MODEL 310

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
(716) 271-8380
sales@dalanco.com
http://www.dalanco.com

TABLE OF CONTENTS

Table of Contents

1. INTRODUCTION

This manual is the reference to the Model 310 Digital Signal Processor Board and its accompanying software. The reference for the Texas Instruments TMS320C31 Digital Signal Processor and its assembly language is the TMS320C3x User's Guide available from TI.

Partial List of References

1. TMS320C3x User's Guide , Reference no. SPRUO31A
2. TMS320 Development Support Reference Guide, Ref no. SPRU007A
3. Digital Signal Processing Applications with the TMS320 Family, Ref no. SPRA017

2. INSTALLATION

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

B. Set the address jumpers for the IO mapping.

These jumpers determine the Base IO addressing at which the Model 310 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 J1 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 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 310 boards, each Model 310 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 | 0
The correct jumper configuration is

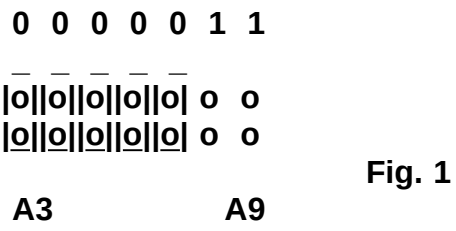


Fig. 1

Example for Base IO address 310h:

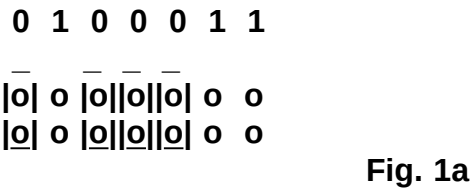


Fig. 1a

OTHER SETTINGS

C. System Memory

The Model 310 is fitted with memory ranging in size from 32K words to 512K words of static RAM. The memory resides in the four socket assemblies located to the left of the TMS320C31.

Each socket assembly (Fig. 2) is designed to accept bytewide SRAMS of varying configurations and physical size.

The memory ICs are placed such that the IC markings are rightside up. This is the same orientation as the neighboring ICs. The ICs are seated such that the IC bottoms are in the rightmost socket position.

Acceptable configurations are:

32K x 8: Jumper J2 connects center and LEFT post.
128K x 8: Jumper J2 connects center and LEFT post.
512K x 8: Jumper J2 connects center and RIGHT post.

Acceptable IC widths:

.6 " : Use the entire width of the assembly.
.4 " : Use the 2 lower socket strips in each assembly.
.3 " : Use the 2 lower socket strips in each assembly. The IC pins will have to be slightly bent in order to accomodate the 4." socket width.

For 0 wait state operation, very fast SRAMs need to be used.

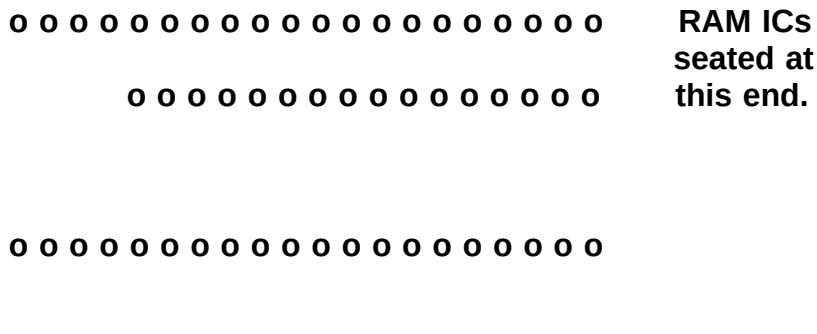


Fig. 2

D. Interrupt Jumper Settings

Jumper J000 is used to implement one of the following PC interrupts: INT10, INT11, INT12, or INT15.

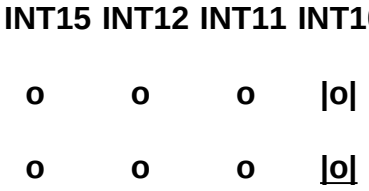


Fig. 3: J000 in INT10 position

Only one connection should be made.

JI3-I5 is located to the right of J000.

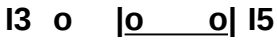


Fig. 4: JI3-I5 in INT5 position

JI3-I5 is jumpered if either INT3 or INT5 are desired. Jumpers J000 and JI3-I5 cannot both be jumpered. Only one of the connections may be made.

3. OPERATION EXAMPLE

The best way to become familiar with the Model 310 is to use the D300 debugger. We will use it to:

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

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

Invoke D300:
Type D300 at the MS-DOS prompt. A dash (-) appears as the new prompt.

Display memory:
To display the first 60h words of (0 to 5fh), type

-**d0,5f<cr>**

Note that the addressing points to 32 bit words, not 8 bit bytes.

Fill memory with a constant:
Enter the Fill command and D300 responds with three questions. In the example below, we are filling 1000h locations with zeroes.

-**f<cr>**
Enter constant?**0<cr>**
Starting Address?**0<cr>**
Number of Locations to fill (HEX)?**1000<cr>**

Write a short program:
To insert the Reset Vector at location 0, use the Substitute command. The Reset Vector is the address location of the first instruction in your program. Type

-**s0<cr>**

The address 0 and its current contents appear and D300 awaits your entry. Type

40

Note: If D300 responds with ???, you made an entry error. Reenter the instruction. D300 provides the next address (in this example, 1) 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 exit the Substitute command, type

.<cr>

The dash (-) prompt reappears.

