MODEL C32-104

SIGNAL PROCESSING BOARD

FOR THE PC/104 BUS AND STANDALONE

APPLICATIONS

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

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

TABLE OF CONTENTS

1. INTRODUCTION

This manual is the reference to the Model C32-104 Digital Signal Processor Board and its accompanying software. The reference for the Texas Instruments TMS320C32 Digital Signal Processor and its assembly language are the TMS320C3x User's Guide and the TMS320C32 User's Guide available from TI.

Partial List of References

1. TMS320C3x User's Guide , Reference no. SPRUO31A
2. TMS320C32 User's Guide , Reference no. SPRU132B
3. Digital Signal Processing Applications with the TMS320 Family, Ref no. SPRA017

The Model C32-104 may be operated in one of three environments:

Mode 1 - Slave Mode
This mode is used in a 16 bit PC/104 bus system or in a 16 bit ISA slot with the use of a PC/104 to ISA adapter. In this mode the bus master is a 386-686 or Pentium Host CPU on a PC/104 card (or on the motherboard in ISA systems). The Model C32-104 in this mode is a peripheral or 'slave' to the Host CPU which is accessed for processing and data acquisition tasks. This is the most common environment for program development, for programming the Flash memory, and for many applications.

Mode 2 - Standalone Mode
In this mode the Model C32-104 operates independently of the Host CPU, if there is one present. The Flash memory will have been programmed in advance in order for the Model C32-104 to operate in this mode. The Model C32-104 does not affect and is not affected by activity on the PC/104 bus. A Model C32-104 operating as a standalone computer with no other cards uses this mode.

Mode 3 - Limited Bus Master Mode
In this mode there is NO HOST CPU card present. This mode cannot be used with an ISA motherboard, which has a Host CPU. The Model C32-104 in this mode controls other boards on the PC/104 stack, acting as the master of these boards. With the proper software, the Model C32-104 may directly control such PC peripherals as serial ports. System cost may be appreciably reduced by allowing the Model C32-104 to take care of some of the 'housekeeping' functions in place of the Host CPU.

2. INSTALLATION

A. Make a backup working copy of the software.
Please see your DOS or Windows documentation if you have any questions on the copying of disk files.

B. Set the address jumpers for the IO mapping.

These jumpers determine the Base IO addressing at which the Model C32-104 resides in the host CPU's IO address space. (The host CPU is the master CPU in the PC/104 system, usually an Intel x86 or Pentium compatible processor). This address should not conflict with existing functions or adapters in the PC or PC/104 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 5 jumpers are read bottom to top (least significant bit to most significant), and determine the 8 byte block of IO addresses used to control the board. Only bits 5 thru 9 of the address are selectable. The upper bits (bits 10-11) are forced to represent a 0 while bits 0-4 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 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 32 byte boundaries.

Note that in systems with multiple Model C32-104 boards, each Model C32-104 must be addressed at its own unique Base IO address.

Examples
For a Base IO address of 300h, bits 9 thru 5 are 11000. For a Base IO address of 320h, bits 9 thru 5 are 11001.

Fig.2-1. Base IO at 300h Fig.2-2. Base IO at 320h

C. CPU Jumper

J6, the CPU jumper, is located just above J7. It is jumpered when there is a Host CPU present in the system. Thus it is jumpered in Slave Mode.

Slave Mode	Jumper ON
Standalone Mode	Jumper OFF
Limited Bus Master	Jumper OFF

D. MASTER Jumper

J11, the MASTER jumper, is located at the lower center of the board. It is jumpered when the Model C32-104 is a Limited Bus Master. Thus it is NOT jumpered in Slave Mode.

Slave Mode	Jumper OFF
Standalone Mode	Jumper OFF
Limited Bus Master	Jumper ON

E. Interrupt Jumper Settings

J10 is located at the lower center of the board. In Slave Mode, it is generally used to implement one of the following PC interrupts:

IRQ10, IRQ12, IRQ14, IRQ15, IRQ3, IRQ4, IRQ6, IRQ7

The particular IRQ used must not be used by other system resources. This generally favors the use of IRQ10 and above.

These interrupts may be used to interrupt the Host CPU from the Model C32-104.

The TMS320C32 on the Model C32-104 may also be interrupted by the Host CPU. See Section 4.G.2.

