Electromechanical Timer Replacement

The Galactic Timer

Author: *Michael Kirkhart*
GSE. Inc.
Farmington Hills, MI
USA
email: kirkhart@rust.net

OVERVIEW

The Swiss Army Timer is a generic, programmable timer circuit which can electronically replicate the functionality of some of the more popular electromechanical timer relay circuits. Using a PIC12C508 along with a 24C04A serial EEPROM (for configuration storage), the Swiss Army Timer can function as an on delay timer, off delay timer, one-shot, re-triggerable one-shot, astable multivibrator, and enabled astable multivibrator. Its event counter and auto-resetting event counter modes can replicate the function of an electromechanical counter as well. It uses two inputs (a timer input and timer reset input) and has one output. The two inputs and the output can be programmed to be either active high or active low, and the active edge of the timer input can be programmed to be either low-to-high or high-to-low. Time intervals can be programmed from 0.1 seconds to 6553.5 seconds, and terminal counts can be programmed from 1 to 65535.

MODES OF OPERATION

The Swiss Army Timer can operate in one of the following eight modes:

- Mode 1 - On delay timer: When the timer input goes active, the output waits for N seconds before going active. When the timer input goes inactive, so does the output.
- Mode 2 - Off delay timer: When the timer input goes active, the output goes active. When the timer input goes inactive, the output waits for N seconds before going inactive.
- Mode 3 - One shot: When an active transition of the timer input occurs, the output goes active for N seconds. If another active transition of the timer input occurs while the output is active, it is ignored.
- Mode 4 - Retriggerable one shot: This mode is similar to the one shot mode except if an active transition of the timer input is detected while the output is active, the output will remain active for N seconds after detection of the transition.
- Mode 5 - Astable multivibrator: This mode replicates the action of an oscillator. The output goes active for N seconds, and then goes inactive for M seconds. This sequence repeats indefinitely.
- Mode 6 - Enabled astable multivibrator: This mode is similar to the astable multivibrator mode except the sequence is reset and the output is held inactive if the timer input is not active.
- Mode 7 - Event counter: Each active transition of the timer input is counted. When this count reaches N counts, the output goes active and remains active until the timer reset input becomes active. When the timer reset input goes inactive, the accumulated count goes back to zero. The count is reset any time the timer reset input goes active.
- Mode 8 - Auto-reset event counter: This mode is similar to the event counter except, the next active transition of the timer input, after the terminal count is reached, will automatically reset the counter and deactivate the output. The timer reset input can still be used to manually reset the accumulated count.

HARDWARE

The Swiss Army Timer circuit consists of; a PIC12C508 8-pin microcontroller, a 24C04A serial EEPROM, an output indicator LED, some resistors, some decoupling capacitors, and several connection points. The timer input is connected to the GP5 I/O pin, the timer reset input is connected to the GP4 I/O pin, and the timer output is tied to the GP2 I/O pin. An HLMP-4700 low current LED is connected to the timer output via a 1 Kohm resistor and serves as an output high indicator. The PIC12C508 is configured to use the internal 4 MHz RC oscillator, and the GP3/$\overline{\text{MCLR}}$ pin is programmed to function as a $\overline{\text{MCLR}}$ input. For non-volatile storage of the operation configuration, a 24C04A serial EEPROM is used. The SDA pin of the EEPROM is tied to the GP1 I/O pin of the PIC12C508, and the SCL pin of the EEPROM is tied to the GP0 pin of the PIC12C508. The CPU reset, EEPROM SDA, and EEPROM SCL lines are brought out as connection points. This is to allow for re-programming the timer with the EEPROM in-circuit. A PC program for in-circuit configuration of the Swiss Army Timer is being worked on, but is not yet complete. For testing, it was necessary to remove the EEPROM from the circuit and use a PROM programmer to change the timer configuration.

Since this circuit was meant to be generic, all I/O was left as logic level. No power supply circuit was included in this circuit for the same reason; thus, an external +5V supply is necessary to power the circuit.

SOFTWARE

The software consists of an initialization block, a main loop, debouncer state machines for each of the inputs, a mode sequencer, a small state machine for each of the operational modes available, several utility subroutines, and subroutines for accessing the EEPROM.

KEY VARIABLES

- Mode state: This one byte variable is used by the sequencer mode state machines, and can vary in value from 0 to 2. Some state machines only have two states, whereas most have three states.
- Target counter: This two byte variable is used for timing/counting purposes. When in a timer mode, if this value is greater than zero, it is decremented by the time handler in the main loop every 0.1 seconds. Therefore, to setup a timed interval, the mode state machine initializes this variable to the number of tenths of a second to time. When the variable goes to zero, the time interval has elapsed. When in a counter mode, the counter mode state machines directly initialize and decrement this variable.
- Rollover counter: This two byte variable is used to determine when a tenth of a second of time has elapsed. It is initialized to 391, which corresponds to the number of RTCC rollovers per one tenth of a second. Whenever the RTCC rolls over, this counter is decremented. When it reaches zero, a tenth of a second of time has elapsed since it was initialized. It is re-initialized, and the process repeats indefinitely.
- Timer input debounce state: This one byte variable is used by the timer input debounce state machine, and can vary in value from 0 to 3. Whenever this variable is 0 or 1, the input is considered to be low, and if it is 2 or 3, the input is considered to be high.
- Timer reset input debounce state: This one byte value is used by the timer reset input debounce state machine. It operates similarly to the timer input debounce state variable.

CONFIGURATION BLOCK

These values, which are read from the EEPROM and are stored in contiguous memory locations, are used to configure the operation of the Swiss Army Timer. They are as follows:

- Mode: This one byte value determines which of the eight available modes the Swiss Army Timer operates in. These values are defined as follows:
  - 0 = on delay timer
  - 1 = off delay timer
  - 2 = one-shot
  - 3 = retriggerable one-shot
  - 4 = astable multivibrator
  - 5 = enabled astable multivibrator
  - 6 = event counter
  - 7 = auto-resetting event counter
- Time parameter 1: This two byte value is used as the delay/pulse width/terminal count value. In the astable modes, it is used as the on time value. It can vary from 1 to 65535.
- Time parameter 2: This two byte value is used by the astable modes as the off time value. It can vary from 1 to 65535.
- Timer options: This one byte value is used to select the active levels of the timer input, reset input, timer output, and timer input active edge. Four of the bits in this value are used for this purpose, and they are as follows:
  - Bit 0: If high, the timer input is configured as active low. Otherwise, the timer input is active high.
  - Bit 1: If high, the timer output is configured as active low. Otherwise, the timer output is active high.
  - Bit 2: If high, the timer reset input is configured as active low. Otherwise, the timer reset input is active high
  - Bit 3: If high, the timer input active edge is configured as high to low. Otherwise, the timer input active edge is low to high.

INITIALIZATION BLOCK

Invoked on a CPU reset, this section trims the on-board oscillator, sets up the OPTION register, initializes the GPIO register as well as the TRIS register, initializes the file registers (RAM) used by the program, and reads the configuration from the EEPROM.