We will continue our program at address 40h, so type

-**a40<cr>**

Then enter the rest of the program

40 **LDP 0**
41 **LDI 900H,SP**
42 **LDF @50H,R0**
43 **LDF @51H,R1**
44 **ADDF3 R0,R1,R2**
45 **STF R2,@52H**
46 **B 46H**
47 **. <cr>**

The instruction at location 41h sets the stack pointer. It is necessary if you later decide to trace through this new program using the Extended instructions.

Using the Substitute command, let's enter data at locations 50h and 51h.

-**s50<cr>**
0050 ABCDEF89-**0.75<cr>**
0051 C0000000-**1.05<cr>**
0052 12340987-**.<cr>**

Here we are entering the floating point number 0.75 at location 5Oh. Note that we entered **0.75**, not **.75**. The leading zero is necessary.

Disassemble the program:
The following command disassembles the statement at location 40h.

-**u40**

Continue to press <cr> to disassemble subsequent instructions. Correct any errors using the Assemble or Substitute commands.
To return to the prompt, enter **.<cr>**.

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:
SInce you have just performed a high speed floating point calculation, you will probably want to look at the data in its floating point format. To do this, enter a dot <.> after the Display or Substitute command.

-**d.50,52<cr>**

or

-**s.50<cr>**

Does location 52h contain the sum of locations 51h and 50h ?

4. HARDWARE FEATURES

A. Onboard Clocks

The TMS320C31 has two internal clocks , TCLK0 and TCLK1.
These are brought out at **Header J4**.

TCLK1 |o o| TCLK0

TCLK1 is not used by the Model 310A, while TCLK0 is used to trigger conversions on the A/D converter.

Jumper J40 connects TCLK0 to the A/D converter. It should be jumpered if analog IO is to be used.

Jumper J41 connects the CLKR0 of the serial port to the A/D converter. It should be jumpered if analog IO is to be used.

Jumper J42 connects the CLKIN of the A/D converter to CLKR0 ('A' position) or to the 5 Mhz oscillator ('B' position). The default is the 'B' position.

Fig ..

B. Memory Access

The Model 310 holds a single bank of of 32 bit wide memory.

From the TMS320 point of view, access to memory 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 64K Page value. The 64K Page value is bits a16-a19 of the desired address value. The 64K Page value is latched and need not be changed once it is set unless accesses are made across page boundaries.
To write the 64K Page value, use a byte write instruction to Base IO address + 6.

```
outbyte(page_port_value, page_value);
```

Assuming a Base IO address of 300h ,the page_port_value is 306h.

The next step is to set the address counter for the lower 16 bits on the Model 310 to the desired 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 a1 a0

where

a15 - a0 are the 16 LSBs of the address to be accessed.

Remember to use 16 bit word (not 8 bit byte) IO instructions when accessing the address counter or memory on the Model 310.

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:

```
outword16(port_value, address_value);
```

```
outbyte(306h,0)       ; set addr. to location C00h in memory
outword16(302h,C00h)

outbyte(306h,1)       ; set addr. to location 10C00h in memory
outword16(302h,C00h)
```

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.
Two reads (or writes) are needed because the data is 32 bits wide.

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

write:
outword16(port_value, data_value_lower_16_bits); then,
outword16(port_value, data_value_upper_16_bits);

read:

```
data_value_lower_16_bits = inword16(port_value);  then,
data_value_upper_16_bits = inword16(port_value);
```

Examples:

1) Assume address ctr. has been set to C00h.

```
outword16(300h,1234h) ; write 789A1234h to Data location C00h
   outword16(300h,789Ah);
```

2) Assume address ctr. has been set to C00h.

```
   ; read from location C00h
data_value_at_C00h = inword16(300h)'or'(inword16(300h)<<16)
```

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

Note:

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. In practice there is no need to shift the upper data by 16 bits when using these instructions. The data is properly placed in memory automatically. See the **sendio** and **recvio** functions in the Link Package (Chapter 7).

The PC may access the RAM when the TMS320C31 is in the RESET (not running) condition or the SET (running) condition.

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

An Example:

TMS320C31 writes the contents of register R7 to location C10h in memory:

```
sti r7,@0C10h
```

PC reads location C10h in memory (Model 310 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 lower
                    ; word16
in  ax,dx      ; read data value from RAM upper
                    : word16
```

or, in Turbo C:

```
long int data_value;
outport16(0x302,0xC10);
data_value = inport16(0x300) |
(inport16(0x300)<<16);
```

or, even better when transferring arrays of data, use the recvio() function in the Link Library.

