

Innovative Integration

COM44 Hardware/Software Manual

First Edition

The COM44 Hardware/Software Manual was prepared by the technical staff of Innovative Integration, May, 1995.

For further assistance contact

Innovative Integration
31352 Via Colinas #101
Westlake Village, CA 91362

PHONE: (818) 865-6150
FAX: (818) 879-1770
BBS: (818) 879-1729
email: techsprt@innovative-dsp.com
FTP: ftp.innovative-dsp.com
WWW: <http://www.innovative-dsp.com>

This document is Copyright 1996 by Innovative Integration. All rights are reserved.

Document: \\p60\manuals\com44\com44.doc

1. Introduction

The COM44 is a TIM40 standard compliant processor module based around the Texas Instruments TMS320C44 processor. The module implements a single processor with up to 512Kwords of SRAM and 4 Mbits of Flash EEPROM on the processor's local bus, and implements long distance communication port drivers to allow two COM44 modules to communicate over distances as long as 200 feet.

2. Installation

The COM44 requires a TIM40 compatible carrier card for proper operation. The COM44 is installed in one of the carrier board's TIM connector sites. Consult the hardware installation instructions provided by the carrier board's manufacturer before installing the COM44.

If the COM44 is to be used with an Innovative Integration PC44 or PCI44 processor card, the module must be installed in the following locations on each board:

Processor Card	TIM Site
PC44	TIM Site A
PCI44	TIM Site B

When installing the COM44 module, keep the following warnings in mind:

- Be careful to align the module properly over the carrier board's TIM connectors. The connectors are keyed and the module may only be installed in one direction.
- Do not use excessive force when installing the module. If the connectors do not seat properly, double check the alignment of the connectors. Make sure there are no vertical obstructions under the COM44.
- Never exert force on the semiconductor devices themselves. Always press on the printed circuit board only.

3. Hardware Features

3.1 Memory Map

The COM44 implements memory for the onboard processor on the local bus. Both SRAM and boot Flash EEPROM are provided. The following table gives the memory map for the onboard memory devices included on the card.

Memory Type	Starting Address	Physical Width	Length	Notes
Flash EEPROM	0x00300000	x8	128Kbytes	standard flash size
			512Kbytes	512K flash option
SRAM	0x7FE00000	x32	128Kwords	standard SRAM size
			512Kwords	512K SRAM option

3.2 Long Distance Comm Ports

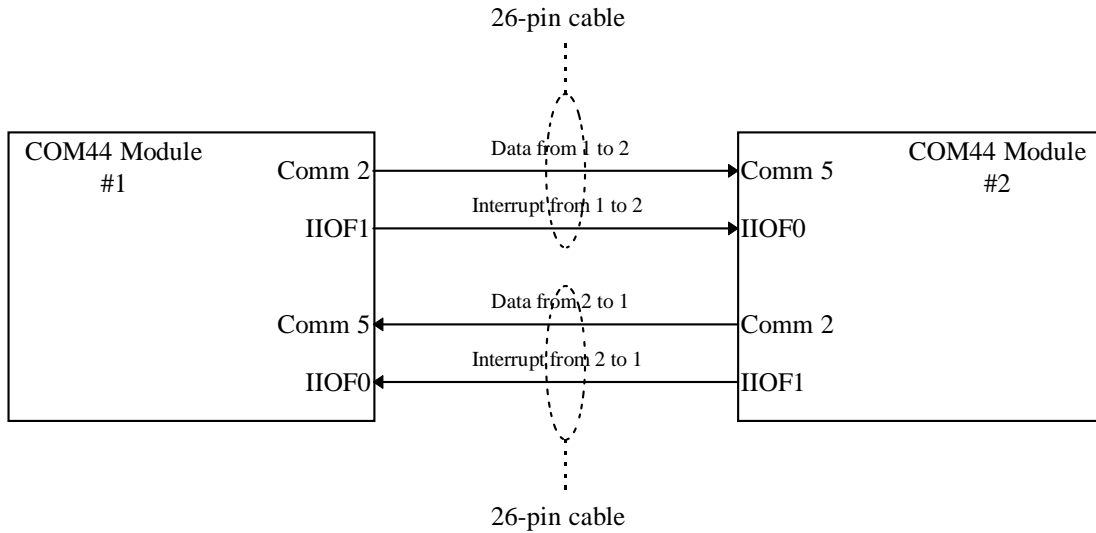
The COM44 implements two unidirectional long-distance communication ports for use in creating networks of C44 processor boards which can easily communicate with each other over long distances. Two of the processor's comm ports (2 and 5) are used for the long distance links, and the two remaining ports (1 and 4) are connected normally to the TIM40 module connectors for use on the carrier board.