MAIN LOOP

After initialization, the main loop runs indefinitely. Each pass through the main loop, we:

- Check to see if the RTCC has rolled over (this will occur every 256 microseconds). If it has, we decrement the rollover counter and the debounce counter. When the rollover counter is zero (this occurs every 0.1 seconds), we re-initialize it to 391 (0.1 seconds / 256 microseconds = 390.625), and if the timer target counter is not equal to zero and we are not in an event counter mode, we decrement it as well.
- If the debounce counter is zero, we re-initialize it to 20 (20 * 256 microseconds = 5.12 milliseconds) and call the timer input and reset input state machine routines. Based on the current state of each of these state machines and the timer option flags in the configuration, the timer input and reset input flags in the flags register are set to either active or inactive. If an active transition of the timer input has occurred, the timer input edge flag in the flags register is also set.
- The mode sequencer routine is called. Based on the configured mode of operation, the sequencer jumps to the appropriate mode state machine routine. Each of these state machines determine whether the timer output should be active or inactive based on the current mode sequencer state. This is done by either setting or clearing the output active flag in the flags register. They also determine whether to switch states based on the timer input and reset input, and the value of the timer target counter.
- The timer output is set either high or low based on the output active flag in the flags register and the active output level setting in the timer option flags in the configuration.
- The watchdog timer is reset, and we jump back to the beginning of the loop.

DEBOUNCER STATE MACHINES

There are two of these 4-state machines. One is for the timer input, and one is for the timer reset input. For a given input level to be considered valid, the state of the corresponding input pin must remain the same for at least two passes through the state machine routine. This helps minimize false input triggering if mechanical switches are used.

MODE SEQUENCER

The mode sequencer routine is nothing more than a jump table. Based on the configured mode of operation, the sequencer causes program execution to jump to the appropriate mode state machine. Each one of these state machines jump back to the end of the mode sequencer when they are finished.

MODE STATE MACHINES

There are eight small mode state machines, one for each of the available modes of operation. Each one of these state machines determine the state of the timer output based on its current state.

They also determine whether or not to change state based on the timer inputs, the timer target counter value. When a state change occurs, the value of the timer target counter may also be updated.