Connect the pin labelled with the desired interrupt number to the pin labelled 'INTPC' in fig. 2-3.
Only one connection should be made. A shunt may be placed between IRQ10 and IntPC to implement Interrupt 10. For the other interrupts, connections should be made with wire wrap wire.

The remaining pins on J10 are not used in Slave Mode.

F. Summary of Slave Mode Settings

J7 - Set for desired I/O Address
J6 - JUMPER ON
J11 - JUMPER OFF
J10 - Set for desired interrupt, if any

Figure 2-3. Interrupt options for Interrupt to Host CPU under Slave Mode.

3. OPERATION EXAMPLE

The best way to become familiar with the Model C32-104 is to use the D300 debugger. To use D300 and other PC based programs with the Model C32-104, set the Model C32-104 to Slave Mode.

We will use D300 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 6Oh words of (0 to 5fh), type

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

Note that the addressing always points to 32 bit words.

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. SRAM Memory Access

The Model C32-104 holds a single block of SRAM (static RAM) memory which starts at memory location 0. In most configurations there are 64K 32 bit words (256 KBytes) of SRAM.

All discussion of memory addressing will refer to 32 bit words.

From the TMS320C32 point of view, access to the SRAM is quite simple and is a natural part of the TMS320 instruction set.
From the Host CPU point of view, SRAM 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 16 through 19 (denoted as 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. In configurations with the standard 64K of SRAM there will only be one page, and in this case the page value should be set to 0.

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 lower 16 bits of the desired address value on the Model C32-104 by writing to the address counter. 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.

Use 2 8 bit byte IO instructions when writing to the address counter on the Model C32-104 from the Host. Low byte first (a7-a0), then high byte (a15-a8).

Use 16 bit word (not 8 bit byte) IO instructions when accessing (reading from and writing to) SRAM on the Model C32-104 from the Host.

Examples:
These assume a Base IO address of 300h. Therefore the port_value for the page value is 306h and the port value for a15-a0 of the address is 302h. Our pseudo-language IO instructions are of the form:

```
outbyte(port_value, address_value);
```

```
; set addr. to location C00h in memory
outbyte(306h,0)      ; Page value is 0
outbyte(302h,0)      ; Low byte of C00h
outbyte(302h,0Ch)  ; High byte of C00h

; set addr. to location 10C02h in memory
outbyte(306h,1)      ; Page value is 1
outbyte(302h,2)      ; Low byte of C02h
outbyte(302h,0Ch)  ; High byte of C02h
```

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.

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  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 SRAM 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 ISA compatible PC/104 bus, particularly when using the 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 source code for these functions is on the distribution diskette.

The PC may access the SRAM when the TMS320C32 is in the RESET (not running) condition or the SET (running) condition.

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

CAUTION: There is one exception to this rule. In Slave Mode, the PC may not access SRAM or other board resources when the TMS320C32 is running at reduced clock speed following the LOPOWER instruction. Doing so will disrupt any ongoing program and corrupt memory.

An Example:

TMS320C32 writes the contents of register R7 to location C10h in SRAM:

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

PC reads location C10h in SRAM (Model C32-104 mapped at 300h):

```
mov dx,306h         ; set Page = 0
mov al,0
out dx,al
mov dx,302h         ; set address value
mov ax,0C10h
out dx,al           ; low Byte
mov al,ah
out dx,al           ; high Byte
mov dx,300h         ;
in  ax,dx           ; read data value from RAM lower
                    ; word16
mov ,ax             ;
in  ax,dx           ; read data value from RAM upper
                    : word16
```

or, in Turbo C:

```
long int data_value;
outportb(0x306,0);
outportb(0x302,0x10);
outportb(0x302,0xC);
data_value = inport(0x300) |
                (inport(0x300)<<16);
```

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

```
recvio(&data_value,1,0xc10L,Baseio);
```

B. Flash Memory Access

Flash memory appears in the memory space of the TMS320C32 at 90000h and continues for 512K Bytes. The Flash memory data bus is 8 bits (1 byte) wide and is wired to the low byte of the TMS320C32's 32 bit data bus (d0-d7). The first byte of Flash appears at 90000h, the second byte at 90001h, etc.. Bits 8-31 appear 'empty' to the TMS320C32 in this memory region.

To access Flash memory from the Host CPU, a simple address translation is first performed. The address is shifted right by one bit.