```
recvio(&data_value,1,0xc10l,Baseio);
```

C. Start TMS32OC31 Operation

The host PC instruction to begin TMS32OC31 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 TMS32OC31 Operation

The host PC instruction to halt TMS32OC31 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 TMS32OC31

The host PC instruction to send an INT0 interrupt to the TMS32OC31 (to set the INT0 pin on the TMS32OC31 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 IN AL,DX
BASIC:	INP(&H3O5)
Debug:	i305
Turbo C:	inportb(0x305);

E.2 Model 310 Interrupt to the host PC's CPU

The Host CPU may be interrupted by the Model 310. In this way the TMS32OC31 may tell the PC that it has completed a computation or that it has some data to pass to the PC.

From the TMS320C31 point of view, the operation is quite simple. A positive pulse at the XF0 pin 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 PC 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 enable an interrupt, its corresponding bit is set to zero .

3) The Interrupt Driver on the Model 310 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).

TSR310-10.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 310. The interrupt may be generated by assembling TESTXF0.ASM and then running it from within D300. 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 TMS320C31 Code

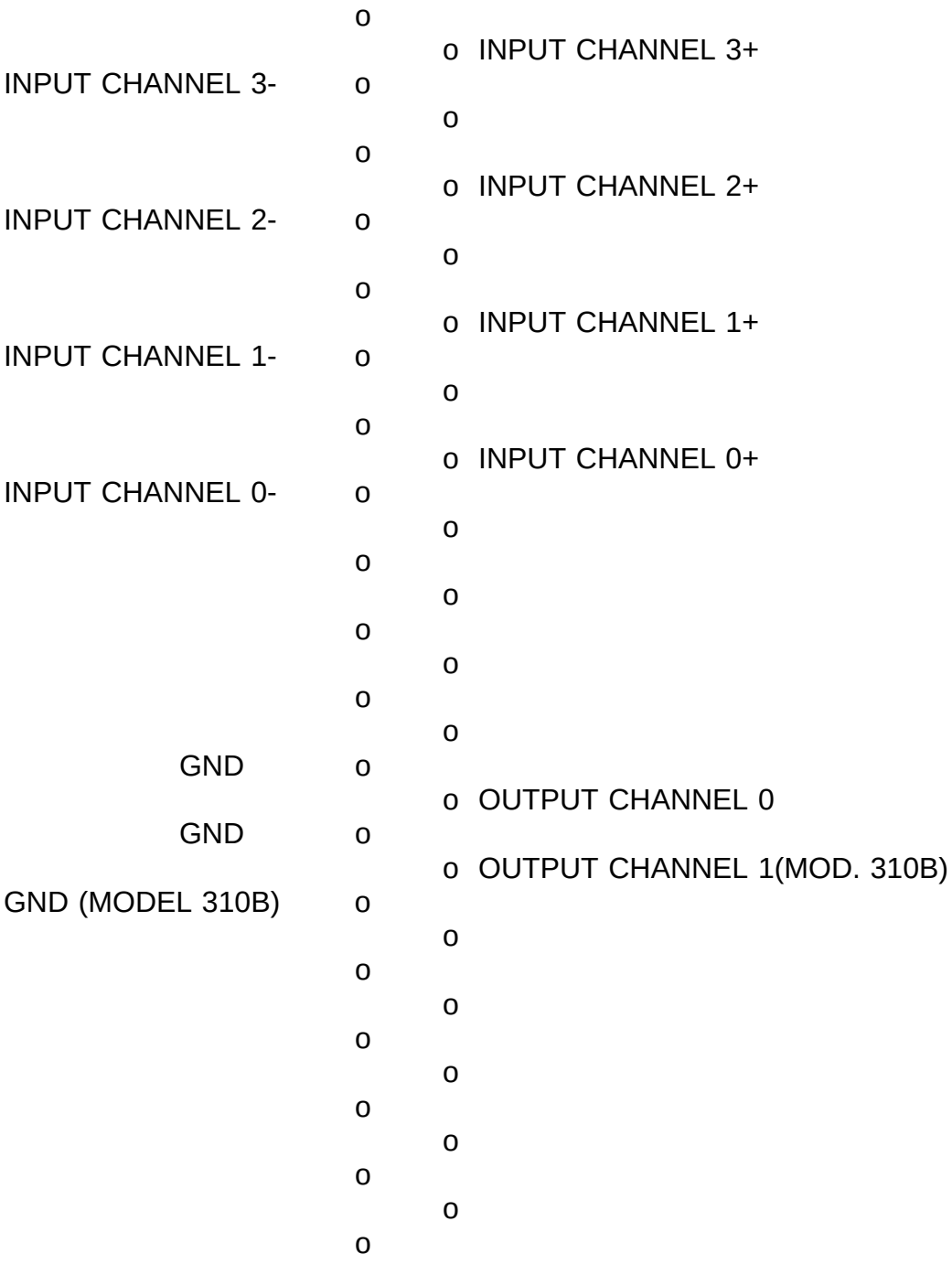
```
ldi 73,r0         ; send a '73' to location
sti r0,@0C00h ; C00h
DO FFT HERE
     .

