MOTOROLA
SEMICONDUCTOR
APPLICATION NOTE

Order this document
by AN1066/D

AN1066

Interfacing the MC68HC05C5 SIOP to an I^{2}C Peripheral

By Naji Naufel

INTRODUCTION

When designing a system based on a standard, non-custom-designed, microcontroller unit (MCU), the user is faced with the problem of not having all the desired peripheral functions on-chip. This problem can be solved by interfacing the MCU to an off-chip set of peripherals. An ideal interface is a synchronous serial communication port. Unfortunately, these peripherals may not have a serial interface that is compatible with the Motorola simple synchronous serial I/O port (SIOP).

This document demonstrates how the SIOP on the MC68HC05C5 can be interfaced to an I^{2}C peripheral, the PCF8573 clock/timer. The MC68HC05C5 was chosen because its SIOP has a programmable clock polarity.

The serial peripheral interface (SPI) on the MC68HC05C4 cannot be used in the interface because the SPI pins cannot be used as output pins when the SPI is off.

SIOP DEFINITION

The SIOP (see Figure 1) is a three-wire master/slave system including serial clock (SCK), serial data input (SDI), and serial data output (SDO). A programmable option determines whether the SIOP handles data most significant bit (MSB) or least significant bit (LSB) first.

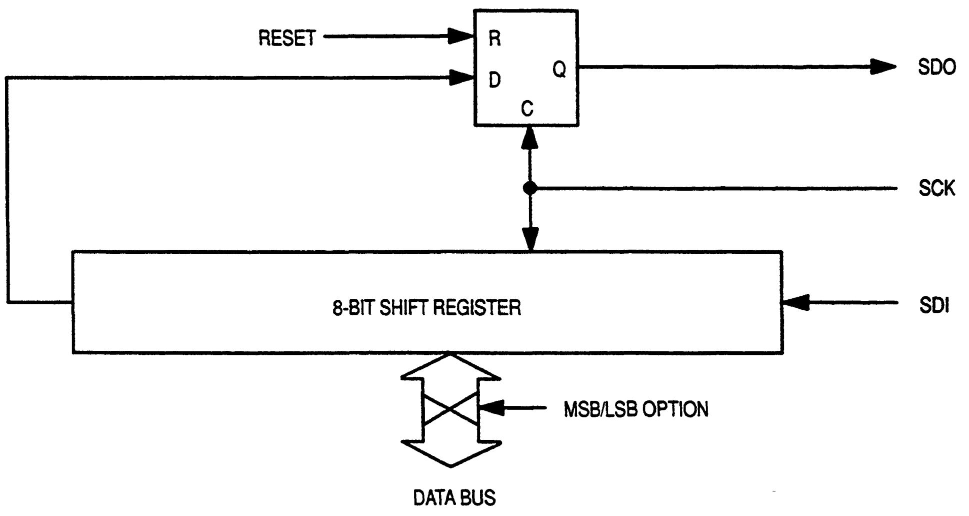

Figure 1. SIOP Block Diagram

SIOP SIGNAL FORMAT

The SCK, SDO, and SDI signals are discussed in the following paragraphs.

SCK

The state of SCK between transmissions can be either a logic one or a logic zero. The first falling edge of SCK signals the beginning of a transmission. At this time, the first bit of received data is presented to the SDI pin, and the first bit of transmitted data is presented at the SDO pin (see Figure 2). When CPOL = 0, the first falling edge occurs internal to the SIOP. Data is captured at the SDI pin on the rising edge of SCK. Subsequent falling edges shift the data and accept or present the next bit. When CPOL = 1, transmission ends at the eighth rising edge of SCK. When CPOL = 0, transmission ends at the eighth falling edge of SCK.

Format is the same for master mode and slave mode except that SCK is an internally generated output in master mode and an input in slave mode. The master mode transmission frequency is fixed at E/4.

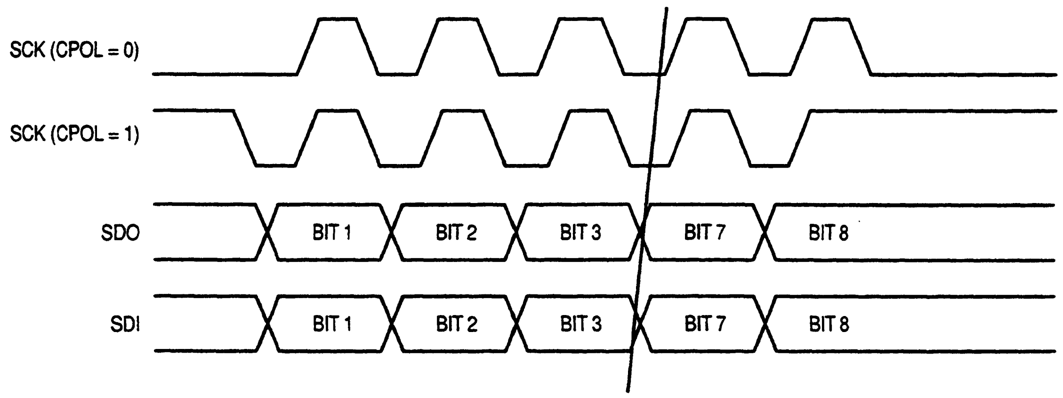

Figure 2. SIOP Timing

SDO

The SDO pin becomes an output when the SIOP is enabled. The state of SDO always reflects the value of the first bit received on the previous transmission, if a transmission occurred. Prior to enabling the SIOP, PB5 can be initialized to determine the beginning state, if necessary. While the SIOP is enabled, PB5 cannot be used as a standard output since that pin is coupled to the last stage of the serial shift register. On the first falling edge of SCK, the first data bit to be shifted out is presented to the output pin.

SDI

The SDI pin becomes an input when the SIOP is enabled. New data may be presented to the SDI pin on the falling edge of SCK. Valid data must be present at least 100 ns before the rising edge of the clock and remain valid 100 ns after that edge.

SIOP REGISTERS

The SIOP contains the following registers: SCR, SSR, and SDR.

