
IO write to Ports 2 and 3	7 wait states
All other IO Ports	0 wait states

Here is the code segment for the correct initialization of the Wait State Registers:

```
PDWSR               equ 028h
IOWSR               equ 029h
CWSR                equ 02Ah
CWSR_VAL            equ 4               ; IO port 0-7
IOWSR_VAL           equ 0Ch             ; 7 wait states for port 2-3
PDWSR_VAL           equ 0               ; 0 s/w wait states for
                                        ; memory access


          b    Start
          . . . . . .

Start:
          ldp  #0
          . . . . . .

          lacc #CWSR_VAL      ; set wait states
          sacl CWSR
          lacc #PDWSR_VAL
          sacl PDWSR
          lacc #IOWSR_VAL
          sacl IOWSR
          . . . . . . .
```

REFERENCES

1. TMS320C5x User's Guide , Reference no. SPRUOO56B
2. TMS320 Development Support Reference Guide, Ref no. SPRU007A
3. Digital Signal Processing Applications with the TMS320 Family,
   Ref no. SPRA012