Address as seen from Host = Address on Flash >> 1

Then this new address (we will call it 'Translated Addr') is presented from the Host in a manner similar to the procedure for accessing the SRAM.

outbyte(page_port_value, page_value of 'Translated Addr');
outbyte(address_counter, low byte value of 'Translated Addr');
outbyte(address_counter, high byte value of 'Translated Addr');

The Flash memory is then accessed with byte I/O to port [Baseio+3].
The address counter does not auto-increment for Flash memory accesses.

Reading from Flash memory is similar to reading from SRAM. Other operations (writing, erasing, etc) to Flash memory require special byte sequences which are described in detail in Reference 5.

C. Start TMS320C32 Operation

The host PC instruction to begin TMS320C32 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

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

D. Halt TMS320C32 Operation

The host PC instruction to halt TMS320C32 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

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

E. Set TMS320C32 to Microprocessor Mode

The host PC instruction to set the MCBL/MP pin on the TMS320C32 to LOW is a byte output at (Base IO address + 7). This is the default configuration on the Model C32-104 in Slave Mode. This example shows a system with the Base IO address at 300h.

```
Assembly Language:    MOV DX,307h
                      OUT DX,AL

Turbo C:              outportb(0x307,0);
```

F. Set TMS320C32 to Boot Loader Mode

The host PC instruction to set the MCBL/MP pin on the TMS320C32 to HIGH for a subsequent Boot Loader operation is a byte input at (Base IO address + 2). This example shows a system with the Base IO address at 300h.

```
Assembly Language:    MOV DX,302h
                      IN AL,DX

BASIC:                INP(&H302)
Debug:                i302
```

Turbo C: inportb(0x302);

G. Interrupts

G.1 The host toggles the INT0 pin on the TMS320C32

The host PC instruction to send an INT0 interrupt to the TMS320C32 (to set the INT0 pin on the TMS320C32 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);

G.2 Model C32-104 Interrupt to the host PC's CPU

The Host CPU may be interrupted by the Model C32-104. In this way the TMS320C32 may tell the PC that it has completed a computation or that it has some data to pass to the PC.

From the TMS320C32 point of view, the operation is quite simple. A write to address 818001h will, depending on the setting of jumpers on J10, send one of the following hardware interrupts to the host PC.

INT10
INT12
INT14
INT15

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 C32-104 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).

TSR_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 C32-104. The interrupt may be generated by assembling TINTPC.ASM and then running it from within D300. Make sure that J10 is jumpered in the 'INT10' position. On some computers it may be necessary to modify the IRQ settings in the system BIOS to ensure that IRQ10 is available on the ISA bus.

G.3 Polling

In this practice, the Host PC polls the memory location in the Model C32-104's SRAM and takes action (or keeps on polling) according to the value that it receives.

Example TMS320C32 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.

H. Digital IO Connector

J8 is configured as follows:

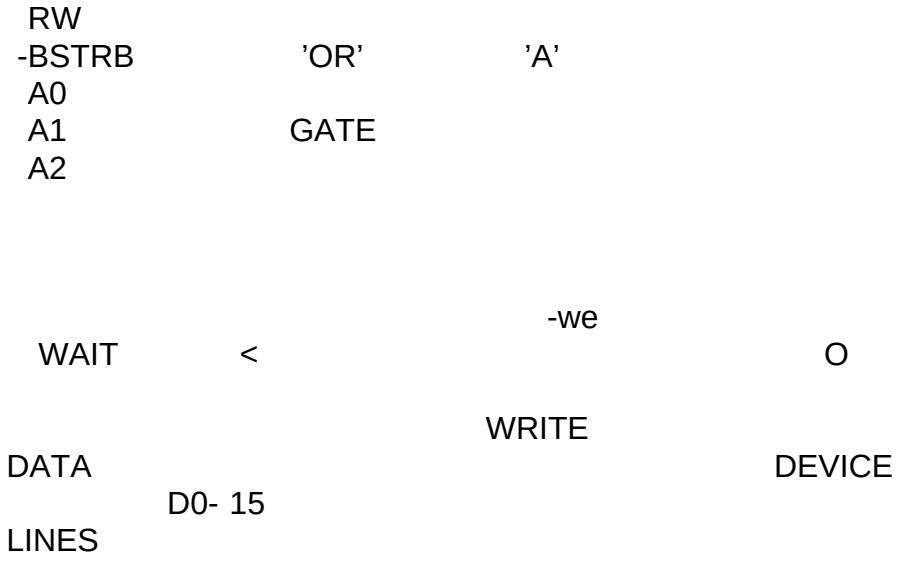
```
-DIOSTRB o o BH3
   TCLK1 o o INT1
      A2 o o A1
      A0 o o -BSTRB
      RW o o TCLK0
    INT3 o o WAIT
      D6 o o D7
      D4 o o D5
      D2 o o D3
      D0 o o D1
     D14 o o D15
     D12 o o D13
     D10 o o D11
      D8 o o D9   -- Pin 1
```

(O) means that the signal is an OUTPUT from the Model C32-104.
(I) means that the signal travels from the external module INTO the Model C32-104.
(IO) means that the signal is bi-directional.