The long distance communication ports are implemented as unidirectional due to the distances supported by the COM44. Communication port 2 should always be used for output and communication port 5 should always be used for inputs. Since the ports are token-forced into their respective driver directions, attempts to transfer data in the opposing direction will hang (i.e. software should not attempt to read data from the FIFO on comm port 2, and should not attempt to write to the FIFO on comm port 5).

The communication ports are connected via the two 26-pin connectors on the module. JP1 is the output connector (connected to port 2), while JP2 is the input connector (connected to port 5). Appropriate cables should be 26-conductor ribbon or flat flex with a .100" square standard header connector on each end. Cables should be connected in a straight 1-1 configuration (i.e. pin 1-1, pin 2-2, etc.). Cables are available from Innovative Integration in various lengths.

A full duplex link may be created between two COM44 modules by connecting the first module's output port to the second module's input port, and connecting the second module's output port to the first module's input port. This allows data to be passed in both directions simultaneously.

Also present on the COM44 long distance cables are signals IIOF0 and IIOF1, driven as bit I/O pins between the COM44 modules. These interrupt input/output pins allow COM44 modules on each end of the link to generate an asynchronous interrupt to the other processor. IIOF0 is received from the companion processor (which, if programmed as an interrupt input, can trip an interrupt on the receiving processor), and IIOF1 is driven as an output to the companion processor (which can similarly trip an interrupt on the companion's IIOF0 input if programmed appropriately). The diagram below shows the interconnections.



3.3 Interrupts

Per the TIM40 specification, external carrier board hardware should not attempt to drive the external interrupt input lines on the COM44 during the processor bootup. Onboard logic drives these lines after a processor reset to assure that the processor boots properly from the onboard Flash EEPROM. The onboard logic drive automatically turns off on reception of a processor IACK pulse (indicating the processor bootloader has detected the boot configuration and is proceeding to boot from Flash). Carrier hardware interrupt lines should remain tristated or non-driven after a reset and prior to the IACK pulse.

Note: IIOF0 should NEVER be driven by the carrier board while a COM44 is installed. Since IIOF1 is used as a processor interrupt received from a partner COM44, it is driven by onboard hardware (see the section above for more information).

3.4 Connector Pinouts

The following tables show the pinouts of the long-distance comm port connectors.

Pin #	Function
1	GND
2	GND
3	D0+
4	D0-
5	D1+
6	D1-
7	D2+
8	D2-
9	D3+
10	D3-
11	D4+
12	D4-
13	D5+
14	D5-

15	D6+
16	D6-
17	D7+
18	D7-
19	STRB+
20	STRB-
21	RDY+
22	RDY-
23	IIOF1+
24	IIOF1-
25	IIOF0+
26	IIOF0-

Table 1: Pinout for JP1 (output comm port connector from comm port 2)

Pin #	Function
1	GND
2	GND
3	D0+
4	D0-
5	D1+
6	D1-
7	D2+
8	D2-
9	D3+
10	D3-
11	D4+
12	D4-
13	D5+
14	D5-
15	D6+
16	D6-
17	D7+
18	D7-
19	STRB+
20	STRB-
21	RDY+
22	RDY-
23	IIOF0+
24	IIOF0-
25	IIOF1+
26	IIOF1-

Table 2: Pinout for JP2 (input comm port connector from comm port 5)

4. Software Features

4.1 Talker

The COM44 comes with Innovative Integration's Talker program preprogrammed into the Flash EEPROM on the module. The program will automatically start on deassertion of processor reset and serves to initialize the card and provide a monitor interface to Innovative Integration's host debugger environments. The Flash EEPROM may also be used to provide customer-specific boot options.

The Flash EEPROM used on the card is an AMD 29F0x0 series part, and may be reprogrammed in circuit by the 'C44 processor according to AMD's published programming guidelines. Contact Innovative Integration for more information.