FIGURE 1: ON DELAY MODE STATE DIAGRAM

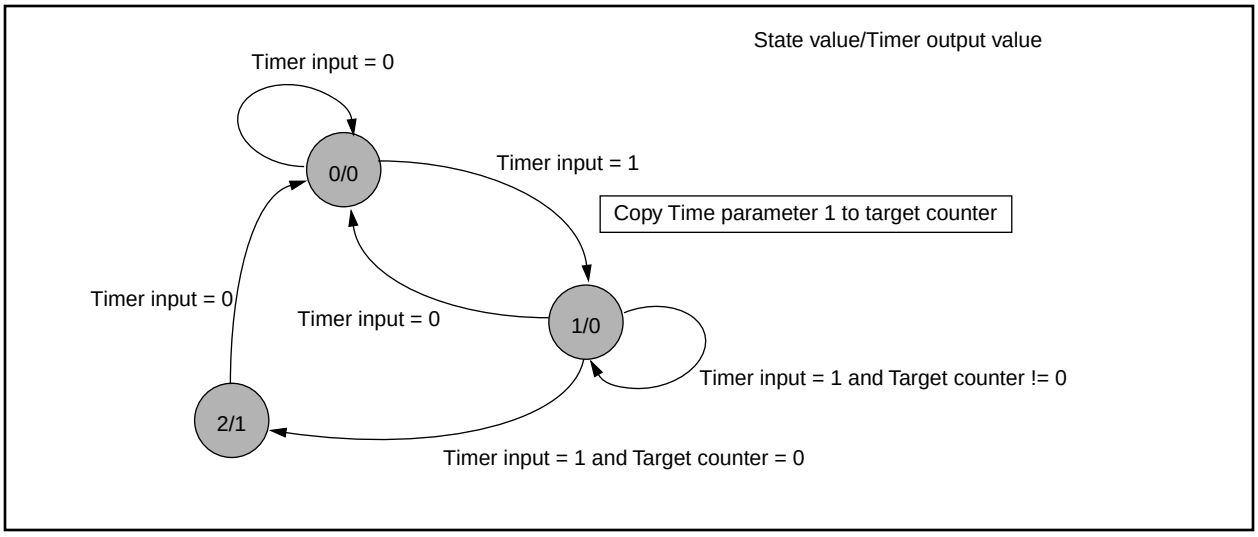


FIGURE 2: OFF DELAY MODE STATE DIAGRAM

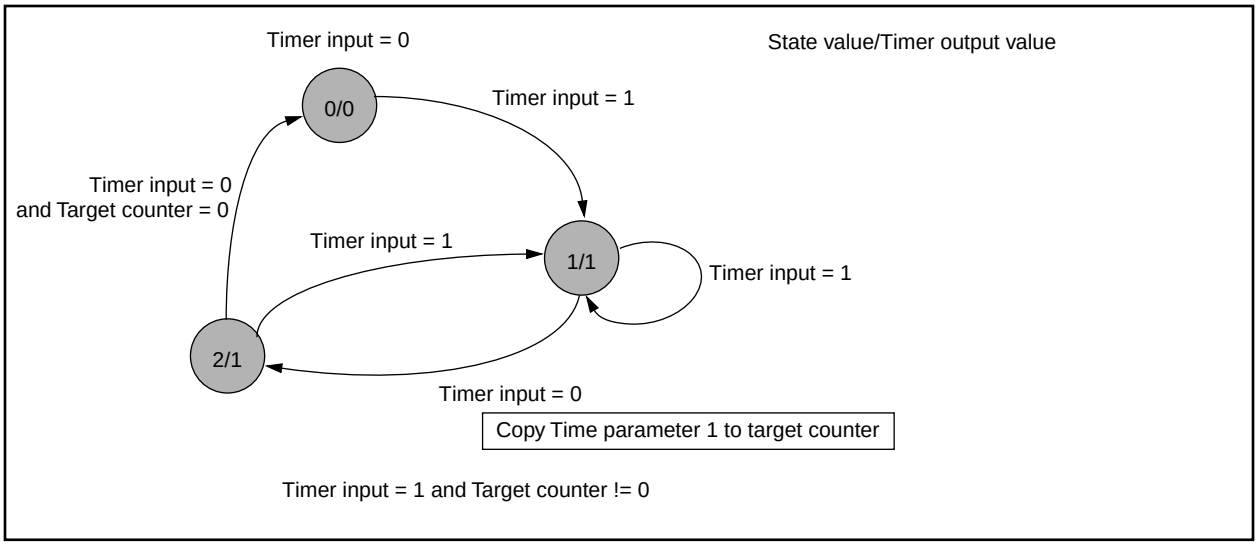


FIGURE 3: ONE SHOT MODE STATE DIAGRAM

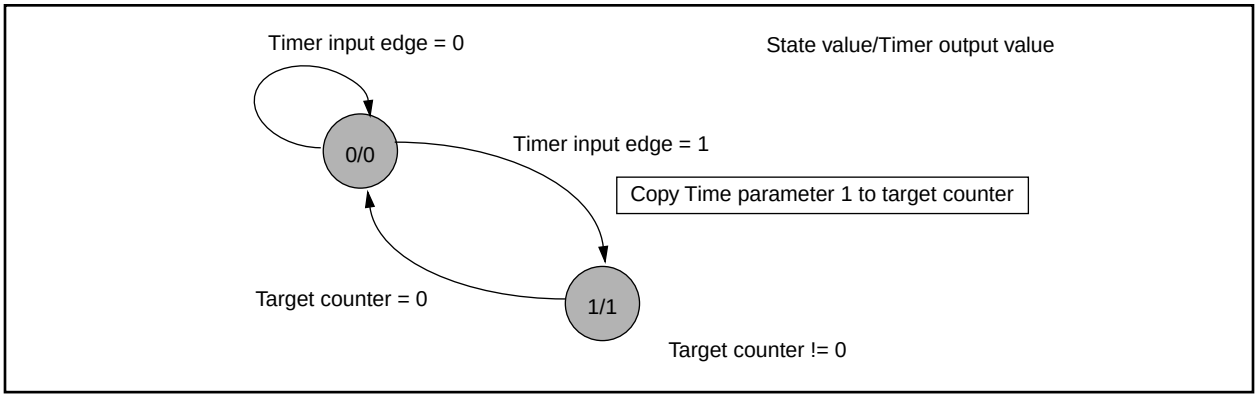