ldi 88,r0         ; send an '88' to location
sti r0,@0C00h ; C00h
```

In this case the host would poll location C00H in memory 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. The IO Connectors

The external 37 PIN connector is configured as follows:



G. The Latch

The TMS320 controls the following Model 310 parameters by writing to a latch mapped at memory location 0FFFFFFh.

a. The next A/D input channel to be sampled (Chan 0-3).
b. The Gain setting of the Programmable Gain Amplifier.

Format:

bit3	bit2	bit1	bit0
gain1	gain0	muxa1	muxa0

where

Bits 0-1 select the Input Channel value, which ranges from 0 to 3.

Bits 2-3 select the Programmable Gain value. The actual gain will depend on the type of Programmable Gain Amplifier used on your Model 310.

bit3	bit2	GAIN PGA204/202	GAIN PGA205/203
0	0	1	1
0	1	10	2
1	0	100	4
1	1	1000	8

Example Using TMS320 Assembly language:

```
LATCH_VAL       .word  5        ; Channel =1, Gain =2 (forPGA203/5)
LATCH_AREA      .word  0FFFFFFH

        LDI     @LATCH_AREA,AR3
        LDI     @LATCH_VAL,R0
        STI     R0,*AR3
```

5. D300 DEBUGGER

D300, the debugger, is the basic tool for 'manually' controlling the Model 310 board.

Syntax: **>D300 { -BaseIO_Address }<cr>**

The '{ }' brackets indicate an optional parameter. The Base IO address may be set on the command line, and also set using the Log command. The default Base IO address is 300h.

The commands are explained below:

A - Assemble
The A command places TMS320C3x instructions into 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>**
B92 **sti r0,@100h<cr>**
B93 **rolc r2 <cr>** ;

Press <cr> to assemble the next instruction. D300 displays the assembled object code. Exit by entering a dot (.) followed by <cr>.

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 memory in hexadecimal or floating point notation. Syntax: Daddress1,address2 or D.address1,address2 where address2 is greater than address1.

-**d54,110<cr>**

displays all memory locations between 54h and 110h in hexadecimal format.

-**d.54,110<cr>**

displays all memory locations between 54h and 110h in floating point format.

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, as well as memory and register modification, during an active debugging session. The program under study must conform to the following standard format:

1. The statement at address 0 must be the reset vector to the beginning of the program, which itself must reside at address 40h or above.
2. A RAM area is reserved for use by the D300 breakpoint handler. This memory area may be specified by the user and must not conflict with the memory areas used by the user program.
3. The user program should specify a stack location in one of its first few instructions. The stack is shared by the user program and the breakpoint handler, and will grow by as many as 50 words. For example, if 1000h is chosen as the stack location, then the memory area 1000h-1030h should be reserved for stack use only.

For an example of a user program compatible with the D300 extended commands, trace through FIR300.ASM.

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.
S - Substitute values in RAM (onchip or off chip) and in the onchip Registers. A floating point number placed in R0-R7 will be placed in extended (40 bit) format. Change the value of the instruction pointer.
A - Assemble command.
U - Unassemble command.
F2 - Function key 2 is the Single Step key.
D - Select 5 memory areas (onchip or offchip) for display. A dot placed before the address indicates that the data will be displayed in floating point format.

Offchip RAM refers to the RAM (memory) ICs on the Model 310.
Onchip RAM refers to the RAM internal to the TMS320 DSP.

F - Fill

The F command fills a portion of memory with a 32 bit constant. The user is in turn prompted for the constant in hexadecimal or floating point notation, the first address at which the constant is to be placed, and the number of memory locations to be filled.

-**f** Enter constant in hex ? **11200 <cr>**
Starting Address in hex ? **BCD <cr>**
Number of Locations to Fill (hex) ? **300 <cr>**

This example fills the 300h locations in memory starting at BCDh with 11200h.

G - Go
The G command sets the TMS320 set/reset pin, thus setting the processor into operation.

-**g<cr>**

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

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

-**h<cr>**

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 310 boards from within a single session of the D300 program. You may also select an area in memory for the debug monitor used with the Extended Instructions if the default area is not suitable.

M - Move
The Move command moves data from one block of memory (the source) to another (the destination). D300 prompts the user for the start and end addresses of the source block and the beginning address of the destination block. The memory must be offchip, ie external to the TMS320.

Q - Quit
Used to exit the D300 program.

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

-**rb:crypto.bin<cr>**

reads the file b:crypto.bin into memory.

S - Substitute
The Substitute command allows the replacement of the 32 bit word at each memory location. The input may be in hexadecimal integer or floating point format. Floating point numbers are converted to the TMS320C3x floating point format.

-**s556<cr>**
556 111CA321-**8000<cr>**
557 A098AD37-**1.07e-3<cr>**
558 ffff0000-**0.987<cr>**
559 C8000000-**.<cr>**

In the above example the user has replaced the words at addresses 556h through 558h. Enter <cr> to modify the next memory location. Exit by entering a dot (.) followed by <cr>. In the case of fractional floating point numbers, a zero must be the leading character, ie enter **0.123**, not **.123**.

U - Unassemble
This command disassembles memory contents into their TMS320C3x 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 ROR R5 **<cr>**
6 NEGI 100,R5 **<cr>**
8 LDI *-AR1(IR0),R3 **. <cr>**

V - View
The View command continuously scans the desired range of offchip RAM. In this way one may examine areas of memory during Model 310 operation. Input syntax is similar to **Display**.

-**v400,41f<cr>** for hexadecimal display.

-**v.400,41f<cr>** for floating point format display.

W - Write to Disk
This command writes the contents of memory to disk.

-**wa:test<cr>**
Starting Location in memory (Hex) ?**10<cr>**
Number of Locations to write to file ?**12<cr>**

writes memory locations 10h to 22h to a:test.

X - Read TI Coff file from disk
This command reads the contents of a Texas Instruments format Coff file into the Model 310 Memory. This allows programs written with the Texas Instruments Assembler and C Compiler to be run directly on the Model 300. This command serves the same function as the LOAD300 program.

-**xb:fft.out<cr>**

6. A300 ASSEMBLER

A300 is the assembler for the Model 310.

The Command line

A300 {-output mode} {-base_address} infile

The output mode can be:
direct -- program is loaded to MODEL 310, no file produced.
a ascii -- ASCII format. This output may be used as an array in the user's C language programs.

The default output mode is direct.

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

Enter the base address if the Model 310 is not addressed at 300h.

eg: A300 -308 myfile

Instructions

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

Directives

.aorg <position pointer>

Changes the current position pointer to the one specified. This feature is not compatible with the TI Assembler, which relies on its Linker for placement of code and data segments in memory.
Example:

```
.aorg 10h
```

.space <number of memory locations>

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

```
        .space 100
```

.word <data word>

Stores the data word.
Example:

```
SinTab  .word    0
        .word    abcd324h
```

.float <floating point data word>

Stores the data word in floating point format. It is useful for storing sine tables, window and filter coefficients, etc..
Example:

```
        .float  1.070098e-1
        .float  0.98120                 ; NOT .98120 !
```

.**end**

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

```
        .end
```

.**set**

Equates a variable to an expression.
Example:

```
Here  .set  400h
```

.go

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

```
        .go
```

The following TI Linker related directives have no effect on the assembly process:
.data, .text, .global

Reserved Words

All of the TMS320 Opcodes and Assembler directives are Reserved Words. If they are used as labels or variable names, a "Syntax Error" will result.
Example:

```
End:    br      End     ; illegal because ".END" is a directive
End1:   br      End1    : OK
```

Differences betweeen the TI and Dalanco Assemblers.

A300 has the following requirements:

1. Labels must be followed by a colon ':'
example:

```
here:  nop
       br      @here
```

2. Comments must begin with a semicolon ';'

3. Three operand instructions must have a '3' in their opcode.
example:

```
       mpyf    r0,r4       ; 2 operand instruction
       mpyf3   r0,r4,r7    ; 3 operand instruction, OK
       mpyf    r0,r4,r7    ; SYNTAX ERROR
```

4. Source and destination operand must be explicitly written.
example:

```
        neg     r0,r0       ; OK
        neg     r0          ; SYNTAX ERROR
```

5. A Model 310 board must be present in the system in which A300 is run, even in the ascii output mode. A300 must also know the correct base IO address of the Model 310 board.
example:

To create an ascii file in a system where the Model 310 is addressed at 308h,

```
A300 -A -308 test
```

Example program

```
; May be assembled with TI or Dalanco Assemblers

; AD to DA Pass through example for use with Dalanco Spry Model 310

; Uses tclk0 as the ConvertStart signal.
; The tclk0 pin is programmed in IO mode, ie the ConvertStart signal
; is toggled by the DSP. See the other software examples to see how
; the tclk0 pin is programmed in clock mode.

; Uses the onboard 5 mhz oscillator as the serial port clock signal
; Uses xf1 to turn oscillator ON/OFF.

; Output to D/A converter at serial port 0
; Input data is also passed to location 100h, where it can be viewed ; using the D300
Debugger.
;
;
RESET                   .word  StartPt

        .data
IOF_SET_XF1             .word  62H          ; Word to make XF1 HI
IOF_RESET_XF1           .word  22H          ; Word to make XF1 LO
CTRL                    .word  808000H      ; Base addr. of onchip
                                            ; peripherals
TIMGB0CONHI             .word   6H          ; Word to make tclk0 HI
TIMGB0CONLO             .word  2H           ; Word to make tclk0 LO
SERGLOBA                .word  150144H      ; Ser. Pt. Global Control  Reg.
SERGLOB0                .word  0C150144H        ; Ser. Pt. Global Control Reg.
SERPRTX0                .word  111H         ; Ser. Pt. transmit pin particulars
SERPRTR0                .word  111H         ; Ser. Pt. receive pin particulars
SERTIM0                 .word  03CFh        ; Ser. Pt. timer control
SERTIM0VAL              .word  01h          ; Ser. Pt. timer period
DATA_TO_HOST            .word  100H         ; Memory loc.
LATCH_VAL               .word  0H           ; Value to write to Model 310 Latch
LATCH_AREA              .word  0FFFFFFH     ; Address of Latch in TMS320 memory

        .text
StartPt:
;

        LDP   0h,DP                 ; Set Data Page
        LDI   1800H,ST              ; Initialize status reg.
```

```
        LDI     0985H,SP                ; Initialize stack ptr



        LDI     @IOF_SET_XF1,R1         ; Turn oscillator on
        LDI     R1,IOF                  ; to enable serial port
                                        ; transmissions

        LDI     @CTRL,AR0
        LDI     @DATA_TO_HOST,AR1

        LDI     @LATCH_AREA,AR3         ; Set the A/D Channel
        LDI     @LATCH_VAL,R0           ; and Gain
        STI     R0,*+AR3(0)