Source code for the Talker program is attached below.

```

*
* TALKERB.ASM
*
* PC44-side talker code.
*
* This code is downloaded to the PC44 by the Host monitor.
* Then, the Host can invoke the PC44 IIOF2 interrupt by accessing
* the PC44 I/O base address + 802h + cpu# in order to activate the
* interrupt handler herein. The handler provides basic access
* to PC44 memory and registers so that the Host can gain access to
* these resources to perform functions like COFF downloading,
* and debugging.
*
*
        .global  main, break
        .global  ID_start, LBASE0, LBASE1,
        .global  GBASE0, GBASE1, LBASE0, LBASE1
        .global  GSIZE0, GSIZE1, LSIZE0, LSIZE1, FSIZE
        .global  TIMER0_PERIOD, TIMER1_PERIOD
        .global  TIMER0_CTL, TIMER1_CTL
        .global  GBCR, LBCR
        .global  MONITOR_MAILBOX, MONITOR_SEMAPHORE, FLASH

; DPRAM Monitor mailbox structure
MONITOR    .set MONITOR_MAILBOX
XMT        .set 0h
REQ        .set 1h
RCV        .set 2h
ACK        .set 3h

; Timer structure
CONTROL    .set 0h
COUNTER    .set 4h
PERIOD     .set 8h

        .text
;
; Obtain target PC44 address
;
break     PUSH ST
          OR 2000h, ST    ; Globally enable interrupts
          PUSH DIE
          PUSH IIE
          PUSH R0        ; break must always remain
          PUSHF R0       ; the first code assembled into
          PUSH R1        ; the monitor section.
          PUSHF R1
          PUSH R2
          PUSHF R2
          PUSH R3
          PUSHF R3
          PUSH R4
          PUSHF R4
          PUSH R5
          PUSHF R5
          PUSH R6
          PUSHF R6
          PUSH R7
          PUSHF R7
          PUSH R8
          PUSHF R8
          PUSH R9
          PUSHF R9
          PUSH R10
          PUSHF R10
          PUSH R11
          PUSHF R11
          PUSH AR0
          PUSH AR1
          PUSH AR2
          PUSH AR3
          PUSH AR4
          PUSH AR5
          PUSH AR6

```

```

PUSH AR7
PUSH DP
PUSH IRO
PUSH IRL
PUSH IIF
PUSH BK
PUSH RS
PUSH RE
PUSH RC

start   LDP addresses ; DP points to address table below
        LDI @dpram, AR0
        ADDI MONITOR, AR0 ; AR0 points to Monitor mailbox structure
            ; AR0(XMT) = data out mailbox
            ; AR0(REQ) = data out semaphore
            ; AR0(RCV) = data in mailbox
            ; AR0(ACK) = data in semaphore

        LDI @flash, AR3 ; AR3 = start of FLASH PROM
        LDI @semaphore, AR4 ; AR4 = addr of talker's semaphore

identify STIK 0, *+AR0(XMT) ; Clear monitor mailbox slots
        STIK 0, *+AR0(REQ)
        STIK 0, *+AR0(RCV)
        STIK 0, *+AR0(ACK)

        LDI 31, R0
        CALL emit ; Signal "monitor running" to PC

interpret CALL key ; Wait for valid command

        CMPI 1, R0
        BZ store ; Command 1 = Store to memory

        CMPI 2, R0
        BZ fetch ; Comamnd 2 = Fetch from memory

        CMPI 3, R0
        BZ launch ; Command 3 = Launch at address

        CMPI 4, R0
        BZ resume ; Command 4 = Resume parent program

        CMPI 5, R0
        BZ regs ; Command 5 = Register dump

        CMPI 6, R0
        BZ section ; Command 6 = Section download

        CMPI 7, R0
        BZ burnaddr ; Command 7 = Change flash burn address

        B identify

;
; Store into target address
;
store   CALL key ; Get address
        LDI R0, AR2

        CALL key ; Get data
        STI R0, *AR2 ; Store data at address

        B interpret

;
; Fetch from target address
;
fetch   CALL key ; Get address
        LDI R0, AR2

        LDI *AR2, R0 ; Load contents of address
        CALL emit ; Pass to PC