FIGURE 4: RETRIGGERABLE ONE SHOT MODE STATE DIAGRAM

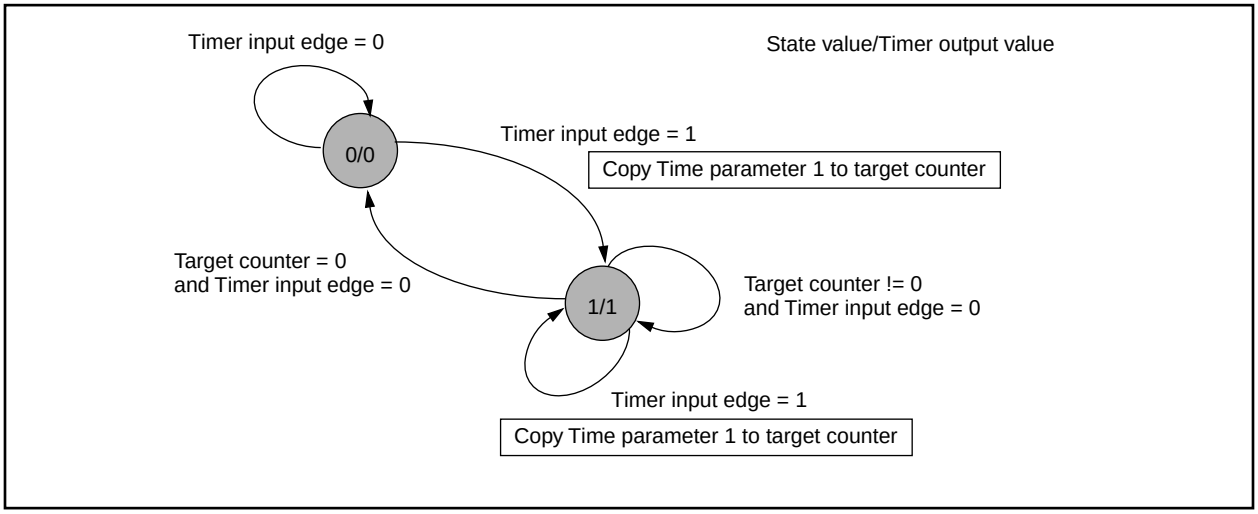


FIGURE 5: ASTABLE MULTIVIBRATOR MODE STATE DIAGRAM

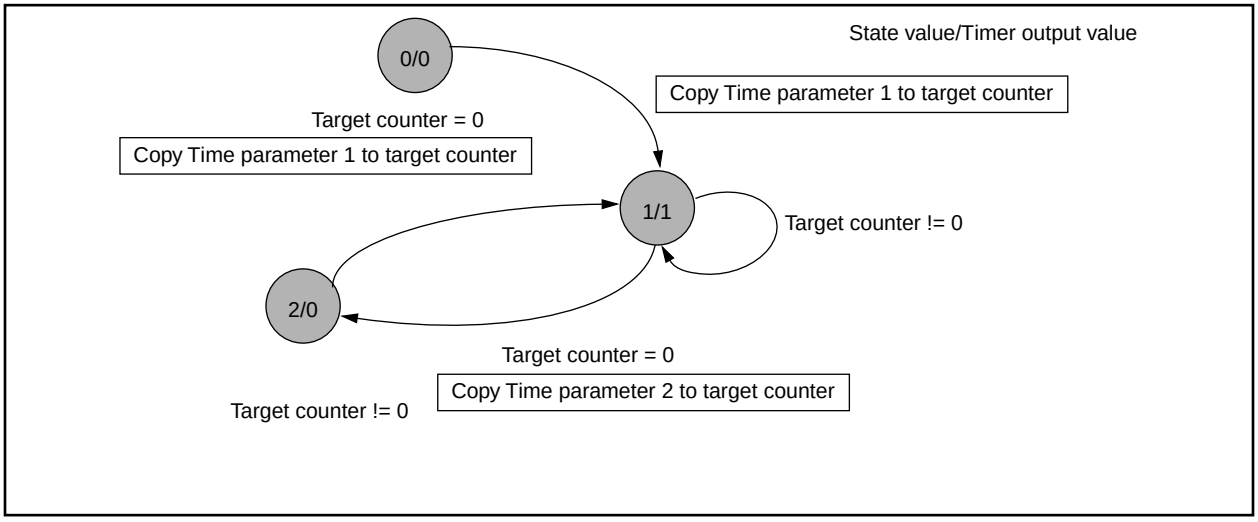


FIGURE 6: ENABLED ASTABLE MULTIVIBRATOR MODE STATE DIAGRAM