D0-D15 (IO) are bits 0-15 of the TMS320C32's buffered data bus.
A0-A2 (O) are the TMS320C32 buffered address lines 0-2.
INT1 (I) is the active low INT1 interrupt to the Model C32-104.
INT3 (I) is the active low INT3 interrupt to the Model C32-104.
TCLK0 (O) is the output of the TCLK0 pin of the TMS320C32.
TCLK1 (IO) is the TCLK1 pin of the TMS320C32.
-BSTRB (O) is the buffered TMS320 -IOSTRB Control Signal.
-DIOSTRB (O) is -IOSTRB active during Digital IO accesses only.
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.
BH3 (O) is the buffered H3 clock signal from the TMS320C32.
BRW (O) is buffered TMS320 RW Control Signal. Read active high. Write active low.

H.1 Digital IO Hardware Connection

This example demonstrates a fast write device accessed at address 810000h in the TMS320C32's memory space.



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

In the case of multiple devices connected to the IO connector, multiple WAIT lines must be logically or'ed to form a single wait line to the WAIT pin.

H.2 Digital IO Software

The address range for accessing the digital IO connector is 810000h - 810007h. Address lines A0-A2 on the DIO connector correspond to the lowest 3 bits of the address. This permits the decoding of 8 read and 8 write addresses.

Example 1. Read from 810000h

A0-A3 at the DIO connector are LOW

```
DIO             .word   810000h
RAM             .word  100h

        LDI     @DIO,AR3
        LDI     @RAM,AR4
        . . . . . .
        LDI     *AR3,R1
        STI     R1,*AR4
        . . . . . .
```

Example 2. Write to 810007h

A0-A2 at the DIO connector are HIGH

```
DIO             .word   810000h
RAM             .word  100h

        LDI     @DIO,AR3
        LDI     @RAM,AR4
        . . . . . .
        LDI     *AR4,R1
        STI     R1,*+AR3(7)
        . . . . . .
```

Figure 4-1. Location of Serial and Digital IO Port.

I. Serial Port

The buffered serial port of the TMS320C32 is available at J9.

```
CLKR o o DR
  DX o o FSX
 FSR o o CLKX -- Pin 1
```

The pins are buffered in the following configuration:

CLKR	INPUT
FSR	INPUT
DR	INPUT
CLKX	OUTPUT
FSX	OUTPUT
DX	OUTPUT

If necessary, changes to this configuration may be made by cutting and redirecting traces at U18.

Test configuration
At J9, connect the following:
CLKR to CLKX
FSR to FSX
DR to DX

Run SERTEST.ASM on the distribution diskette.

J. Clock/Timers

There are two clock/timer pins on the TMS320C32. They are TCLK0 and TCLK1.

TCLK0 is available as a clock (OUTPUT only) on the Digital IO connector J8. See section 4.H.

TCLK1 is available as an input or output on the Digital IO connector. The direction is controlled by the XF0 pin on the TMS320C32.

When XF0 = 0, TCLK1 is an INPUT (timer)
When XF0 = 1, TCLK1 is an OUTPUT (clock)

Test files for the TCLK0 and TCLK1 are TT0.ASM and TT1.ASM:

5. D300 DEBUGGER

D300, the debugger, is the basic tool for 'manually' controlling the Model C32-104 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>.

B - Boot Loader Mode
Places the TMS320C32 on the Model C32-104 in Boot Loader Mode
by setting the MCBL/MP pin HIGH.

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 ETEST.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 C32-104.
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 C32-104 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.

MP - Microprocessor Mode
Places the TMS320C32 on the Model C32-104 in Microprocessor Mode
by setting the MCBL/MP pin LOW.

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 C32-104 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 C32-104 Memory. This allows programs written with the Texas Instruments Assembler and C Compiler to be run directly on the Model C32-104. This command serves the same function as the LOAD300 program.