SIOP Control Register (SCR)

This register, located at address $000A, contains three bits (see Figure 3).

	7	6	5	4	3	2	1	0
$0A	0	SPE	0	MSTR	CPOL	0	0	0
RESET:	0	0	0	0	1	0	0	0

Figure 3. SIOP Control Register

SPE — Serial Peripheral Enable

When set, this bit enables the SIOP and initializes the port B data direction register (DDR) such that PB5 (SDO) is output, PB6 (SDI) is input, and PB7 (SCK) is an input in slave mode and an output in master mode. The port B DDR can be subsequently altered as the application requires, and the port B data register (except for PB5) can be manipulated as usual; however, these actions could affect the transmitted or received data. When SPE is cleared, port B reverts to standard parallel I/O without affecting the port B data register or DDR. SPE is readable and writable any time, but clearing SPE while a transmission is in progress will abort the transmission, reset the bit counter, and return port B to its normal I/O function. Reset clears this bit.

MSTR — Master Mode

When set, this bit configures the SIOP for master mode, which means that the transmission is initiated by a write to the data register and SCK becomes an output, providing a synchronous data clock at a fixed rate of the bus clock divided by 4. While the device is in master mode, SDO and SDI do not change function; these pins behave exactly as they would in slave mode. Reset clears this bit and configures the SIOP for slave operation. MSTR may be set at any time regardless of the state of SPE. Clearing MSTR will abort any transmission in progress.

CPOL — Clock Polarity

When CPOL is set, SCK idles high, and the first data bit is seen after the first falling edge. When CPOL is cleared, the SCK idles low, and the first data bit is seen after the first falling edge, which occurs internally (see Figure 2).

SIOP Status Register (SSR)

Located at address $000B, the SSR contains only two bits (see Figure 4).

	7	6	5	4	3	2	1	0
$0B	SPIF	DCOL	0	0	0	0	0	0
RESET:	0	0	0	0	0	0	0	0

Figure 4. SIOP Status Register

SPIF — Serial Peripheral Interface Flag

This bit is set on the last rising clock edge, indicating that a data transfer has occurred. SPIF has no effect on further transmissions and can be ignored without problem. SPIE is cleared by reading the SSR with SPIF set, followed by a read or write of the SDR. If it is cleared before the last edge of the next byte, it will be set again. Reset also clears this bit.

DCOL — Data Collision

DCOL is a read-only status bit that indicates an invalid access to the data register has been made. A read or write of SDR during a transmission results in invalid data being transmitted or received. DCOL is cleared by reading the SSR with SPIF set, followed by a read or write of the SDR. If the last part of the clearing sequence is done after another transmission has been started, DCOL will be set again. If DCOL is set and SPIF is not set, clearing DCOL requires turning the SIOP off, then turning it back on using the SPE bit in the SCR. Reset also clears this bit.

SIOP Data Register (SDR)

Located at address $000C, SDR is both the transmit and receive data register (see Figure 5). This system is not double buffered; thus, any write to this register destroys the previous contents. The SDR can be read at any time, but if a transmission is in progress, the results may be ambiguous. Writes to the SDR while a transmission is in progress can cause invalid data to be transmitted and/or received. This register can be read and written only when the SIOP is enabled (SPE = 1).

	7	6	5	4	3	2	1	0
$0C								
RESET:	U	U	U	U	U	U	U	U

Figure 5. SIOP Data Register

I²C DEFINITION

The inter IC (I²C) is a two-wire half-duplex serial interface with data transmitted/received MSB first. The two wires are a serial data line (SDA) and a serial clock line (SCL).

The protocol consists of a start condition, slave address, n bytes of data, and a stop condition (see Figure 6). Each byte is followed by an acknowledge bit. A start condition is defined as a high-to-low transition on SDA while SCL is high; a stop condition is defined as a low-to-high transition on SDA while SCL is high (see Figure 9). An acknowledge is a low logic level sent by the addressed receiving device during the ninth clock period. A master receiver signals the end of data by not generating an acknowledge after the last byte has left the slave device.

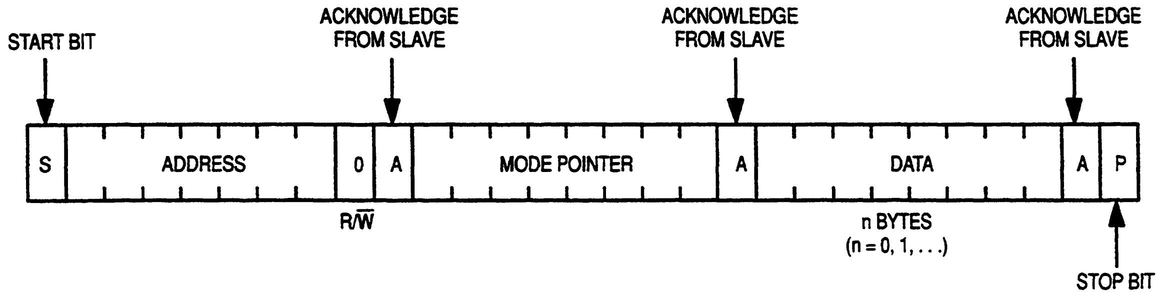

Figure 6. PCF8573 Serial Data Format

INTERFACING THE SIOP TO THE PCF8573

The PCF8573 has an address of 11010 A1 A0, where A1 and A0 give the device a one-of-four address assigned by two hardware pins. Bit 0 of the address byte is the read/write indicator (see Figure 7).

7	6	5	4	3	2	1	0
1	1	0	1	0	A1	A0	R/W̄
MSB							LSB

Figure 7. Address Byte

The byte following the address byte is the mode pointer used to control register access inside the PCF8573. Subsequent bytes following the mode pointer contain data read from or written to the clock/timer. Clock data is in binary-coded decimal format with two digits per byte.