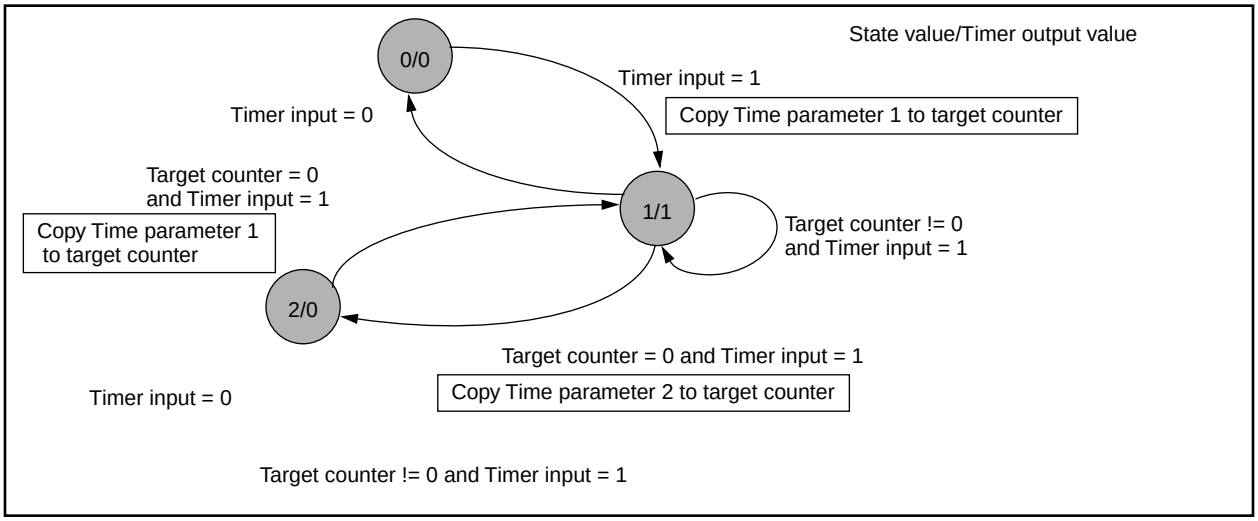


FIGURE 7: EVENT COUNTER MODE STATE DIAGRAM

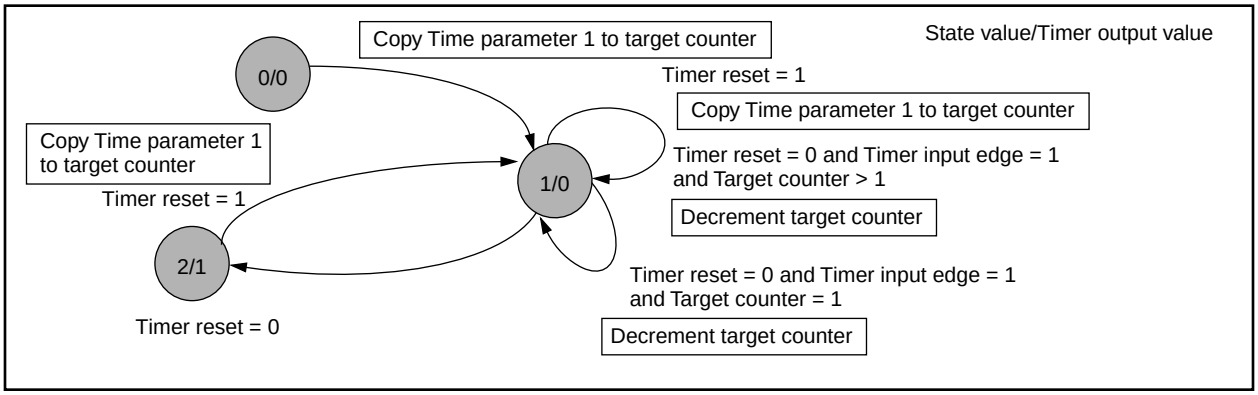


FIGURE 8: AUTO-RESET EVENT COUNTER MODE STATE DIAGRAM

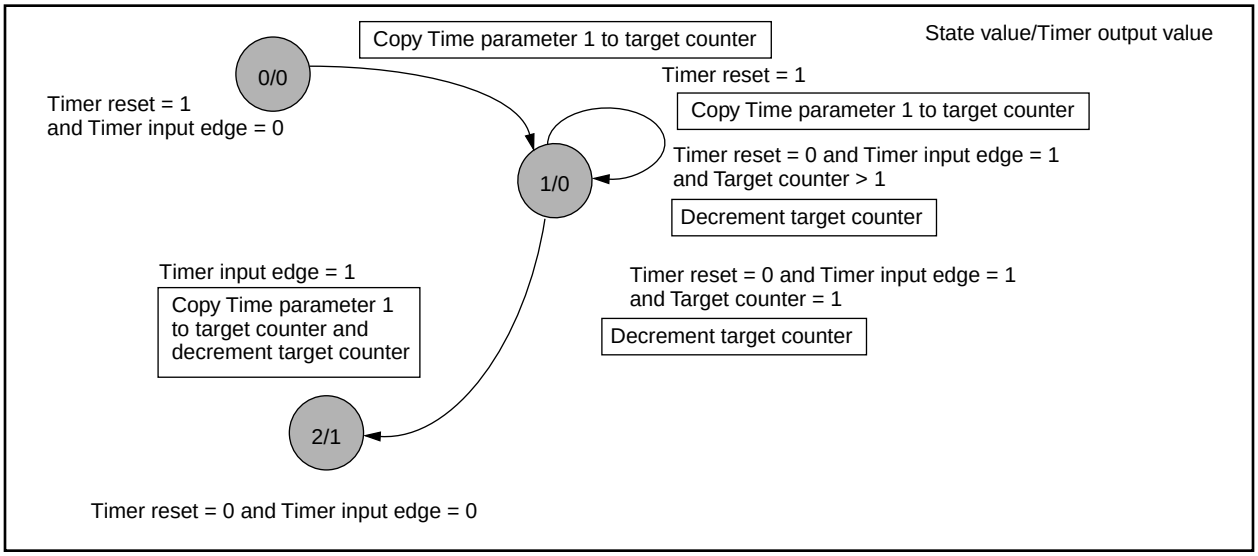
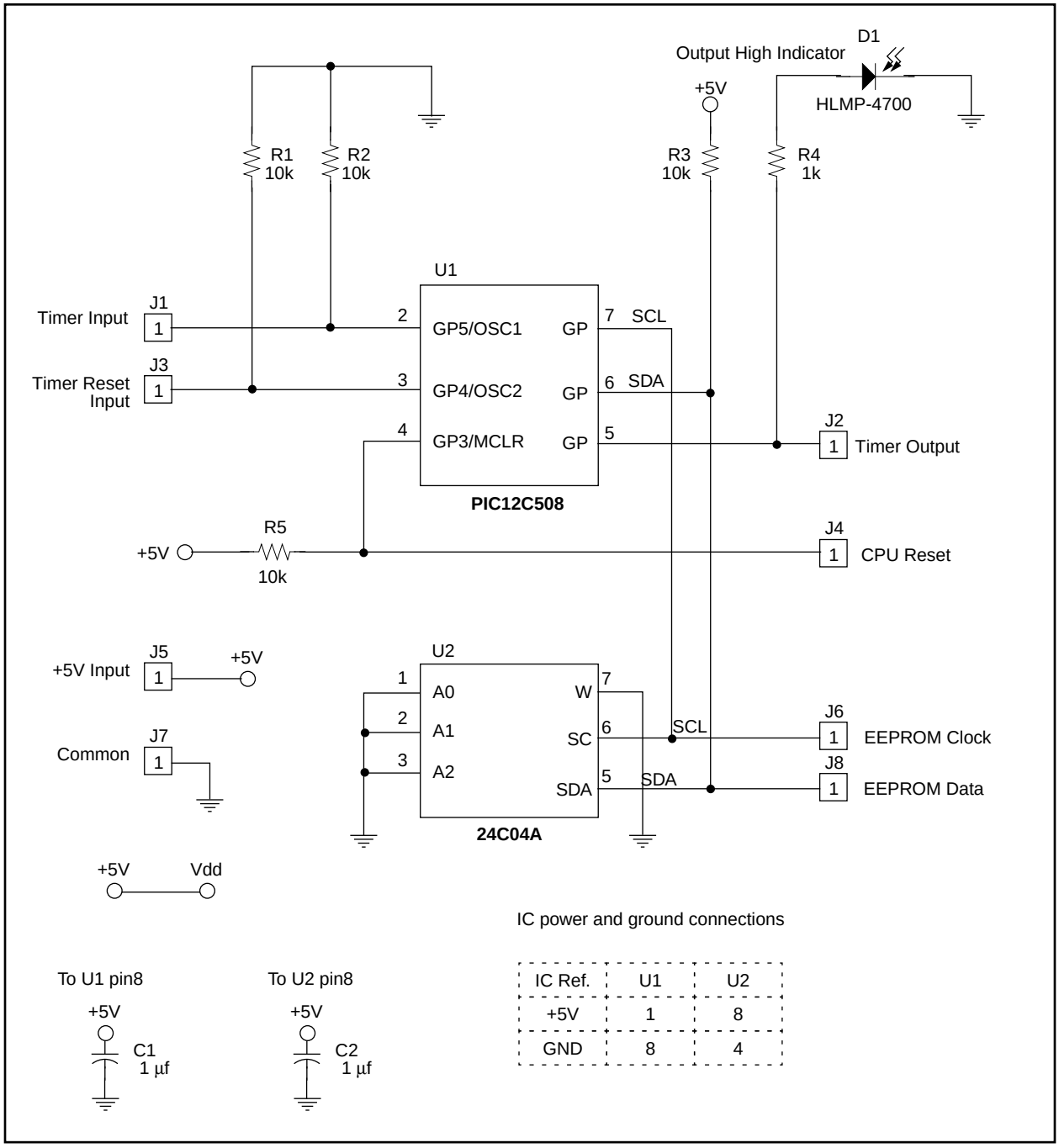