```

```

        B interpret

;
; Branch to address
;
launch CALL key
        LDI R0, AR2

        B AR2

;
; Report register save address
;
regs   LDI SP, R0
        SUBI start-break - 1, R0
        CALL emit           ; Return address of register save area

        B interpret

;
; Download a complete data/program section
;
section CALL key           ; Discard page #
        CALL key           ; Get count
        LDI R0, RC
        CALL key           ; Get address
        LDI R0, AR2

        CMPI 0, RC        ; If cnt is positive
        BGE download     ; download section.

        ABSI RC           ; Else,
        SUBI 1, RC        ; clear the section to zero.
        CALL key           ; Get fill seed value
        RPTB zero
zero   STIK 0, *AR2++
        B interpret

download LSH -20, R0
        CMPI 3h, R0      ; Is section in FLASH PROM?
        BNE ram_section

        SUBI 1, RC
        RPTB block1     ; If so, burn section into FLASH
            CALL key
            CALL burn
            CALL key
            CALL burn
            CALL key
            CALL burn
            CALL key
            CALL burn
            CALL key
            CALL burn
block1  NOP

        B interpret

ram_section SUBI 1, RC
            RPTB block2     ; Else, download section in RAM
            CALL key
block2  STI R0, *AR2++
        B interpret

;
; Change current flash burn address
;
burnaddr:
        CALL key ;get address
        LDI R0, AR3
        B interpret

;

```

```

; Resume execution
;
resume POP RC
        POP RE
        POP RS
        POP BK
        POP IIF
        POP IR1
        POP IR0
        POP DP
        POP AR7
        POP AR6
        POP AR5
        POP AR4
        POP AR3
        POP AR2
        POP AR1
        POP AR0
        POPF R11
        POP R11
        POPF R10
        POP R10
        POPF R9
        POP R9
        POPF R8
        POP R8
        POPF R7
        POP R7
        POPF R6
        POP R6
        POPF R5
        POP R5
        POPF R4
        POP R4
        POPF R3
        POP R3
        POPF R2
        POP R2
        POPF R1
        POP R1
        POPF R0
        POP R0
        POP IIE
        POP DIE
        POP ST
        RETI                ; Return to caller

;
; burn
;
; Burn contents of R0 into four cells pointed to by AR3
;
; Entry: AR3 points to next unburned FLASH byte
;        R0 contains 8-bit value to burn into next FLASH byte.
; Exit:  AR3 = AR3 + 1
; Modifies: R0 R1 AR3
;
burn    PUSH DP            ; Save state...
        LDP FLASH         ; DP points to FLASH PROM

        LDI 0AAh, R1      ; Output "byte burn" command
        STI R1, @5555h

        LDI 055h, R1
        STI R1, @2AAAh

        LDI 0A0h, R1
        STI R1, @5555h

        AND 0FFh, R0     ; Write LSByte to current FLASH address
        STI R0, *AR3

burning LBU0 *AR3, R1    ; Wait for burn to complete
        CMPI R1, R0
        BNE burning

```

```

        ADDI 1, AR3      ; Increment AR3 to point to next FLASH byte

        POP DP          ; Restore state

        RETS

;
; key
;
; Reads data in input mailbox
;
; Entry: AR0 = Base address of dual-port monitor mailbox
; Exit: R0 = 32-bit received data
; Modifies: R0 R1
;
;
key      CALL get_dport ; IF mailbox full
        LDI  *+AR0(ACK), R1      ; proceed with data receive
        BNE eat                  ; Else
        CALL release_dport      ; release semaphore and repeat
        B key

eat      LDI  *+AR0(RCV), R0      ; Get data
        LDI  0, R1
        STI  R1, *+AR0(ACK)
        CALL release_dport      ; Signal PC

        RETS

;
; emit
;
; Writes data to output mailbox
;
; Entry: R0 = 32-bit data to be output
;        AR0 = Base address of dual-port monitor mailbox
; Exit:
; Modifies: R1
;
emit     CALL get_dport ; IF mailbox empty
        LDI  *+AR0(REQ), R1      ; proceed with data transmit
        BEQ spit                 ; Else
        CALL release_dport      ; release semaphore and repeat
        B emit

spit     STI  R0, *+AR0(XMT)      ; Send data
        LDI  -1, R1
        STI  R1, *+AR0(REQ)      ; Signal PC
        CALL release_dport

        RETS

;
; get_dport
;
; Obtains dual port semaphore 0
;
; Entry:
; Exit: C44 controls monitor/terminal mailbox
; Modifies: R1
;
get_dport STIK 0, *AR4 ; Request dual port semaphore

get1     LH0 *AR4, R1 ; Wait till dual-port available
        BNE get1

        RETS

;
; release_dport
;
; Obtains dual port semaphore 0