HARDWARE DESCRIPTION

The SIOP is used as master by setting the MSTR bit in the SPCR. PB7/SCK is connected to SCL. Since the PCF 8573 has a bidirectional data line (SDA) and the SIOP has separate input and output pins, the SDO and the SDI pins need to be connected. A resistor must be used for this connection because port B is not open-drain (see Figure 8). The SEC pin, which goes high every second, is connected to PA7, which is polled by software to keep a seconds count.

When receiving data from the clock timer, an $FF is transmitted by the SIOP, which makes the resistor (R3), in series with the SDO pin, look like a pullup to V_{DD}; therefore, SDO will not interfere with data coming from the SDA pin.

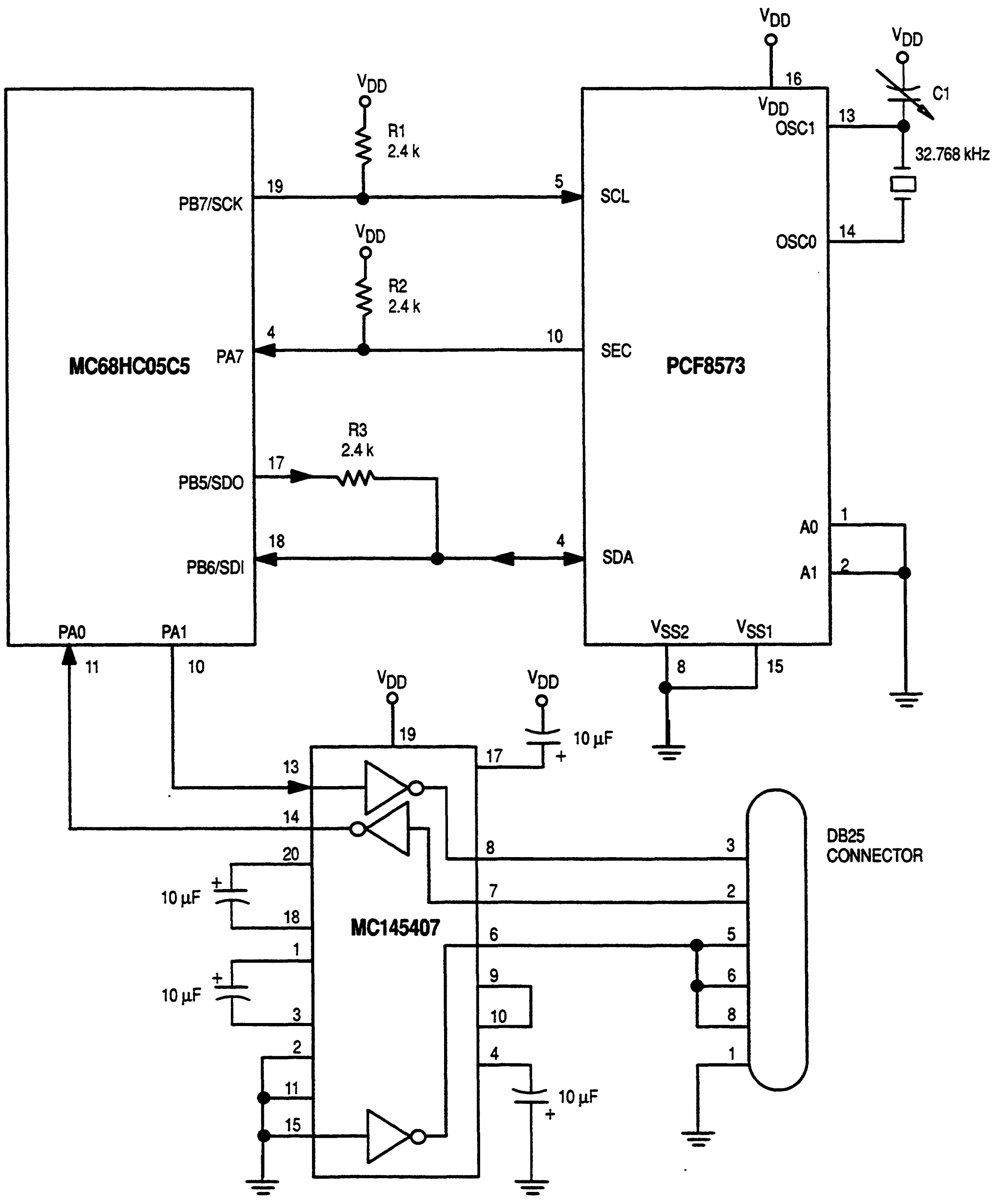

Figure 8. MC68HC05C5 Connection to PCF8573

SOFTWARE DESCRIPTION

To generate the timing required by the I^{2}C, the user has to manipulate the port B pins as I/O and SIOP pins (see Figure 9). Before any data transactions, PB5 and PB7 are initialized high. While the SIOP is off (SPE = 0), PB5 is cleared to zero while PB7 is still high, creating a start condition. The SIOP is then enabled with CPOL = 0 and MSTR = 1 and a byte is transmitted. After transmission is complete, the SIOP is turned off (SPE = 0), and PB7 is toggled high, then low, to generate the acknowledge clock. If the MCU is sending data, PB5 is forced high during the acknowledge pulse; otherwise, it is forced low to let the slave know that the byte has been received. If needed, the stop condition is accomplished by clearing PB5, setting PB7, then setting PB5.

To satisfy the 100-kHz serial clock maximum rating of the PCF8573, the MC68HC05C5 must be slowed to run at a bus speed of 250 kHz, which gives a serial clock rate of 62.5 kHz.