FIGURE 9: SCHEMATIC



MICROCHIP TOOLS USED

Hardware Development Tools:

The PICMASTER emulator with a PIC16C54 POD

Assembler/Compiler Version:

MPLAB 3.22.00 with MPASM 1.50

APPENDIX A: SOURCE CODE

```
The PICMASTER emulator with a 16C54 pod was used to debug the PIC16C54 test version.
Assembler/Compiler version:
MPLAB 3.22.00 development software with MPASM version 1.50
Almost all debugging was done running in simulator mode.
;*************************************************
;*    Swiss Army Timer Project                   *
;*    using the PIC12C508 Version 1.0            *
;*                                               *
;*    Version 1.0 written 5/20/1997              *
;*    by Michael Kirkhart                        *
;*    Release date: 5/30/1997                    *
;*                                               *
;*************************************************
        list         p=12C508        ;specifies 12C508 microcontroller
        list         r=DEC           ;specifies decimal radix as default
        list         x=ON            ;specifies to expand macros in listing
        errorlevel   1               ;print warnings and errors only in list file

;************************
;* General system info  *
;************************
;
;Instruction clock frequency = 4MHz
;Non-branching instruction execution time     = 1 microsecond
;Configuration word settings:  Watchdog timer = ON
;                              Code Protect   = OFF
;                              MCLR pin       = ENABLED
;                              Oscillator     = INTERNAL RC
     __config     0xff6

;************************
;* CPU Register equates *
;************************
     IND0    equ     00              ;indirect file register
     RTCC    equ     01              ;real time clock/counter
     PC      equ     02              ;program counter
     STATUS  equ     03              ;status register
     FSR     equ     04              ;file select register (pointer)
     OSCCAL  equ     05              ;on chip oscillitor calibration register
     GPIO    equ     06              ;general purpose I/O register

;**************************************
;* Status register bit definitions      *
;**************************************

     CARRY   equ     0               ;carry/!borrow flag
     DCARRY  equ     1               ;BCD carry/!borrow flag
     ZERO    equ     2               ;zero flag
     PDOWN   equ     3               ;powerdown flag
     TIMEOUT equ     4               ;watchdog timeout flag


;********************************
;* GPIO bit definitions         *
;********************************

     SCL       equ     0            ;EEPROM serial clock (O)
     SDA       equ     1            ;EEPROM serial data (I/O)
     TMROUT    equ     2            ;Timer output (O)
     TMRRST    equ     4            ;Timer reset (I)
     TMRIN     equ     5            ;Timer input (I)

;**************************************
;* Equates for register files (variables)*
```

```
;***************************************

; configuration block from EEPROM (must be in contiguous memory locations)
     mode          equ     0x07     ;timer mode from EEPROM
     param1L       equ     0x08     ;parameter 1 low byte from EEPROM
     param1H       equ     0x09     ;parameter 1 high byte from EEPROM
     param2L       equ     0x0a     ;parameter 2 low byte from EEPROM
     param2H       equ     0x0b     ;parameter 2 high byte from EEPROM
     tmropt        equ     0x0c     ;option flags from EEPROM

; file registers used in timer handler
     oldRTCC       equ     0x0d     ;keeps track of RTCC value from last run through loop
     rollL         equ     0x0e     ;RTCC rollover counter low byte
     rollH         equ     0x0f     ;RTCC rollover counter high byte

; miscellaneous state variables, target counters, etc.
     inDbnc        equ     0x10     ;Timer input debouncer state
     rstDbnc       equ     0x11     ;Reset input debouncer state
     dbncTmr       equ     0x12     ;Debouce check timer
     tgtL          equ     0x13     ;target counter low byte
     tgtH          equ     0x14     ;target counter high byte
     flags         equ     0x15     ;timer status flags
     modest        equ     0x16     ;state of selected timer mode
     temp          equ     0x17     ;temporary storage register

; file registers used by EEPROM read routines
     eeprom        equ     0x18     ;bit buffer
     datai         equ     0x19     ;data input register
     txbuf         equ     0x1a     ;transmit buffer
     count         equ     0x1b     ;bit counter
     bcount        equ     0x1c     ;byte counter

;***************************************
;* Miscelaneous equates (constants)     *
;***************************************

;GPIO initialization values
     GPINIT        equ     00000010b     ;GPIO initial value (SDA line high, all others low)
     GPTRIS        equ     00111010b     ;GPIO TRIS register initial value
     SDAINP        equ     00111010b     ;GPIO TRIS register value when SDA line to EEPROM
                                         ; needs to be an input
     SDAOUT        equ     00111000b     ;GPIO TRIS register value when SDA line to EEPROM
                                         ; needs to be an output

;mode constants
     ONDLY         equ     0            ;mode 0 = on delay timer
     OFFDLY        equ     1            ;mode 1 = off delay timer
     ONESHT        equ     2            ;mode 2 = non-retriggerable one-shot
     RONESHT       equ     3            ;mode 3 = retriggerable one-shot
     ASTABL        equ     4            ;mode 4 = astable multivibrator
     EASTABL       equ     5            ;mode 5 = astable multivibrator with enable
     COUNTR        equ     6            ;mode 6 = event counter
     COUNTRA       equ     7            ;mode 7 = event counter with auto-reset

;option flags bit values
     LOWIN         equ     0            ;active low input flag
     LOWOUT        equ     1            ;active low output flag
     LOWRST        equ     2            ;active low reset flag
     TRLEDG        equ     3            ;trailing input edge active flag

;flags register bit values
     INPHI         equ     0            ;timer input high
     OUTHI         equ     1            ;timer output high
     RSTHI         equ     2            ;reset input high
     EDGON         equ     3            ;timer input transistion detected
     INPON         equ     4            ;timer input active
```

```
     OUTON         equ     5          ;timer output active
     RSTON         equ     6          ;reset input active

;bit defines for EEPROM routines
     do          equ     6            ;eeprom output bit
     di          equ     7            ;eeprom input bit

;timer constants
     DBNCTM     equ     20            ;debounce timer interval (20 * 256 microseconds =
5.12milliseconds)
     TENTHL     equ     0x87          ;tenth second rollover count (low byte)
     TENTHH     equ     0x01          ;tenth second rollover count (high byte)

;************************
;* Macro definitions    *
;************************

CLC     macro                        ;this macro will clear the C flag
        bcf     STATUS,CARRY
        endm

SEC     macro                        ;this macro will set the C flag
        bsf     STATUS,CARRY
        endm

SCC     macro                        ;used after an instruction that affects the C
        btfsc   STATUS,CARRY         ; flag, this macro will skip the next
        endm                         ; instruction if the C flag is clear

SCS     macro                        ;used after an instruction that affects the C
        btfss   STATUS,CARRY         ; flag, this macro will skip the next
        endm                         ; instruction if the C flag is set

SLT     macro                        ;used after a subtract instruction, this macro
        btfsc   STATUS,CARRY         ; will skip the next instruction if the result
        endm                         ; of the subtraction is < 0

SGE     macro                        ;used after a subtract instruction, this macro
        btfss   STATUS,CARRY         ; will skip the next instruction if the result
        endm                         ; of the subtraction is >= 0

SEQ     macro                        ;used after an instruction that affects the Z
        btfss   STATUS,ZERO          ; flag, this macro will skip the next
        endm                         ; instruction if a result is zero

SNE     macro                        ;used after an instruction that affects the Z
        btfsc   STATUS,ZERO          ; flag, this macro will skip the next
        endm                         ; instruction if a result is non-zero


;************************
;* Start of program     *
;************************
; actual reset vector - instruction at address 0x1ff was movlw XX, where
; XX is the calibration value to be copied into the OSCCAL register

     org         0                   ;start of program memory
     movwf       OSCCAL              ;calibrate on-chip oscillator
     goto        start               ;jump to start of program

;*******************************
;* Subroutines                 *
;* These must be located in the *
;* lower 256 bytes of program   *
;* memory                      *
;*******************************
```

```
;*******************************
;* 24C04 EEPROM read routines   *
;* Modified versions of 24CXX   *
;* routines from Microchip      *
;* AN567                        *
;*******************************

;*************************************************************
;       Start Bit Subroutine                                 *
;       this routine generates a start bit                   *
;       (Low going data line while clock is high)            *
;*************************************************************
bStart
        bsf         GPIO,SDA          ;make sure data is high
        movlw       SDAOUT
        tris        GPIO              ;set data and clock lines for output
        bcf         GPIO,SCL          ;make sure clock is low
        nop
        bsf         GPIO,SCL          ;set clock high
        call        eeDelay           ;wait for a few cycles
        bcf         GPIO,SDA          ;data line goes low during
                                      ; high clock for start bit
        call        eeDelay           ;wait for a few cycles
        bcf         GPIO,SCL          ;start clock train
        call        eeDelay           ;wait for a few cycles
        retlw       0                 ;return from subroutine

;*************************************************************
;       Stop Bit Subroutine                                  *
;       This routine generates a stop bit                    *