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

6. A300 ASSEMBLER

A300 is the assembler for the Model C32-104.

The Command line

A300 {-output mode} {-base_address} infile

The output mode can be:

direct -- program is loaded to MODEL C32-104, no file produced.
a ascii -- ASCII format. This output may be used as an array in the user's C language programs.
f flash -- Flash memory format. This output is used to generate a file to be written to Flash memory.

The default output mode is direct.

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

Enter the base address if the Model C32-104 is not addressed at 300h.

eg:

```
>A300 -320 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 C32-104 board must be present in the system in which A300 is run, even in the ascii or Flash output modes. A300 must also know the correct base IO

address of the Model C32-104 board.
Example:

To create an ascii file in a system where the Model C32-104 is addressed at 320h,

```
>A300 -A -320 test
```

7. TI COFF LOADER

Load300 is a program for loading COFF Executable files onto the Model C32-104. In this way, users of the TI Assembler and C compiler tools may easily run their programs on the Model C32-104. It may also be used to convert COFF files to .ASC or .FLA format files. The COFF file version compatible with LOAD300 is Version 0. On newer versions of the TI linker, use the -v0 option to ensure compatibility.

Example 1:

```
>LOAD300 TIFFT
```

loads the contents of TIFFT.OUT into the correct locations on the Model C32-104.

Example 2:

```
>LOAD300 -A TIFFT
```

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

Example 3:

```
>LOAD300 -F TIFFT
```

reads the contents of TIFFT.OUT and produces TIFFT.FLA for subsequent programming of the Model C32-104's onboard Flash memory.

8. LINK PACKAGE

This section describes the utility functions for controlling the Model C32-104 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 C32-104. The necessary Model C32-104 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 Drivers and Visual Basic Examples.

int320(BASEIO)

BASEIO - Base IO Address of Model C32-104

Sends an INT0 interrupt to the TMS320C32 on the Model C32-104

go320(BASEIO)

BASEIO - Base IO Address of Model C32-104

Sets the TMS320C32 into operation.

hlt320(BASEIO)

BASEIO - Base IO Address of Model C32-104

Resets the TMS320C32, 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 C32-104 memory
BASEIO - Base IO Address of Model C32-104

Copies the array X into Model C32-104 memory starting at memory location START.

Example
C Language:

```
sendio(pgm1, 0x88, 0l, baseio);
/* copy 88h words from array pgm1 into Model C32-104 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 C32-104 memory
BASEIO - Base IO Address of Model C32-104

Copies LENGTH words of Model C32-104 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. FLASH MEMORY

The Model C32-104 features 512K Bytes of Flash memory. Flash memory generally contains bootup code, an entire application, or subroutines or data that the user's program might require.

The Flash memory is accessed from the Host CPU by bytewide IO instructions to BaseIO+3. Please see section 4.B Flash Memory Access.

The Flash memory resides in the TMS320C32's memory space starting at address 900000h. This is the Boot 3 address identified by a low INT2 input at startup.

To program the Flash memory:

1) Write the code that you wish to place in the Flash memory. See the examples on the distribution diskette.

2) Assemble your code with the '-f' flag.

```
>A300 -f myprog
```

This will create the file myprog.fla.

3) Erase any code that might be on the Flash memory.

```
>ERASE_F
```

4) Program the Flash memory with your new code.

```
>LOADF myprog
```

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. STANDALONE MODE

Standalone Mode operation refers to the situation where the Model C32-104 does not interact with the PC/104 bus except for the purpose of drawing power.
Of course, the DIO and serial IO connectors may (and probably will) be used in this mode. The jumpers are set as follows:

J6 -	CPU	OFF	Jumper removed
J11-	MASTER	OFF	Jumper removed

In this mode the Model C32-104 begins operation at power-up. The TMS320C32 is set for Boot Loader operation (the MCBL/MP pin of the TMS320C32 is set high) and your application is loaded from the Flash memory. After the loading process, the TMS320C32's program counter jumps to 1000h, which is where the start of your program should reside. Normal operation of your program then begins.

Steps for creating software for Standalone Mode operation.

Assume you have created and debugged a program in the PC environment with the Model C32-104 in Slave Mode which you wish to 'port' to Standalone Mode. Such a program might, for example, read and write to the DIO connector and perform some calculations.