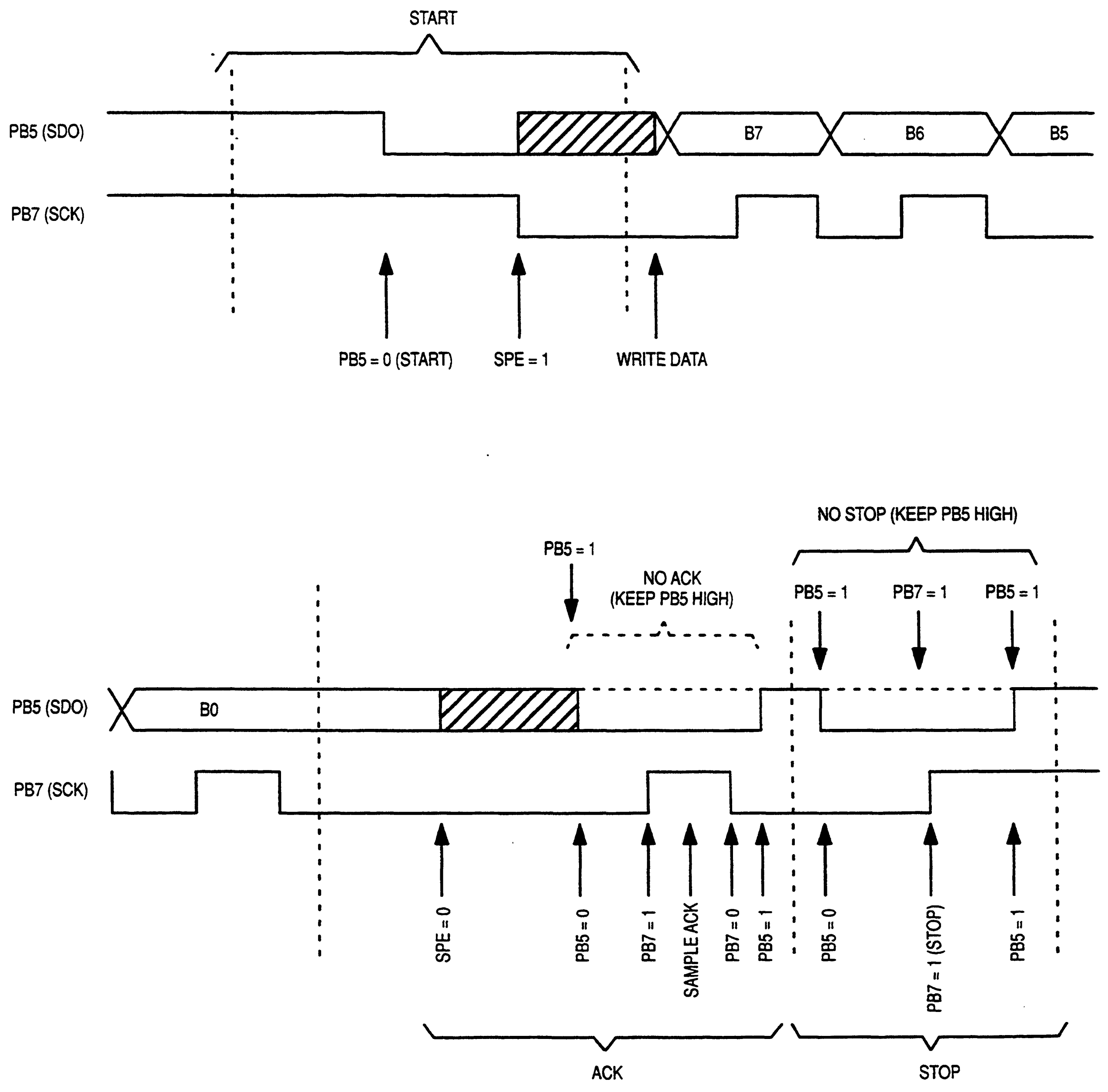

Figure 9. SIOP-Generated Timing

SOFTWARE APPLICATION

In demonstrating how the SIOP is interfaced to an I^{2}C peripheral, the author developed a complex application having time and calendar functions.

This application interfaces serially with a terminal to allow the user to initialize the PCF8573 with the time and date (see Figure 8). After the software prompts the user to enter the date (month, day, hour, and minutes), it starts displaying the information every second (see Figure 10). Every second the SEC pin goes high, telling the software to read the PCF8573 data and display it along with the software-kept seconds.

To initialize clock data, use the following sequence:

Send $D0 with start bit (ADDRESS)
Send $00 without start bit (CONTROL)
Send hours data without start bit
Send minutes data without start bit
Send day data without start bit
Send month data without start bit
Generate stop bit

To read clock data, use the following sequence:

Send $D0 with start bit (ADDRESS)
Send $00 without start bit (CONTROL)
Set up for low acknowledge bit transmit
Send $D1 with start bit (ADDRESS)
Send $FF without start bit to receive hours
Send $FF without start bit to receive minutes
Send $FF without start bit to receive day
Send $FF without start bit to receive month
Generate stop bit

Since the MC68HC05C5 does not have a universal asynchronous receiver transmitter (UART), the interface to the terminal was implemented in software. See subroutines INCHAR and OUTCHAR in **APPENDIX A PROGRAM LISTING.**