        LDI     @SERGLOBA,R0    ; Program the Serial
        STI     R0,*+AR0(64)    ; Port and its
        LDI     @SERTIM0VAL,R0  ; Timer
        STI     R0,*+AR0(70)
        LDI     @SERTIM0 ,R0
        STI     R0,*+AR0(68)
        LDI     @SERPRTX0,R0
        STI     R0,*+AR0(66)
        LDI     @SERPRTR0,R0
        STI     R0,*+AR0(67)
        LDI     @SERGLOB0,R0
        STI     R0,*+AR0(64)

        LDI     @TIMGB0CONHI,R0         ; ConvertStart HI
        STI     R0,*+AR0(20H)

        LDI     18H,R0                  ; Set Wait States in TMS320 to ZERO
        STI     R0,*+AR0(64H)           ; (See Chap. 7 in TMS320C3x Guide)

next_sample:
;
;       start conversion
;
        LDI     @TIMGB0CONLO,R0         ; ConvertStart LO
        STI     R0,*+AR0(20H)
        NOP
        LDI     @TIMGB0CONHI,R0         ; ConvertStart HI
        STI     R0,*+AR0(20H)
```

```
wait_sample:                    ; Has data been received ?
        LDI   *+AR0(64),R2      ; Check status
        AND   01H,R2
        BZ    wait_sample ; Branch if not received


        LDI   *+AR0(76),R3      ; Read value from serial port
        STI   R3,*+AR1(0)       ; Send to memory for viewing
                                ; by host

        LSH   -20,R3            ; Convert to 12 bit format


        AND   0FFFH,R3          ; for D/A converter

try_send:                       ; Serial port ready to send ?
        LDI   *+AR0(64),R2      ; Check status
        AND   02H,R2
        BZ    try_send          ; Branch if not ready

        STI   R3,*+AR0(48H)     ; send to serial port
                                ; (the D/A converter)

        BR    @next_sample      ; Get next sample

              .end
```

7. TI COFF LOADER

Load300 is a program for loading Coff Executable files onto the Model 310. In this way, users of the TI Assembler and C compiler tools may easily run their programs on the Model 310.

Example:

```
>LOAD300 TIFFT
```

loads the contents of TIFFT.OUT into the correct locations on the Model 310.

Example:

```
>LOAD300 -A TIFFT
```

reads the contents of TIFFT.OUT and produces TIFFT.ASC for inclusion as array(s) in the user's C program.

8. LINK PACKAGE

This section describes the utility functions for controlling the Model 310 that are linkable to the user's software. These routines run on the host PC and assume the user's use of the 80x86 Large memory model. If another model is used, these routines may be modified for the desired memory model. It should be noted that the user can also write his own functions for controlling the Model 310. The necessary Model 310 information is in Chapter 4 of this manual. These functions are also available in a Windows 3.1 compatible DLL file. See Appendix E, Windows 3.1 Drivers and Visual Basic Examples.

int320(BASEIO)

BASEIO - Base IO Address of Model 310

Sends an INT0 interrupt to the TMS320C31 on the Model 310

go320(BASEIO)

BASEIO - Base IO Address of Model 310

Sets the TMS320C31 into operation.

hlt320(BASEIO)

BASEIO - Base IO Address of Model 310

Resets the TMS320C31, halting its operation.

sendio(long * X, unsigned LENGTH, long START, unsigned BASEIO)

X - Array of 32 bit words
LENGTH - Number of words in array X to send
START - Source start address in Model 310 memory
BASEIO - Base IO Address of Model 310

Copies the array X into Model 310 memory starting at memory location START.

Example
C Language:

```
sendio(pgm1, 0x88, 0l, baseio);
/* copy 88h words from array pgm1 into Model 310 memory starting at
location 0 */
```

recvio(long * X, unsigned LENGTH, long START, unsigned BASEIO)

X - Array of 32 bit words
LENGTH - Number of words in array X to peek
START - Source start address in Model 310 memory
BASEIO - Base IO Address of Model 310

Copies LENGTH words of Model 310 memory starting at START into array X.

Example
C Language:

```
recvio(&x,0x300,0x1000l,baseio);
/* copy 300h words from memory location 1000h into array x */
```

NOTE: See the FFT300.C and other listings on the diskette for examples of the use of the Link Package functions.

9. A/D CONVERTER and D/A CONVERTER

A. A/D Converter

Data

Type : Maxim 121 or equivalent
Maximum Rate : 150 KHz (310A), 300 KHz (310B)
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 A/D converter is connected to the three receive lines of the serial port of the TMS320C31.

First, **STEP 1**, the ADC must be told to begin the conversion. This is done by bringing the Convert pin low. This strobe is generated by the output of the onchip clock TCLK0.

Step 2: After conversion, the data is sent to DSP via the serial port. The DSP can poll the Serial Port Global Control Register to determine if the data has been received. Upon reception, the data may be read at the Serial Port Data Receive Register.

TCLK0 Particulars

The TCLK0 pin of the TMS320C31 may be programmed in one of two ways:

a. As an I/O pin. The TMS320C31 brings the TCLK0 signal (and thus the CONVERT signal) low, then high. The conversion is explicitly triggered by the DSP. See ADDA300.ASM or the listing in Section 6.

b. As a clock signal. TCLK0 is programmed with the desired A/D sampling rate and is left alone until a new sampling rate is desired. See ADDA301.ASM on the distrbution diskette.

The data format is 16 bit, two's complement binary code. The 2 LSBs of the 16 bit word are set to zero as the MAX121 is a 14 bit A/D converter.

B. D/A Converter

Data
Type : 1 AD7243 (310A), 2 AD7243 (310B)
Rate : 250 Khz single / 140 Khz dual channel
Output Voltage Range : +/- 5 Volts

Operation - Model 310A
The D/A Converter is connected to the three transmit lines of the serial port. The DSP polls the Serial Port Global Control Register to determine if the serial port transmit buffer is ready for a new data word. Upon notification of readiness, the new data word may be written to the Serial Port Data Transmit Register. The D/A output will be updated with the value in the lower 16 bits of the 32 bit data word after reception of the data word.

Operation - Model 310B Single Channel Output
This "Model 310A compatible" mode is set by setting jumpers **J44** and **J45** to the **'B'** position. The D/A Channel 0 Output will be updated with the value in the lower 16 bits of the 32 bit data word after reception of the data word.

Operation - Model 310B Dual Channel Output
This mode is set by setting jumpers **J44** and **J45** to the **'A'** position. The D/A Channel 0 Output will be updated with the value in the lower 16 bits of the 32 bit data word. The D/A Channel 1 Output will be updated with the value in the upper 16 bits of the 32 bit data word.
The updates of the D/A Channel outputs occur when the TCLK1 is pulsed low. See **ADDA302.ASM** example on the distribution diskette.

The data format of the 16 bit upper and lower words is two's complement binary code placed in the 12 least significant bits.

Desired Output Voltage	Value
- 5 Volts	800h
.	.
0 Volts	0
.	.
4.9976 Volts	7FFh

Unless otherwise noted, all Dalanco applications software uses Jumpers J40, J41 and J42 shunted in the default positions.

10. FFT ROUTINES

The FFT routine provided was downloaded from the TI bulletin board and may also be found in the TMS320C3x User's Guide:

FFT.ASM : The above FFT routine but modified for use with the A300 Assembler. There is no link step when using the Dalanco tools. Therefore the sine table is included as part of FFT.ASM.

In each case the **real** values are loaded in at **1400H** and the **imaginary** values are loaded in at **1500H**. Thus locations 1400H through 15FFH are reserved for input.

The results of the FFT replace the input.

Example: a 'manual' FFT in D300.

```
-g <cr>          ; Start TMS320
-h               ; Stop
-d1400,14ff <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 FFT300.C file on the distribution diskette. This program is also on your disk in executable form. A Color, EGA, or VGA Adapter is required.

11. DISPLAY PROGRAM - Time and Frequency Domain Display

Displays input signals on the VGA Graphics Adapter. This program enables the PC screen to function as a time domain display and as a frequency spectrum domain display with sample rates ranging to 250 KHz. This program requires a mouse.

1) Type **DISPLAY <cr>**

2) DISPLAY will ask you for the IO addressing of the Model 310 board.

Example

IO Address Base ? **300 <cr>**

Note that the answers are hexadecimal numbers.

3) Connect signal sources to the INPUT channels of interest. Make sure the **signals do not exceed +/- 5 Volts.**

4) The signals in their time and frequency domain representations should appear on the screen. There should be 8 signal windows (4 time domain + 4 frequency domain) on the screen.

The mouse buttons next to the signal windows are for modifying the acquisitiion sample rates associated with each window.

To exit DISPLAY, place the mouse cursor on the EXIT button and click the left button, or press CTRL-BREAK.

12. DATA TRANSFER

A. RECORD - Model 310 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 150 Khz are typical on 386 computers with IDE drives.

The RECORD program prompts the user for the following data:

1. IO addressing of the Model 310.
2. Sample Rate
3. Disk file name (for writing)

B. PLAYBACK - PC to Model 310 Transfer

This program transfers data from the PC's hard disk to the Model 310 for output to the D/A converter. The program prompts for the following data:

1. IO addressing of the Model 310.
2. Sample Rate
3. Disk file name (for reading)

13. MODA - MANAGER OF DATA ACQUISITION

This version of the MODA Program provides a graphical user interface and multi-channel recording & playback.

First enter the Board Setup and Acquisition information, confirm the setup by pressing the SETUP OK button, and then move to the Action menu.

If the Action window gets lost behind the Setup window, the windows may be moved by pressing the right mouse button. Or the Setup window may be removed by double clicking as in Microsoft Windows.

File Format
Our other file related products (RECORD, PLAYBACK) make use of single channel files only. MODA allows the use of multi-channel files. To allow compatibility between the various Dalanco Spry boards (Model 250, 500, 310), the data format is 12 bit offset binary with the Channel Number placed in the 4 MSBs of a 16 bit data word. Thus the 2 LSBs are lost in the case of the Model 310 with a 14 bit A/D Converter. The 12 LSBs are the data itself.