Step 1.
Make the changes to your software which are needed for Standalone Mode operation. This stems from the fact that addresses 0-FFFh are reserved for Boot Loader operations and cannot be used by your program (at least, not initially).
Your program will need to start at 1000h.
If you are using interrupts, you will probably need to relocate the Interrupt Vector Table from 0 to an unused block of addresses above 1000h.

Example on disk of the changes:

SA_PRG0.ASM --> SA_PRG1.ASM

Step 2.
Create a .FLA file from this program and load it to the Flash memory.

```
>A300 -F SA_PRG1
```

Erase Flash Memory.

```
>ERASE_F
```

Load your new program into Flash memory.

```
>LOAD_F SA_PRG1
```

You can test the program without actually setting the board jumpers to Standalone Mode. Leave the board in Slave Mode. This allows for rapid modification and testing of your program.
Set the MCBL/MP pin of the TMS320C32 to BOOTLOAD operation by entering the ’b’ command in D300.
Next, using D300, start the board by entering the ’g’ command. Since the TMS320C32 is set for Boot Loader operation, the software will be loaded from Flash memory and then run, just as it is in Standalone Mode.

```
>D300
-b
-g
```

Finally, after the program has been debugged, the last step is to set the Model C32-104 jumpers for Standalone Mode, power up the board (it can still be in your PC), and watch your application run.

Note: After boot up, the RAM addresses 0-FFFh can reclaimed for program use by a dummy write to the Flash memory by the TMS320C32.

12. LIMITED BUS MASTER MODE

In Limited Bus Master Mode (abbreviated as LBM Mode), the Model C32-104 behaves as the master of the PC/104 bus in place of the Host CPU card or motherboard. TO AVOID BUS CONTENTION AND POSSIBLE DAMAGE there must be no Host CPU card or motherboard present in the system when the jumpers are set for Limited Bus Master Mode. Examples of systems using this mode are a two board set consisting of a Model C32-104 and a PC/104 serial port board, or a system of 3 Model C32-104s, consisting of one board in Limited Bus Master Mode and 2 boards in Slave Mode.

The jumpers are set as follows for LBM Mode:

J6 -	CPU	OFF	(Jumper removed)
J11-	MASTER	ON	(Jumper in place)

The Model C32-104 in LBM Mode processes many, but not all, of the signal functions normally provided by the Host CPU board or motherboard. The Model C32-104 may perform PC/104 bus I/O instructions, as well as PC/104 bus memory accesses in the first Megabyte. Thus the Model C32-104 in LBM Mode determines address bits 0-19, known as SA0-SA19, and the /IOR, /IOW, /SMEMR and /SMEMW signals on the PC/104 bus.
PC interrupts are supported. The Model C32-104 in LBM Mode may also generate the /SBHE signal to indicate high and low byte transfers, as well as the REFRESH signal. It does not support PC DMA.

A. Higher Order Address Bits
In Limited Bus Master Mode, the 10 most significant address bits on the PC/104 bus (denoted SA19-SA10) are generated by writing the desired address bit values to the HIGH_ADDRESS_BITS latch located at 818000h in the TMS320C32's memory map. The desired SA19-SA10 are written to bits 15-6 of the HIGH_ADDRESS_BITS latch. This latch need not be written to again until a change in SA19-SA10 is required.

Example 1
Desired result: SA19-SA10 will be zero on subsequent memory or I/O accesses to the PC/104 bus. Action: Write 0 to bits 15-6 of the HIGH_ADDRESS_BITS Latch (bits 5-0 are don't cares).

```
        LDI    0,R0
        LDI    @HIGH_ADDRESS_BITS_LATCH,AR2
        STI    R0,*AR2
HIGH_ADDRESS_BITS_LATCH  .word  818000h
```

Example 2

A more general example. We will later be performing a memory access at B0003H on the PC/104 bus.

```
        LDI @MEM_LOCATION,R0
        LSH -4,R0                       ; Shift register 4 to the right
        LDI @HIGH_ADDRESS_BITS_LATCH,AR2
        STI R0,*AR2
HIGH_ADDRESS_BITS_LATCH  .word  818000h
MEM_LOCATION             .word  0B0003h
```

B. /SBHE Signal

The Model C32-104 supports the transfer of 16 bit data to even addresses. In this case the high and low byte are transferred simultaneously and the /SBHE line would be set LOW to indicate that a high byte is being tranferred on the PC/104 bus.