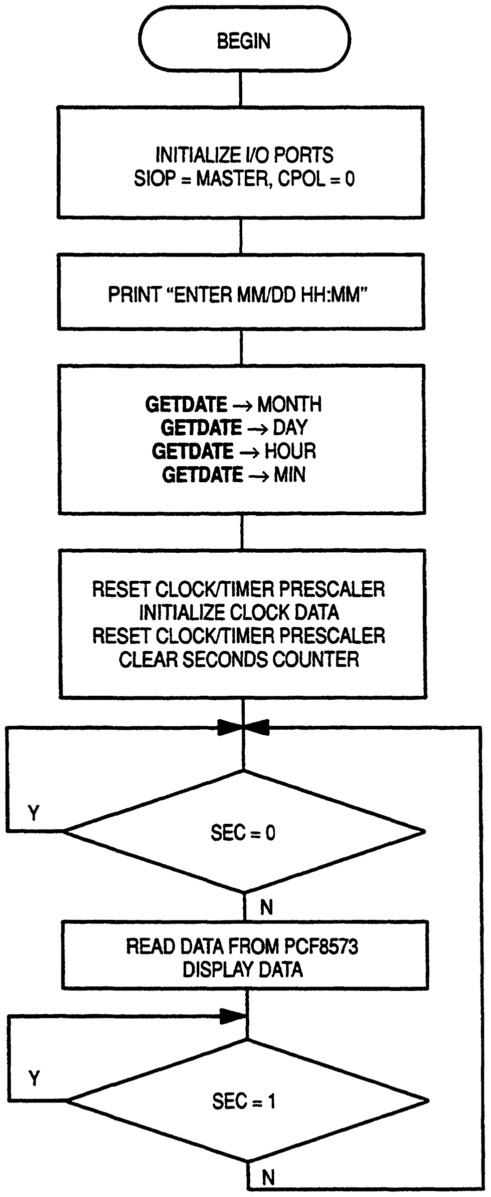

Figure 10. Program Flowchart

APPENDIX A. PROGRAM LISTING

```
                 *********************************************************************
                 *
                 * This program is written to demonstrate interfacing the MOTROLA
                 * Simple Serial I/O (SIOP) bus to the SIGNETICS IIC bus.
                 * The 2 devices used are the MC68HC05C5 MCU and the PCF8573 clock/timer.
                 * Bus speed is 250 Khz.
                 * The MCU is used as a master and the clock/timer is used as a
                 * slave. Some software intervention has to be done so that the
                 * SIOP meets all IIC specifications.
                 * The MCU displays clock data on a terminal screen at 2400 baud
                 *
                 *     Written by : Naji Naufel
                 *                  CSIC MCU Design
                 *                  Austin, Texas
                 *
                 *********************************************************************

0000             porta     equ    $00        ;port a data register
0001             portb     equ    $01        ;port b data register
0002             portc     equ    $02        ;port c data register
0004             ddra      equ    $04        ;port a data direction register
0005             ddrb      equ    $05        ;port b data direction register
0006             ddrc      equ    $06        ;port c data direction register
000a             spcr      equ    $0a        ;spi control register
000b             spsr      equ    $0b        ;spi status register
000c             spdr      equ    $0c        ;spi data register
                 *
00d1             raddr     equ    $d1        ;peripheral address for read
00d0             waddr     equ    $d0        ;peripheral address for write

0080             ram       equ    $80        ;start of ram space
                 *
0080                       org    ram
0080             sec       rmb    1          ;seconds byte
0081             control   rmb    1          ;control byte
0082             ack       rmb    1          ;acknowledge polarity
0083             hour      rmb    1
0084             min       rmb    1
0085             month     rmb    1
0086             day       rmb    1
0087             savx      rmb    1
0088             sava      rmb    1
0089             xtemp     rmb    1
008a             count     rmb    1
008b             InChar    rmb    1
008c             OutChar   rmb    1
008d             atemp     rmb    1
008e             cdelay    rmb    1          ;delay variable for serial routines
```

```


                     ****************************************************************
                     * start of program

0200                           org     $0200
                     * all timing is based on a 500 Khz crystal
                     *

0200                 begin     equ     *
0200 a6 02                     lda     #%00000010
0202 b7 00                     sta     porta        ;TX pin high
0204 b7 04                     sta     ddra         ;PA1=TX pin, PA0=RX pin
0206 a6 a0                     lda     #%10100000   ;pb7=pb5=output, pb6=input
0208 b7 01                     sta     portb
020a b7 05                     sta     ddrb
020c ae 03                     ldx     #3
020e d6 04 07                  lda     delays,x
0211 b7 8e                     sta     cdelay       ;2400 baud
0213 a6 10                     lda     #%00010000   ;mstr=1, cpol=0, siop still off
0215 b7 0a                     sta     spcr
0217 b7 82                     sta     ack          ;ack flag non-zero, high acknowledge

0219 cd 03 85                  jsr     crlf
021c cd 03 76                  jsr     outmsg       ;print "ENTER MM/DD HH:MM"
021f cd 02 85                  jsr     getdate      ;get month
0222 b7 85                     sta     month
0224 cd 03 90                  jsr     inchar       ; dummy char '/'
0227 cd 02 85                  jsr     getdate      ;get day
022a b7 86                     sta     day
022c cd 03 90                  jsr     inchar       ; dummy space
022f cd 02 85                  jsr     getdate      ;get hours
0232 b7 83                     sta     hour
0234 cd 03 90                  jsr     inchar       ;dummt ':'
0237 cd 02 85                  jsr     getdate      ;get minutes
023a b7 84                     sta     min
023c cd 03 90        again     jsr     inchar       ;wait for <CR>
023f a1 0d                     cmp     #$0d
0241 26 f9                     bne     again
0243 cd 03 85                  jsr     crlf
                     *
                     * issue a reset prescaler command
                     *
0246 a6 20                     lda     #$20
0248 b7 81                     sta     control
024a cd 02 d2                  jsr     addrcntl
024d cd 02 cb                  jsr     stop

                     * initialize the clock
                     *
0250 a6 00                     lda     #$00
0252 b7 81                     sta     control
0254 cd 02 d2                  jsr     addrcntl     ;send address/control bytes
0257 cd 02 db                  jsr     senddta      ;send 4 data bytes
                     *
                     * issue a reset prescaler command
                     *
025a a6 20                     lda     #$20
025c b7 81                     sta     control
025e cd 02 d2                  jsr     addrcntl
0261 cd 02 cb                  jsr     stop

0264 3f 80                     clr     sec                  ;clear seconds counter
0266 0f 00 fd        sec_pin   brclr   7,porta,*            ;wait for SEC pin to go high
0269 cd 03 31                  jsr     dispdata             ;display clock data
026c 0e 00 fd                  brset   7,porta,*            ;wait until pin goes low
026f 20 f5                     bra     sec_pin
                     *
0271 45 4e 54 45     msg       fcc     "ENTER MM/DD HH:MM"
     52 20 4d 4d
     2f 44 44 20
     48 48 3a 4d
     4d
0282 0d 0a 04                  fcb     $0d,$0a,$04