```

```

;
; Entry:
; Exit: C44 relinquished monitor/terminal mailbox
; Modifies: R1
;
release_dport STIK -1, *AR4 ; Request dual port semaphore

        RETS

main
;
; Reconfigure the Primary Bus Control Registers
;
        LDP addresses          ; Set data page
        LDI @gbc_r, AR0 ; load AR0 with GBCR
        LDP ID_start
        LDI @GBCR, R0
        STI R0, *AR0 ; Write GBCR
        LDI @LBCR, R0
        STI R0, **AR0(4) ; Write LBCR
;
; Initialize timers per ID ROM
;
        LDP addresses
        LDI @timer0, AR0 ; Point to counter 0
        LDP ID_start
        LHO @TIMER0_CTL, R0
        STI R0, **AR0(CONTROL)
        LDI 0, R0
        STI R0, **AR0(COUNTER)
        LDI @TIMER0_PERIOD, R0
        STI R0, **AR0(PERIOD)

        ADDI 10h, AR0 ; Point to counter 1
        LHL @TIMER1_CTL, R0
        STI R0, **AR0(CONTROL)
        LDI 0, R0
        STI R0, **AR0(COUNTER)
        LDI @TIMER1_PERIOD, R0
        STI R0, **AR0(PERIOD)
;
; Size available memory and update ID table
;
; Global SRAM
        LDI @GBASE0, AR0
            LDI 0FFFFh, AR1 ; dummy rewrite address
            LSH 16, AR1
        LDI *AR0, R1 ; Save current contents
        LDI 0, IR0
        STI IR0, *AR0 ; Init GBASE1 + 0 to zero

gsize  ADDI 1000h, IR0
        LDI **AR0(IR0), R0
        STI IR0, **AR0(IR0)
            STI 0, *AR1 ; clear bus to prevent invalid reads
            CMPI IR0, **AR0(IR0) ; if address is not read/writeable,
            BNE gfail ; end of global RAM has been found
        CMPI 0, *AR0
        STI R0, **AR0(IR0)
        BEQ gsize

gfail  STI R1, *AR0 ; Restore prev contents
        CMPI 1000h, IR0
        LDIEQ 0, IR0
        STI IR0, @GSIZE0

; Local SRAM

        LDI @LBASE1, AR0
        LDI *AR0, R1 ; Save current contents
        LDI 0, IR0
        STI IR0, *AR0 ; Init LBASE1 + 0 to zero

lsize  ADDI 1000h, IR0
        LDI **AR0(IR0), R0

```

```

    STI IR0, *+AR0(IR0)
    CMPI 0, *AR0
    STI R0, *+AR0(IR0)
    BEQ lsize

    STI R1, *AR0 ; Restore prev contents
    CMPI 1000h, IR0
    LDIEQ 0, IR0
    STI IR0, @LSIZE1
    STI IR0, @FSIZE
;
; Initialize vector and trap tables
;
;     LDI @LBASE1, R0
;     LDPE R0, IVTP ; Point IVTP to base of local SRAM
;
;     ADDI 200h, R0
;     LDPE R0, TVTP ; Point TVTP to IVTP + 200h
;
; Init monitor stack
;
;     LDP addresses
;     LDI @stack, SP ; Setup stack
;
; Clear PROM
;
;     LDI @flash, AR0 ; Clear IIOF input pins for
;     IACK *AR0 ; use as interrupts.
;
; Clear dual port hardware and software semaphores
;
;     LDI @semaphore, AR4 ; AR4 = addr of talker's semaphore
;     STIK -1, *AR4 ; Release monitor semaphore
;     STIK -1, *+AR4(1) ; Release terminal semaphore
;
;     LDI @dpram, AR0
;     ADDI MONITOR, AR0 ; AR0 points to Monitor mailbox structure
;
;     STIK 0, *+AR0(XMT) ; Clear monitor mailbox slots
;     STIK 0, *+AR0(REQ)
;     STIK 0, *+AR0(RCV)
;     STIK 0, *+AR0(ACK)
;
;
; Enable Monitor interrupt
;
;     LDI 0h, IIF ; Zero pending interrupts
;     LDI 1911h, IIF ; Enable Monitor BREAK interrupt
;     LDI 2000h, ST ; Enable interrupts
;
;
; Signal "Monitor-up" to host
;
;     LDI 31, R0
;     CALL emit ; Signal "monitor running" to PC
;
;
; Attempt to boot from comm port
;
; CALL BOOT

hang B hang ; Wait for a monitor breakpoint...