Example:

```
        LDI    @SBHE_LOW,AR3
        LDI    *AR3,R0
SBHE_LOW  .word  810008h
```

The transfer of 8 bit data to even or odd addresses is also supported through the low byte. In this case /SBHE is set HIGH.

Example:

```
        LDI    @SBHE_HIGH,AR3
        LDI    *AR3,R0
SBHE_HIGH         .word  810009h
```

To set /SBHE low, read from address 818008h
To set /SBHE high, read from address 818009h

C. PC/104 Bus I/O Access

I/O access on the PC/104 bus is accomplished when the TMS320C32 accesses the 828000h-8283FFh region in its memory space after having set address bits SA19-SA10 and the /SBHE line as described above. To access a PC/104 bus I/O address, the TMS320C32 reads from or writes to [828000h + 10 LSBs of [I/O Port Address]]. For example, to read from port 310h, read address 828310h. The I/O Port address value may be no larger than 7FFFh.

Example 1: Write 77h to PC I/O address 3EBh. Assume that the 10 MSBs SA19-SA10 have been set. In this case, SA19-SA10 = 0.

```
        LDI    @IO3EB,AR3
        LDI    77H,R0
        STI    R0,*AR3
```

```
IO3EB          .word  8283EBh
```

Example 2: Read from PC I/O Address 401h. Assume that the 10 MSBs SA19-SA10 have been set. In this case, SA19-SA11=0, SA10=1.

```
        LDI    @IO401_10LSBs,AR3
        LDI    *AR3,R0
IO401_10LSBs       .word  828001h
```

Example 3: A general example for practical use. SA19-0 are set in this example.

```
; First set SA19-SA10
        LDI @IO_ADDRESS,R0
        LSH -4,R0
        LDI @HIGH_ADDRESS_BITS_LATCH,AR2
        STI R0,*AR2
; Next set up the IO access addr. in the 'C32 address space
        LDI @IO_ADDRESS,R0
        AND   03FFh,R0
        OR @IOBASE,R0
        STI R0,AR2
; Perform the IO Read or Write
        LDI 1234h,R0
        STI R0,*AR2
IOBASE                          .word       828000h
HIGH_ADDRESS_BITS_LATCH  .word       818000h
IO_ADDRESS                      .word       410h
```

D. PC/104 Bus Memory Access

PC/104 Bus Memory accesses may be made from 0 to 1 Mbyte (SA19-SA0, or 20 bits) by accessing the 820000h-8203ffh region in the TMS320C32 memory, after having set the higher address bits SA19-SA10 as described in 12.1 and the /SBHE line as described in 12.2.

Example 1: Assume that SA19-SA10 have been set to B000h as in Example 2. of section 12.1 and that the /SBHE line has been set (if necessary) to the level required by the slave card.

```
; Write to memory location B0003h
        LDI    @MEM03,AR3
        STI    R0,*AR3
MEM03       .word  820003h
```

Example 2: A general example for practical use. SA19-0 are set in this example.

```
; First set SA19-SA10
        LDI @MEM_ADDRESS,R0
        LSH -4,R0
        LDI @HIGH_ADDRESS_BITS_LATCH,AR2
        STI R0,*AR2
; Next set up the memory access addr. in the 'C32 address space
        LDI @MEM_ADDRESS,R0
        AND   03FFh,R0
        OR @MEMBASE,R0
        STI R0,AR2
; Perform the memory Read or Write
        LDI 1234h,R0
        STI R0,*AR2
MEMBASE                     .word       820000h
HIGH_ADDRESS_BITS_LATCH     .word       818000h
MEM_ADDRESS                 .word       0B010h
```

In summary, a memory access on the PC/104 bus is a two part operation. The address "segment" is first set. These will be the SA19-SA10 address bits. Then the "offset", SA9-SA0, are presented to the PC/104 bus during the data transfer instruction.

For additional examples, please see the listings on the distribution diskette.

E. PC/104 Bus Interrupts

This section describes the mechanism of receiving and processing interrupts through the PC/104 bus by the Model C32-104 in Limited Bus Master Mode.

Three interrupt input pins, labelled PCINT0-2 in fig. 12-1, are logically 'or'ed (if the mask bit corresponding to each PCINTx is set) to produce a single interrupt line to the TMS320C32. This line is the /INT0 input to the TMS320C32.
PCINT0-2 are connected to the PC/104 bus interrupt lines by means of jumpers or wirewrap wires at J10. For example, PCINT0 may be connected to IRQ3 on the PC/104 bus (from one PC serial port) while PCINT1 may be connected to IRQ4 (from another PC serial port). The 'INTPC' pin is not connected when the Model C32-104 is in LBM Mode.