;       (High going data line while clock is high)           *
;*************************************************************
bStop
        movlw       SDAOUT           ;
        tris        GPIO             ;set data/clock lines as outputs
        bcf         GPIO,SDA         ;make sure data line is high
        call        eeDelay          ;wait for a few cycles
        bsf         GPIO,SCL         ;set clock high
        call        eeDelay          ;wait for a few cycles
        bsf         GPIO,SDA         ;data goes high while clock high
                                     ;for stop bit
        call        eeDelay          ;wait for a few cycles
        bcf         GPIO,SCL         ;set clock low again
        call        eeDelay          ;wait for a few cycles
        retlw       0                ;return from subroutine

;*************************************************************
;       BITOUT routine takes one bit of data in 'do' and     *
;       transmits it to the serial EE device                 *
;*************************************************************
bitOut
        movlw       SDAOUT           ;set data, clock as outputs
        tris        GPIO             ;
        btfss       eeprom,do        ;check for stat of data bit to xmit
        goto        bitlow           ;
        bsf         GPIO,SDA         ;set data line high
        goto        clkout           ;go toggle the clock
bitlow  bcf         GPIO,SDA         ;output a low bit
clkout  bsf         GPIO,SCL         ;set clock line high
        nop                          ;wait a few cycles
        nop
        bcf         GPIO,SCL         ;return clock line low
        retlw       0                ;return from subroutine
```

```
;*************************************************************
;       eeDelay routine generates a small delay that is       *
;       used by the various EEPROM routines                   *
;*************************************************************
eeDelay
        nop                         ;
        nop                         ;
        retlw       0               ;

;*************************************************************
;       BITIN routine reads one bit of data from the          *
;       serial EE device and stores it in 'di'                *
;*************************************************************
bitIn   bsf         eeprom,di       ;assume input bit is high
        movlw       SDAINP          ;make sdata an input line
        tris        GPIO            ;
        bsf         GPIO,SCL        ;set clock line high
        nop                         ;wait a few cycles
        nop
        btfss       GPIO,SDA        ;read the data bit
        bcf         eeprom,di       ;input bit was low, set 'di' accordingly
        bcf         GPIO,SCL        ;set clock line low
        retlw       0               ;return from subroutine
;
;*********************************************************************
;       Transmit Data Subroutine                                     *
;       This routine takes the byte of data stored in the            *
;       'datao' register and transmits it to the serial EE device.   *
;       It will then send 1 more clock to the serial EE for the      *
;       acknowledge bit.  If the ack bit from the part was low       *
;       then the transmission was sucessful.  If it is high, then    *
;       the device did not send a proper ack bit and the ack         *
;       fail LED will be turned on.                                  *
;*********************************************************************
tx
        movlw       8               ;
        movwf       count           ;

txLoop  bcf         eeprom,do       ;assume bit out is low
        btfsc       txbuf,7         ;is bit out really low?
        bsf         eeprom,do       ;no, set it high
        call        bitOut          ;send the bit to serial EE
        rlf         txbuf           ;rotate txbuf left
        decfsz      count           ;8 bits done?
        goto        txLoop          ;no - go again
        call        bitIn           ;read ack bit
        retlw       0               ;return from subroutine
;
;*************************************************************
;       Receive data Routine                                  *
;       This routine reads one byte of data from the part     *
;       into the 'datai' register.  It then sends a high      *
;       ack bit to indicate that no more data is to be read   *
;*************************************************************
rx
        movlw       8               ;set # bits to 8
        movwf       count
        clrf        datai           ;clear input register
        bcf         STATUS,CARRY    ;make sure carry bit is low
rxLoop  rlf         datai           ;rotate datai 1 bit left
        call        bitIn           ;read a bit
        btfsc       eeprom,di
        bsf         datai,0         ;set bit 0 if necessary
        decfsz      count           ;8 bits done?
        goto        rxLoop          ;no, do another
        retlw       0               ;return from subroutine
```

```
;*******************************
;* MoveP1ToTgt                 *
;*******************************
; This routine moves the timer parameter 1 values read from
; the configuration EEPROM into the target timer file registers.
; This routine is used by the various state handlers in the mode sequencer.

MoveP1ToTgt
        movf        param1L,w       ;initialize
        movwf       tgtL            ; target
        movf        param1H,w       ;  counter
        movwf       tgtH            ;   value
        goto        InitRollOver    ;re-initialize rollover counter
                                    ; (return instruction executed at end of InitRollOver)

;*******************************
;* MoveP2ToTgt                 *
;*******************************
; This routine moves the timer parameter 2 values read from
; the configuration EEPROM into the target timer file registers.
; This routine is used by the various state handlers in the mode sequencer.

MoveP2ToTgt
        movf        param2L,w       ;initialize
        movwf       tgtL            ; target
        movf        param2H,w       ;  counter
        movwf       tgtH            ;   value
        goto        InitRollOver    ;re-initialize rollover counter
                                    ; (return instruction executed at end of InitRollOver)

;*******************************
;* InitRollOver                *
;*******************************
; This routine initializes the rollover counter with the value that
; represents .1 second

InitRollOver
        movlw       TENTHL          ;load
        movwf       rollL           ;  rollover
        movlw       TENTHH          ;    counter
        movwf       rollH           ;
        retlw       0               ;return from subroutine

;*******************************
;* IsTgtZero                   *
;*******************************
; This routine checks to see if the target timer is equal
; to zero.  If it is, it returns with the zero flag in status register set.  Otherwise,
; the zero flag is cleared.  This routine is used by the various state handlers in the
; mode sequencer.

IsTgtZero
        movlw       0               ;is upper byte of target timer
        iorwf       tgtH,w          ; = 0?
        SEQ                         ;if yes, skip
        goto        TgtNotZero      ;if not, target timer not zero - branch
        iorwf       tgtL,w          ;is lower byte of target timer = 0? If so, zero flag in
                                    ; status register will be set
TgtNotZero
        retlw       0               ;return from subroutine (result in zero flag in status reg)

;*******************************
;* DecrTgt                     *
;*******************************
; This routine decrements the target counter.  It is used in the timer handler and the
; counter mode handlers in the mode sequencer.
```

```
DecrTgt
        movlw       1               ;subtract 1 from
        subwf       tgtL            ; tgtL (lower byte of target)
        SGE                         ;did borrow occur?
        subwf       tgtH            ;if so, subtract 1 from tgtH (upper byte of target)
        retlw       0               ;return from subroutine

;*****************************************************
;* doTmrInState                                      *
;* This subroutine runs the debounce state machine   *
;* for the Timer Input.  It is called periodically by*
;* the main program loop.                            *
;*****************************************************
doTmrInState
        movf        inDbnc,w        ;get current timer input debouce state
        addwf       PC              ;add it to the program counter to jump to
                                    ; appropriate state handler
        goto        TInSt0          ;inDbnc = 0 handler
        goto        TInSt1          ;inDbnc = 1 handler
        goto        TInSt2          ;inDbnc = 2 handler
        goto        TInSt3          ;inDbnc = 3 handler

;TimerIn state = 0 handler
TInSt0  btfss       GPIO,TMRIN      ;is Timer input high?
        goto        TInEnd          ;if no - stay in state 0
        incf        inDbnc          ;if yes - move to state 1
        goto        TInEnd          ;go to TInEnd

;TimerIn state = 1 handler
TInSt1  btfss       GPIO,TMRIN      ;is Timer input high?
        goto        TInSt1a         ;if no - go to TInSt1a
        incf        inDbnc          ;if yes - move to state 2
        goto        TInEnd          ;go to TInEnd
TInSt1a clrf        inDbnc          ;move back to state 0
        goto        TInEnd          ;go to TInEnd

;TimerIn state = 2 handler
TInSt2  btfsc      GPIO,TMRIN       ;is Timer input low?
        goto       TInEnd           ;if no - go to TInEnd (stay in state 2)
        incf       inDbnc           ;if yes - move to state 3
        goto       TInEnd           ;go to TInEnd

;TimerIn state = 3 handler
TInSt3  btfsc      GPIO,TMRIN       ;is Timer input low?
        goto       TInSt3a          ;if no - go to TInSt3a
        clrf       inDbnc           ;if yes - move to state 0
        goto       TInEnd           ;go to TInEnd
TInSt3a decf       inDbnc           ;move back to state 2
        goto       TInEnd           ;go to TInEnd

TInEnd  retlw      0                ;return from subroutine

;*****************************************************
;* doRstInState                                      *
;* This subroutine runs the debounce state machine   *
;* for the Reset Input.  It is called periodically by*
;* the main program loop.                            *
;*****************************************************
doRstInState
