



Application Note: PC32

DATE: April 17, 1997
TO: PC32 SDK users
FROM: Technical Staff
RE: Initialization of PC32
CC:

The PC32 powers up in the boot loader mode, ready to boot an application program from its onboard dual port memory. The II Monitor program, **TERM.EXE** automatically downloads a minimal debugging kernel, known as the Talker, into dual port RAM in TI's special boot load format prior to booting the board. When the board is released from reset, the Talker program runs and initializes the PC32 for proper operation and subsequent communications with the **TERM** program.

Excerpts from the Talker are included on the listing below. When developing custom initialization code, be sure to perform each of the functions performed in this example code in order to insure proper operation.

```

*
* TALKER.ASM
*
* PC32-side talker code.
*
* This code is downloaded to the PC32 by the Host monitor.
* Then, the Host can invoke the PC32 EI3 interrupt by accessing
* BASE + 04 in Host I/O space in order to activate the
* interrupt handler herein. The handler provides basic access
* to PC32 memory and registers so that the Host can gain access to
* these resources to perform functions like COFF downloading,
* and debugging.
*
*
IMED .macro p1, p2
      LDI (p1/10000h), p2
      LSH 16, p2
      OR (p1&0ffffh), p2
      .endm

      .global main, break, vectors

XMT      .set 0
REQ      .set 1
RCV      .set 2
ACK      .set 3

IBCR     .set 0000068h
S0BCR    .set 0070128h
S1BCR    .set 00F0108h
SEM0     .set 081B000h
PC_INT   .set 0819000h
BOOTED   .set 0818800h
MAILBOX  .set 03f0h
DPRAM    .set 1000h

      .text

      .
      .
      .

      BODY of Talker code, including Talker interrupt handler

      .
      .
      .

main
;
; Reconfigure the Bus Control Registers
;
      IMED 808000h, AR0      ; Load AR0 with base addr

      IMED IBCR, R0
      STI R0, **AR0(60h)    ; Update IOBCR - 3 software WS

      IMED S0BCR, R0
      STI R0, **AR0(64h)    ; Update S0BCR - 1 software WS

      IMED S1BCR, R0
      STI R0, **AR0(68h)    ; Update S1BCR - 0 software WS

```

```

;
; Clear EI0 after boot in BL mode.
;
    IMED BOOTED,AR0      ; Load AR0 with Boot clear addr
    NOP *AR0             ; and clear.
;
; Deassert PC interrupt signal
;
    IMED PC_INT,AR0     ; Load AR0 with PC interrupt latch
    LDI 0, R0
    STI R0, *AR0       ; and clear.
;
;
; Setup stack
;
    LDP _stack
    LDI @_stack, SP
;
; Clear dual port hardware and software semaphores
;
    IMED SEM0, AR4      ; AR4 = addr of talker semaphore
    LDI 0, R0
    STI R0, *AR4       ; Release monitor semaphore

    ADDI 800h, AR4      ; AR4 = addr of terminal semaphore
    LDI 0, R0
    STI R0, *AR4       ; Release terminal semaphore
;
; Signal "Monitor-up" to host
;
    IMED DPRAM, AR0
    ADDI MAILBOX, AR0   ; AR0 points to Monitor mailbox structure
                        ; AR0(0) = data out mailbox
                        ; AR0(1) = data out semaphore
                        ; AR0(2) = data in mailbox
                        ; AR0(3) = data in semaphore

    LDI 0, R0

    RPTS 3
    STI R0, *AR0++     ; Clear monitor mailbox slots

    SUBI 4, AR0        ; Reset AR0 to XMT mailbox
    LDI 1Fh, R0
    CALL emit          ; Signal "monitor running" to PC
;
; Enable edge-sensitive interrupts
;
    LDP _BL_vec
    LDI @_BL_vec, IF   ; Zero pending interrupts, set ITTP
    LSH 8, IF
    OR 8, IE           ; Enable BREAK interrupt
    LDI 7800h, ST      ; Enable cache & edge-sensitive interrupts

hang    B hang

; Addresses
_BL_vec .word vectors
_stack .word stack

stack .space 80h

```

After the PC32 has booted and has been initialized, accesses to dual port will appear as 32-bit accesses from the perspective of the PC32. The 'C32 actually performs two, physical, 16-bit, one -wait-state accesses to dual port to virtualize the 32-bit access.

Accesses to the dual port from the PC side are also 16-bits wide - two physical accesses are required to transfer 32-bit-wide data. Because the PC is a little endian machine, the less significant 16 bits of the data are stored at the lower address and the most significant 16-bits at the higher address. Byte accesses to dual port memory are permitted but are generally not useful since the 'C32 side is capable of (virtual) 32-bit accesses only.

Protected-mode host software, such as II's TERM program, are capable of virtualizing 32-bit dual port memory accesses. Memory-memory and memory-register instructions involving 32-bit operands will execute properly and are automatically endian-correct with respect to the 'C32.