```

```

                   *******************************************************************
                   * This routine reads 2 ASCII characters and converts them into
                   * 2 BCD digits in Acc. A.
                   *******************************************************************

0285               getdate   equ     *
0285 cd 03 90                jsr     inchar      ;get character
0288 a0 30                   sub     #$30        ;convert to binary
028a 48                      lsla
028b 48                      lsla
028c 48                      lsla
028d 48                      lsla                ;make it upper nibble
028e b7 88                   sta     sava
0290 cd 03 90                jsr     inchar      ;get second ASCII char.
0293 a0 30                   sub     #$30
0295 bb 88                   add     sava        ;2 BCD digit is in Acc. A now
0297 81                      rts

                   ************************************************************
                   * Convert a binary byte in Acc. A into a 2 digit BCD number
                   * in Acc. A and display it as 2 ASCII chars.
                   ************************************************************

0298               bin_dec   equ     *
0298 5f                      clrx                ;clear number of subtraction counter
0299 a0 0a         sub_more  sub     #10         ;see how many times it is divisible
029b 2b 03                   bmi     no_tens     ;by 10
029d 5c                      incx                ;increment counter
029e 20 f9                   bra     sub_more    ;subtract more tens
02a0 ab 0a         no_tens   add     #10         ;restore number to positive
02a2 58                      lslx                ;put 10's digit in upper nibble of X
02a3 58                      lslx
02a4 58                      lslx
02a5 58                      lslx
02a6 bf 88                   stx     sava
02a8 bb 88                   add     sava        ;merge both nibbles in Acc. A
02aa ad 42                   bsr     bcd         ;display the 2 digits
02ac 81                      rts
```

```

                   ********************************************************************
                   *
                   * this subroutine transfers a byte from the hc05's spi to the iic
                   * peripheral. data is in reg X upon entry.
                   * w_start is the entry point for sending a start bit.
                   * start is the entry point for transfering data without a start condition.
                   *
                   *******************************************************************
                   *
02ad               w_start   equ     *
02ad 1e 01                   bset    7,portb      ;take SCL line high
02af 1b 01                   bclr    5,portb      ;start condition
02b1               nostart   equ     *
02b1 1c 0a                   bset    6,spcr       ;enable spi, SPE=1
02b3 bf 0c                   stx     spdr         ;send data
02b5 0f 0b fd      wait      brclr   7,spsr,wait  ;wait for end of transmition
                   *
02b8 1d 0a                   bclr    6,spcr       ;clear SPE, disable spi
                   *
02ba 3d 82                   tst     ack          ;test acknowledge flag
02bc 26 04                   bne     hi_ack       ;keep ack high
02be 1b 01         lo_ack    bclr    5,portb      ;else, clear ack bit
02c0 20 02                   bra     hi_ack1      ;generate ack clock
                   *
02c2 1a 01         hi_ack    bset    5,portb      ;send high ACK bit
02c4 1e 01         hi_ack1   bset    7,portb      ;take pb7 (SCL) high
02c6 1f 01                   bclr    7,portb
02c8 1a 01                   bset    5,portb      ;return data pin high
02ca 81                      rts
```

```



                   *****************************************************************
                   *
                   * the following routine (stop) creates a stop condition
                   *
                   *****************************************************************
                   *
02cb               stop      equ    *
02cb 1b 01                   bclr   5,portb       ;bring sda low
02cd 1e 01                   bset   7,portb       ;bring scl high
02cf 1a 01                   bset   5,portb       ;bring sda high
02d1 81                      rts





                   *****************************************************************
                   *
                   * the following routine sends 2 bytes. an address byte followed
                   * by a control byte in control.
                   *
                   *****************************************************************
                   *
02d2               addrcntl  equ    *
02d2 ae d0                   ldx    #waddr        ;(r/w=0)
02d4 ad d7                   bsr    w_start       ;send address with start condition
02d6 be 81                   ldx    control
02d8 ad d7                   bsr    nostart       ;send control byte without start
02da 81                      rts



                   *****************************************************************
                   *
                   * the following routine sends 4 bytes.
                   *
                   *****************************************************************
                   *
02db               senddta   equ    *
02db be 83                   ldx    hour
02dd ad d2                   bsr    nostart       ;send hours
02df be 84                   ldx    min
02e1 ad ce                   bsr    nostart       ;send minutes
02e3 be 86                   ldx    day
02e5 ad ca                   bsr    nostart       ;send days
02e7 be 85                   ldx    month
02e9 ad c6                   bsr    nostart       ;send months
02eb ad de                   bsr    stop          ;stop condition
02ed 81                      rts
```

```

                   *****************************************************************
                   * Output 2 BCD digits in A as ASCII chars.
                   *****************************************************************

02ee               bcd       equ     *
02ee b7 88                   sta     sava              ;save A
02f0 cd 04 0b                jsr     outlhf            ;output left half
02f3 b6 88                   lda     sava
02f5 cd 04 0f                jsr     outrhf            ;output right half
02f8 81                      rts


                   *****************************************************************
                   *
                   * the following routine reads a data byte.
                   *
                   ****************************************************************
                   *
02f9               read      equ     *
02f9 a6 00                   lda     #$00
02fb b7 81                   sta     control
02fd 4c                      inca
02fe b7 82                   sta     ack          ;high ack bit (ack non-zero)
0300 cd 02 d2                jsr     addrcntl     ;send address/control
0303 ae d1                   ldx     #raddr       ;(r/w=1)
0305 ad a6                   bsr     w_start      ;send address with start condition
0307 3f 82                   clr     ack          ;low ack bit
0309 ae ff                   ldx     #$ff         ;and read 4 data bytes
030b ad a4                   bsr     nostart      ;keep mosi open drain (high)