```

```

BOOT:
    LDHI    0010H,AR0          ; Load peripheral mem. map start addr 100000H
    LDI     0,R0              ; Set start address flag off
    LDI     COM_LOAD,R10      ; Comm. port load subroutine address -> R10

*
*   CHECK THE IIOF1-3 FOR THE BOOT LOADER
*
CHECK:
    NOP

*-----*
*                                     COMMUNICATION PORT BOOT LOADER
*-----*
*
*   CHECK COMMUNICATION PORT INPUT CHANNEL
*
        ADDI    040H,AR0,AR3    ; Point to comm. port 0 control register addr
        LDI     5,AR1          ; Set loop counter for CHECK_CH loop
CHECK_CH: LSH3    -9,*AR3,R1    ; Check comm port input
        BNZ     LOAD0          ; If input exist, start comm port loader
        ADDI    010H,AR3      ; Point to next comm. port channel addr
        DBU     AR1,CHECK_CH   ; Check next comm. port channel input

        B       CHECK         ; ReCheck the input flags
*-----*
*                                     MEMORY BOOT LOADER
*-----*
*
*   START PROGRAM LOADING
*
LOAD0:  CALLU   R10            ; Load new word according to mem. width
        STI    AR2,*AR0      ; Set global bus control register
        CALLU   R10            ; Load new word according to mem. width
        STI    AR2,*+AR0(4)  ; Set local bus control register

LOAD2:  CALLU   R10            ; Load new word according to mem. width
        SUBI3  1,AR2,RC      ; Set block size for repeat loop
        CMPI   -1,RC         ; If 0 block size start PGM
        BEQ    IVTP_LOAD

        CALLU   R10            ; Load new word according to mem. width
        LDI    AR2,AR0       ; Set destination address
        LDI    R0,R0         ; Test start address loaded flag
        LDIZ  AR2,R9         ; Load start address if flag off
        LDI   -1,R0          ; Set start & dest. address flag on
        SUBI  1,R10          ; Sub address with loop

        CALLU   R10            ; Load block words according to mem. width
        LDI    1,R0          ; Set dest. address flag off
        ADDI   1,R10         ; Sub address without loop
        B      LOAD2         ; Jump to load a new block when loop completed

*
*   INITIALIZE IVTP AND TVTP REGISTERS
*
IVTP_LOAD: CALLU   R10            ; Load new word according to mem. width
           LDPE   AR2,IVTP      ; Load the IVTP pointer
TVTP_LOAD: CALLU   R10            ; Load new word according to mem. width
           LDPE   AR2,TVTP      ; Load the TVTP pointer
           CALLU   R10            ; Load new word according to mem. width
           IACK   *AR2          ; Send out IACK signal out
           BU     R9            ; Branch to the start of the program

;-----;
;   COMMUNICATION PORT BOOT LOADER SUBROUTINE
;-----;
LOOP_C   RPTB    LOAD_C         ; PGM load loop
COM_LOAD LSH3    -9,*AR3,R1    ; Check comm port input
        BZ     COM_LOAD        ; Wait for comm port input
        LDI   *+AR3(1),AR2     ; Read a new 32 bits word
        LDI   R0,R0           ; Test load address flag

```

```
LOAD_C      BNN      C_END
C_END      STI      AR2,*AR0++(1) ; Store new word to dest. address
           RETSU    ; Return from subroutine
```

```
;  
; Address table  
;  
addresses:  
gbcr      .word 000100000h  
stack     .word stk  
dpram     .word 0fff80000h  
semaphore .word MONITOR_SEMAPHORE  
flash     .word FLASH  
timer0    .word 0100020h  
  
stk       .space 80h
```