After an /INT0 interrupt is received by the TMS320C32, the DSP reads the PC_INT_SOURCE latch located at 818001h to determine the source of the interrupt. Bits 6, 7, and 8 indicate an interrupt from PCINT0, PCINT1, and PCINT2, respectively. The PC_INT_SOURCE is cleared after it has been read.

A 3 bit interrupt mask determines the PCINTx signals to be or'ed to the /INT0 input of the TMS320C32 DSP. This mask is set by writing to bits 6-8 of the PCINT_MASK latch located at 818001h.

Example: Mask out PCINT0. Include only PCINT1 and PCINT2 as contributors to any future /INT0 interrupt signal.

```
        LDI @PCINT_MASK,AR3
        LDI 0C0h,R1
        STI R1,*AR3
PCINT_MASK      .word 818001h
```

Note that even if a particular interrupt is active but its source is masked off, its occurence may still be monitored by reading the PC_INT_SOURCE latch.

A listing of an application using the COM3 serial port and interrupts in LBM Mode is included on the distribution diskette.

Figure 12-1. Interrupt options to Model C32-104 in LBM Mode

F. REFRESH

The REFRESH pulse is generated with a read from the REFRESH address 81000Ah.

```
        LDI   @REFRESH,AR3
        LDI   *AR3,R0
REFRESH   .word  81000Ah
```

APPENDIX A

HOST PC IO ASSIGNMENTS

Port value = IO Base + Offset

Offset	Read	Write
0	RAM access (word)	
1		
2	MCBLMP Pin HIGH (b)	RAM address (b)
3	Flash memory access (byte)	
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)	MCBLMP Pin LOW (b)

APPENDIX B

Model C32-104 Memory Map

0 - FFFFh
0 Wait State SRAM

810000h - 817FFFh

Digital I/O	Read & Write	810000h-810007h
SBHE_LOW	Read	810008h
SBHE_HIGH	Read	810009h
REFRESH	Read	81000Ah

818000h - 81FFFFh

HIGH_ADDRESS_BITS	Write	818000h
Interrupt PC	Write	818001h
PCINT_MASK	Write	818001h
PCINT_SOURCE	Read	818001h

820000h - 827FFFh
/SMEMR & /SMEMW

828000h - 82FFFFh
/IOR & /IOW

900000h -
512K Byte Flash Memory wired to data bus bits D0-D7

APPENDIX C

Emulation Header

J11 is the emulation header, where the Model C32-104 is used as the target system.

EMU1	o o	GND
EMU0	o o	GND
EMU2	o o	GND
+5V	o o	KEY
EMU3	o o	GND
H3	o o	GND

APPENDIX D

Texas Instruments C Compiler

The Texas Instruments Optimizing C Compiler for the TMS320C3X and TMS320C40 may be used with the Model C32-104. 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 C32-104. 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 C32-104. Note the allocation of a small 'window' of memory for variables shared by the PC and TMS320C32. On newer versions of the TI Linker use the **-v0** option.

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 Drivers and Visual Basic Examples

MC32WIN.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 or Windows 95 is required to run these examples.

APPENDIX F

Emulator Software Notes using Code Composer V3.0(TM)

Slave Mode

In D300, set the DSP to microprocessor mode and then set it to run.

-mp

-g

In Code Composer, reset the DSP, then use the Code Composer debugging commands.

Slave Mode running a Standalone application (see page 11.2)

In D300, set the DSP to BOOTLOAD operation.

-b

Start Code Composer, click on the RUN command

In D300, set the DSP to run.

-g

In Code Composer, click on the HALT command. You may now use the Code Composer debugging commands.

Standalone or LBM Modes

Power up the Model C32-104 based system.

Start Code Composer.

In Code Composer, click on the RUN command, Then click on HALT. You may now use the Code Composer debugging commands.

REFERENCES

1. TMS320C3x User's Guide , Reference no. SPRUO31A
2. TMS320C32 User's Guide , Reference no. SPRU132B
3. Digital Signal Processing Applications with the TMS320 Family, Reference no. SPRA017
4. Technical Reference Personal Computer AT, IBM Corp. Reference no. 1502243
5. AM29F040 Flash Memory, Advanced Micro Devices, Inc. 1994