030d b6 0c                   lda     spdr         ;get received data
030f b7 83                   sta     hour         ;hours
0311 ae ff                   ldx     #$ff
0313 ad 9c                   bsr     nostart
0315 b6 0c                   lda     spdr
0317 b7 84                   sta     min          ;minutes
0319 ae ff                   ldx     #$ff
031b cd 02 b1                jsr     nostart
031e b6 0c                   lda     spdr
0320 b7 86                   sta     day          ;days
0322 3c 82                   inc     ack          ;high ack bit for last bit received
0324 ae ff                   ldx     #$ff
0326 cd 02 b1                jsr     nostart
0329 cd 02 cb                jsr     stop         ;end session
032c b6 0c                   lda     spdr
032e b7 85                   sta     month        ;months
0330 81                      rts
```

```

                         ********************************************************************
                         * This service routine increments the seconds counter
                         * and displays the clock data on the screen every second.
                         ********************************************************************

0331                     dispdata  equ    *
0331 a6 0d                         lda    #$0d
0333 cd 03 ca                      jsr    outchar      ;send <CR>
0336 cd 02 f9                      jsr    read         ;read 4 bytes from clock

0339 b6 85                         lda    month        ;display month
033b a4 1f                         and    #$1f
033d ad af                         bsr    bcd          ;output 2 BCD digit
033f a6 2f                         lda    #'/
0341 cd 03 ca                      jsr    outchar      ;outout '/'

0344 b6 86                         lda    day          ;display day
0346 a4 3f                         and    #$3f
0348 ad a4                         bsr    bcd
034a a6 20                         lda    #$20
034c cd 03 ca                      jsr    outchar      ;output space

034f b6 83                         lda    hour         ;display hours
0351 a4 3f                         and    #$3f
0353 ad 99                         bsr    bcd
0355 a6 3a                         lda    #':
0357 cd 03 ca                      jsr    outchar      ;output ':'

035a b6 84                         lda    min          ;display minutes
035c a4 7f                         and    #$7f
035e cd 02 ee                      jsr    bcd
0361 a6 3a                         lda    #':
0363 cd 03 ca                      jsr    outchar      ;output ':'

0366 b6 80                         lda    sec          ;display seconds
0368 cd 02 98                      jsr    bin_dec      ;convert seconds to BCD and display

036b b6 80                         lda    sec          ;read seconds byte
036d 4c                            inca                ;increment it
036e a1 3c                         cmp    #60
0370 26 01                         bne    not_sixty    ;not 60 yet
0372 4f                            clra
0373 b7 80               not_sixty sta    sec          ;update seconds counter
0375 81                            rts
```

```

                     ********************************************************
                     * The following are the various routines associated with
                     * displaying data.
                     ********************************************************

0376                 outmsg    equ     *                  ;print character string
0376 5f                        clrx
0377 d6 02 71        prtmsg    lda     msg,x              ;get message character
037a a1 04                     cmp     #$04               ;EOT yet?
037c 27 06                     beq     finish             ;yes.
037e cd 03 ca                  jsr     outchar            ;output character
0381 5c                        incx                       ;increment index
0382 20 f3                     bra     prtmsg
0384 81              finish    rts

                     *****************************************************

0385                 crlf      equ     *                  ;print new line
0385 a6 0d                     lda     #$0d
0387 cd 03 ca                  jsr     outchar
038a a6 0a                     lda     #$0a
038c cd 03 ca                  jsr     outchar
038f 81                        rts


                     ********************************************************************
                     *       Register A and location InChar receive the character typed,
                     *       parity stripped and mapped to upper case.  X is unchanged.
                     *       For HC05C5 PA1 and PA0 are txd and rxd respectively.
                     *       i.e. transmit from PA1 and receive from PA0
                     *
                     *       Interrupts are masked on entry and unmasked on exit.
                     ********************************************************************


0390                 inchar    equ     *                  ;input character from terminal
0390 bf 89                     stx     xtemp              ;save X
0392 a6 08                     lda     #8                 ;number of bits to read
0394 b7 8a                     sta     count
0396 9d              getc4     nop                        ;unmask to allow service, then
0397 9b                        sei                        ;mask while looking for start bit
0398 00 00 fd                  brset   0,porta,*          ;wait for hilo transition
039b a6 02                     lda     #2
039d ad 63                     bsr     delay              ;delay 1/2 bit to middle of start bit
039f 00 00 f4                  brset   0,porta,getc4      ;false start bit test

                     *                                    ;main loop for getc
03a2 a6 02           getc7     lda     #2
03a4 ad 5c                     bsr     delay              ;6       common delay routine
03a6 a6 06                     lda     #6
03a8 ad 58                     bsr     delay              ;6
03aa 01 00 00                  brclr   0,porta,getc6      ;5       test input and set c-bit

03ad 36 8b           getc6     ror     InChar             ;5       add this bit to the byte
03af b6 8a                     lda     count              ;3       time-wasting way to decr count
03b1 4a                        deca                       ;3
03b2 c7 00 8a                  sta     >count             ;5       extd addr to waste extra cycle
03b5 26 eb                     bne     getc7              ;3       still more bits to get(see?)

03b7 9d                        nop                        ;2       re-enable interrupts
03b8 a6 02                     lda     #2
03ba ad 46                     bsr     delay              ;3       wait out the 9th bit
03bc a6 06                     lda     #6
03be ad 42                     bsr     delay              ;3
03c0 b6 8b                     lda     InChar             ;get assembled byte
03c2 a4 7f                     and     #%1111111          ; mask off parity bit
03c4 ad 06                     bsr     putc1              ;echo it back
03c6 b6 8b                     lda     InChar             ;get assembled byte
03c8 9b                        sei                        ;re-enable interrupts
03c9 81                        rts
```

```