On playback, a file must contain the same (or fewer) channels of data as there are on the board. There is only one output channel on the Model 310A.

Rate
The sample rate required is the total rate. Thus if you are sampling 3 channels at 10 Khz each, enter 30 Khz as the sampling rate. The hardware is capable of greater flexibility (much larger choice of sample rates, different channels at different rates, separate record & playback rates, etc...) although the software implementation is not trivial in every case.

Not yet Implemented
Algorithmic triggering
FileView

14. 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 FIR300.ASM listing on the distribution diskette. It takes input data from the A/D converter, calls the FIR subroutine to calculate the filter output, and outputs the result to the D/A converter. The FIR subroutine is taken from the TMS320C3x User's Guide. It is a short (6 word), powerful subroutine. Programmers will note that the FIR subroutine demonstrates the use of the repeat instruction, parallel instructions, and circular addressing.

Examine FIR300.ASM with your text editor, and run the program by typing

A300 FIR300<cr> or

A300 -<Baseio_address> FIR300<cr>

at the DOS prompt.

The attributes which differentiate one FIR filter from another are the filter coefficients, and the sampling rate.

GEN300 CODE GENERATOR

GEN300 is a digital filter code generator which produces FIR filter code for the Model 310. It produces a listing similar to FIR300.ASM with the coefficients and sample rate that you have selected. In this way, filter responses may be viewed and studied minutes after their design.

GEN300 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.
rate	: followed by the sampling rate, in Hz.
end	: end of filter specification.

One of the following is selected.
normalize_OFF or normalize_OFF

The default is
normalize_OFF : OFF.

Normalize_ON would be used in the case, for example,in a smoothing formula where you wanted GEN300 to normalize the coefficient values. Thus, after specifying:

-3
12
17
12
-3

as your coefficient values, GEN300 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**.

The Output of GEN300 is a TMS320C3x source code file which may be assembled and run with A300.

Example 1 : Interactive Session at the Console

```
>GEN300 OutCodeFile.asm
 rate
 10000.                          ; rate = 10000 Hz
 coefficients
 -3/35
 12/35
 17/35
 12/35
 -3/35
 normalize_OFF
 end
```

If a heading is entered incorrectly, GEN300 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

```
>GEN300 <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
        .
        .



        .
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).

After using GEN300, assemble the program and run.

```
>A300 OutCodeFile
>D300
-g              !GO
```

You may change the coefficients while the filter is running. You may use the floating point format entry feature of the Substitute Command to do this.

In many filter design programs the coefficient list is produced as an output file. This list 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 GEN300.

Reference

1. Programs for Digital Signal Processing, IEEE Press.

APPENDIX A

HOST PC IO ASSIGNMENTS

Port value = IO Base + Offset

Offset	Read	Write
0	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)	Address Page (b)
7	Reset TMS320 (b)	

APPENDIX B

EISA Configuration

To configure your EISA bus computer for use with the Model 310, 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 310. 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 310, eg: "INT 10 Trigger Edge".

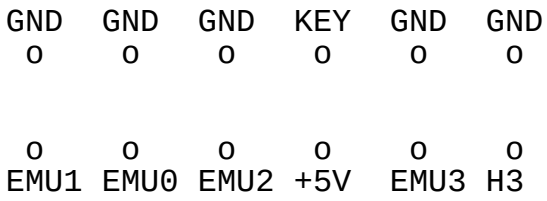
The Model 310 requires no DMA or MEMORY resources on the part of the computer.

Save your configuration and exit.

APPENDIX C

Emulation Header

J11 is the emulation header, where the Model 310 is used as the target system.



APPENDIX D

Texas Instruments C Compiler

The Texas Instruments Optimizing C Compiler for the TMS320C3X and TMS320C40 may be used with the Model 310. The advantages of using the C compiler are clear:

Rapid creation of applications, as well as easier maintenance of code.
Existing algorithms coded in C for other platforms may be recompiled to run on the TMS320C3X or TMS320C40.

The C-COMPILER subdirectory of the distribution diskette contains the examples and batch files needed for a quick start in using the C compiler.

CTEST.C is a test C program.
DOCL30.BAT is the batch file for compiling and linking, and for loading the program to the Model 310. Modify to conform to your own system setup.
CTEST.CMD is the Linker command file called by DOCL30.BAT. It specifies the memory map of the Model 310. Note the allocation of a small 'window' of memory for variables shared by the PC and TMS320C31.

Compile and link the program

```
DOCL30 CTEST
```

then run D300 and trace through the program.

If you don't have the C compiler but would like to trace through the code created by the C compiler, type

```
LOAD300 CTEST
```

and then use D300.

APPENDIX E

Windows 3.1 Drivers and Visual Basic Examples

M310WIN.DLL contains the same functions as the Link Package described in Chapter 8.

An example of its use with Visual Basic may be found on the diskette. Copy VBRUN300.DLL to your hard disk if you do not have Visual Basic. Windows 3.1 is required to run these examples.

REFERENCES

1. TMS320C3x User's Guide , Reference no. SPRUOO31A
2. TMS320 Development Support Reference Guide, Ref no. SPRU007A
3. Digital Signal Processing Applications with the TMS320 Family, Ref no. SPRA017