                        ***********************************************************

03ca                    outchar equ       *           ;output character to terminal
03ca bf 89                      stx       xtemp       ;don't forget about X
03cc                    putc1   equ       *           ;sneaky entry from getc to avoid clobbering x
03cc b7 8c                      sta       OutChar
03ce b7 8d                      sta       atemp       ;save it in both places
03d0 a6 09                      lda       #9          ;going to put out
03d2 b7 8a                      sta       count       ;9 bits this time
03d4 5f                         clrx                  ;for very obscure reasons
03d5 98                         clc                   ;this is the start bit
03d6 9b                         sei                   ;mask interrupts while sending
03d7 20 02                      bra       putc2       ;jump in the middle of things

                        *       main loop for outchar

03d9 36 8c              putc5   ror       OutChar     ;5   get next bit from memory
03db 24 04              putc2   bcc       putc3       ;3   now set or clear port bit
03dd 12 00                      bset      1,porta     ;5
03df 20 04                      bra       putc4       ;3
03e1 13 00              putc3   bclr      1,porta     ;5
03e3 20 00                      bra       putc4       ;3   equalize timing
03e5 a6 02              putc4   lda       #2
03e7 ad 19                      bsr       delay       ;6
03e9 a6 06                      lda       #6
03eb ad 15                      bsr       delay       ;6
03ed 3a 8a                      dec       count       ;5
03ef 26 e8                      bne       putc5       ;3   still more bits
03f1 12 00                      bset      1,porta     ;5   send stop bit
03f3 9d                         nop                   ;2   re-enable interrupts
                       *
03f4 a6 02                      lda       #2
03f6 ad 0a                      bsr       delay       ;6   delay for the stop bit
03f8 a6 06                      lda       #6
03fa ad 06                      bsr       delay       ;6
03fc be 89                      ldx       xtemp       ;3   restore X and
03fe b6 8d                      lda       atemp       ;3   of course A
0400 9b                         sei                   ;2   re-enable interrupts
0401 81                         rts                   ;6


                        ***********************************************************
                        *      delay --- precise 1/2 bit time delay for getc/putc
                        ***********************************************************

                        * caller loop overhead      assumes 24 cycles in external loop

0402                    delay    equ     *
0402 4a                          deca
0403 26 fd                       bne     delay
0405 9d                          nop
0406 81                          rts

                        * 1/2 bit delay = 24 cycles overhead + (6*A)+8+8, where A=2
                        * 1 bit delay = 24 cycles overhead + [(6*A)+8+8] + [(6*B)+8+8], A=2, B=6

                        *-----------------------------------------------------------------*
                        *      delays for baud rate calculation
                        *-----------------------------------------------------------------*
0407 20                 delays   fcb     32          ;300 baud
0408 08                          fcb     8           ;1200 baud
0409 02                          fcb     2           ;4800 baud
040a 01                          fcb     1           ;9600 baud
```

```

                         ***********************************************************
                         * Output the left nibble of Acc A as ASCII character.
                         ***********************************************************

040b                     outlhf    equ     *
040b 44                            lsra
040c 44                            lsra
040d 44                            lsra
040e 44                            lsra
040f a4 0f               outrhf    and     #$0f
0411 ab 30                         add     #$30         ;make ASCII
0413 cd 03 ca                      jsr     outchar      ;send character to terminal
0416 81                            rts


                         **************************************************************


1ffa                               org     $1ffa
1ffa 02 00               irqv      fdb     begin
1ffc 02 00               swiv      fdb     begin
1ffe 02 00               resetv    fdb     begin
```

```
iicc5                                          page  11
InChar       008b *0045 0394 0405 0408
OutChar      008c *0046 0417 0428
ack          0082 *0036 0071 0182 0269 0273 0286
addrcntl     02d2 *0219 0095 0102 0109 0270
again        023c *0086 0088
atemp        008d *0047 0418 0448
bcd          02ee *0251 0159 0307 0313 0319 0325
begin        0200 *0059 0496 0497 0498
bin_dec      0298 *0146 0330
cdelay       008e *0048 0068
control      0081 *0035 0094 0101 0108 0222 0267
count        008a *0044 0379 0395 0397 0420 0438
crlf         0385 *0358 0073 0089
day          0086 *0040 0079 0239 0285 0311
ddra         0004 *0021 0062
ddrb         0005 *0022 0065
ddrc         0006 *0023
delay        0402 *0459 0384 0389 0391 0402 0404 0435 0437 0444 0446
                        0461
delays       0407 *0471 0067
dispdata     0331 *0300 0114
finish       0384 *0354 0350
getc4        0396 *0380 0385
getc6        03ad *0394 0392
getc7        03a2 *0388 0398
getdate      0285 *0128 0075 0078 0081 0084
hi_ack       02c2 *0187 0183
hi_ack1      02c4 *0188 0185
hour         0083 *0037 0082 0235 0277 0317
inchar       0390 *0376 0077 0080 0083 0086 0129 0136
irqv         1ffa *0496
lo_ack       02be *0184
min          0084 *0038 0085 0237 0281 0323
month        0085 *0039 0076 0241 0291 0305
msg          0271 *0118 0348
no_tens      02a0 *0152 0149
nostart      02b1 *0175 0223 0236 0238 0240 0242 0275 0279 0283 0288
not_sixty    0373 *0337 0335
outchar      03ca *0414 0302 0309 0315 0321 0327 0351 0360 0362 0488
outlhf       040b *0481 0253
outmsg       0376 *0346 0074
outrhf       040f *0486 0255
porta        0000 *0018 0061 0113 0115 0382 0385 0392 0430 0432 0440
portb        0001 *0019 0064 0173 0174 0184 0187 0188 0189 0190 0203
                        0204 0205
portc        0002 *0020
prtmsg       0377 *0348 0353
putc1        03cc *0416 0407
putc2        03db *0429 0424
putc3        03e1 *0432 0429
putc4        03e5 *0434 0431 0433
putc5        03d9 *0428 0439
raddr        00d1 *0028 0271
ram          0080 *0031 0033
read         02f9 *0265 0303
resetv       1ffe *0498
sava         0088 *0042 0135 0138 0157 0158 0252 0254
savx         0087 *0041
sec          0080 *0034 0112 0329 0332 0337
sec_pin      0266 *0113 0116
senddta      02db *0234 0103
spcr         000a *0024 0070 0176 0180
spdr         000c *0026 0177 0276 0280 0284 0290
spsr         000b *0025 0178
stop         02cb *0202 0096 0110 0243 0289
sub_more     0299 *0148 0151
swiv         1ffc *0497
w_start      02ad *0172 0221 0272
waddr        00d0 *0029 0220
wait         02b5 *0178 0178
xtemp        0089 *0043 0377 0415 0447
```

Literature Distribution Centers:
USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan.
ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.

MOTOROLA

A26879-0 PRINTED IN USA (1994) MPS/POD MCU YGACAA