        movf       rstDbnc,w        ;get current reset input debounce state
        addwf      PC               ;add it to the program counter to jump to the
                                    ; appropriate state handler
        goto       RstSt0           ;rstDbnc = 0 state handler
        goto       RstSt1           ;rstDbnc = 1 state handler
        goto       RstSt2           ;rstDbnc = 2 state handler
        goto       RstSt3           ;rstDbnc = 3 state handler
```

```
;ResetIn state = 0 handler
RstSt0  btfss      GPIO,TMRRST      ;is timer reset input high?
        goto       RstEnd           ;if no - stay in state 0
        incf       rstDbnc          ;if yes - move to state 1
        goto       RstEnd           ;go to RstEnd

RstSt1  btfss      GPIO,TMRRST      ;is timer reset input high?
        goto       RstSt1a          ;if no - go to RstSt1a
        incf       rstDbnc          ;if yes - move to state 2
        goto       RstEnd           ;go to RstEnd

RstSt1a clrf       rstDbnc          ;move back to state 0
        goto       RstEnd           ;go to RstEnd

RstSt2  btfsc      GPIO,TMRRST      ;is timer reset input low?
        goto       RstEnd           ;if no - go to RstEnd (stay in state 2)
        incf       rstDbnc          ;if yes - move to state 3
        goto       RstEnd           ;go to RstEnd

RstSt3  btfsc     GPIO,TMRRST       ;is timer reset input low?
        goto      RstSt3a           ;if no - go to RstSt3a
        clrf      rstDbnc           ;if yes - move to state 0
        goto      RstEnd            ;go to RstEnd

RstSt3a decf      rstDbnc           ;move back to state 2
        goto      RstEnd            ;go to RstEnd

RstEnd  retlw     0                 ;return from subroutine

;;doSeqState
;*******************************************************
;* doSeqState                                          *
;* This subroutine runs the main sequencer state       *
;* machine.  It is called periodically by              *
;* the main program loop.                              *
;*******************************************************
doSeqState
        movf      mode,w            ;get configuration mode value
        addwf     PC                ;add it to the program counter to jump to the
                                    ; appropriate mode handler
        goto     doOnDelay          ;on delay timer handler
        goto     doOffDelay         ;off delay timer handler
        goto     doOneShot          ;one shot handler
        goto     doROneShot         ;retriggerable one shot handler
        goto     doAstable          ;astable multivibrator handler
        goto     doEAstable         ;enabled astable multivibrator handler
        goto     doCounter          ;event counter handler
        goto     doACounter         ;auto resetting event counter handler

;*************************************
;* On Delay mode handler             *
;*************************************
doOnDelay
        movf     modest,w           ;get current sequencer state value
        addwf    PC                 ;add it to the program counter to jump to the
                                    ; appropriate mode handler
        goto     OnDly0             ;On delay state 0 handler
        goto     OnDly1             ;On delay state 1 handler
        goto     OnDly2             ;On delay state 2 handler

;*************************************
;* Off Delay mode handler            *
;*************************************
doOffDelay
        movf     modest,w          ;get current sequencer state value
```

```
        addwf    C                ;add it to the program counter to jump to the
                                  ; appropriate mode handler
        goto     OffDly0          ;Off delay state 0 handler
        goto     OffDly1          ;Off delay state 1 handler
        goto     OffDly2          ;Off delay state 2 handler

;*************************************************
;* Non-retriggerable one-shot mode handler      *
;*************************************************
doOneShot
        movf     modest,w         ;get current sequencer state value
        addwf    PC               ;add it to the program counter to jump to the
                                  ; appropriate mode handler
        goto     OneShot0         ;one shot state 0 handler
        goto     OneShot1         ;one shot state 1 handler

;*************************************************
;* Retriggerable one-shot mode handler          *
;*************************************************
doROneShot
       movf      modest,w         ;get current sequencer state value
       addwf     PC               ;add it to the program counter to jump to the
                                  ; appropriate mode handler
       goto      OneShot0         ;retriggerable one shot state 0 handler
                                  ; (same as non-retriggerable
                                  ; one shot state 0 handler)
       goto     ROneShot1         ;retriggerable one shot state 1 handler

;*************************************************
;* Astable mode handler                         *
;*************************************************
doAstable
       movf     modest,w          ;get current sequencer state value
       addwf    PC                ;add it to the program counter to jump to the
                                  ; appropriate mode handler
       goto     Astable0          ;astable state 0 handler
       goto     Astable1          ;astable state 1 handler
       goto     Astable2          ;astable state 2 handler

;*************************************************
;* Enabled astable mode handler                 *
;*************************************************
doEAstable
       movf     modest,w         ;get current sequencer state value
       addwf    PC               ;add it to the program counter to jump to the
                                 ; appropriate mode handler
       goto     EAstable0        ;astable state 0 handler
       goto     EAstable1        ;astable state 1 handler
       goto     EAstable2        ;astable state 2 handler

;*************************************************
;* Event counter mode handler                   *
;*************************************************
doCounter
      movf      modest,w        ;get current sequencer state value
      addwf     PC              ;add it to the program counter to jump to the
                                ; appropriate mode handler
      goto      Counter0        ;event counter state 0 handler
      goto      Counter1        ;event counter state 1 handler
      goto      Counter2        ;event counter state 2 handler

;*************************************************
;* Auto-reset event counter mode handler        *
;*************************************************
doACounter
      movf      modest,w        ;get current sequencer state value
```

```
      addwf     PC              ;add it to the program counter to jump to the
                                ; appropriate mode handler
      goto      Counter0        ;auto-reset event counter state 0 handler
                                ; (same as event counter state 0 handler)
      goto      Counter1        ;auto-reset event counter state 1 handler
                                ;(same as event counter state 1 handler)
      goto      ACounter2       ;auto-reset event counter state 2 handler

;*********************************************
;* Main sequencer subroutine return point       *
;*********************************************
seqEnd     retlw     0          ;return from subroutine

; on delay state 0 - waiting for input to go active
OnDly0
           bcf     flags,OUTON      ;clear timer output on flag
           btfsc   flags,INPON      ;is timer input active?
           goto    OnDly0a          ;if yes, go to OnDly0a
           goto    seqEnd           ;go to seqEnd
OnDly0a    incf    modest           ;move to mode state 1
           call    MoveP1ToTgt      ;initialize target counter value
           goto    seqEnd           ;go to seqEnd

; on delay state 1 - input active, waiting for target timer to time out
OnDly1
           bcf     flags,OUTON     ;clear timer output on flag
           btfsc   flags,INPON     ;is timer input still active?
           goto    OnDly1a         ;if yes, go to OnDly1a
           clrf    modest          ;otherwise, go back to state 0
           goto    seqEnd          ;go to seqEnd

OnDly1a    call    IsTgtZero       ;is target timer = 0?
           SEQ                     ;
           goto    seqEnd          ;if not, go to seqEnd
           incf    modest          ;otherwise, move to mode state 2
           goto    seqEnd          ;go to seqEnd

OnDly2
           bsf     flags,OUTON     ;set timer output on flag
           btfsc   flags,INPON     ;is timer input still active?
           goto    seqEnd          ;if yes, go to OnDly2a
           clrf    modest          ;otherwise, go back to state 0
           goto    seqEnd          ;go to seqEnd

OffDly0
           bcf     flags,OUTON     ;clear timer output flag
           btfsc   flags,INPON     ;is timer input active?
           goto    OffDly0a        ;if yes, go to OffDly0a
           goto    seqEnd          ;otherwise, goto seqEnd

OffDly0a
           incf    modest          ;move to mode state 1
           goto    seqEnd          ;go to seqEnd

OffDly1
           bsf     flags,OUTON     ;set timer output on flag
           btfsc   flags,INPON     ;is timer input still active?
           goto    seqEnd          ;if yes, go to seqEnd
           incf    modest          ;move to mode state 2
           call    MoveP1ToTgt     ;initialize target counter value
           goto    seqEnd          ;go to seqEnd

OffDly2
           bsf     flags,OUTON     ;set timer output on flag
           btfsc   flags,INPON     ;is timer input active again?
           goto    OffDly2a        ;if so, go to OffDly2a
```

```
           call    IsTgtZero       ;is target timer = 0?
           SEQ                     ;
           goto     seqEnd          ;if not, go to seqEnd
           clrf     modest          ;otherwise, move to mode state 0
           goto     seqEnd          ;go to seqEnd

OffDly2a
           decf     modest          ;move back to mode state 1
           goto     seqEnd          ;go to seqEnd

OneShot0
           bcf     flags,OUTON     ;clear timer output on flag
           btfsc   flags,EDGON     ;has an active edge been detected?
           goto    OneShot0a       ;if yes, go to OnDly0a
           goto    seqEnd          ;go to seqEnd

OneShot0a
           incf    modest          ;move to mode state 1
           call    MoveP1ToTgt     ;initialize target counter value
           goto    seqEnd          ;go to seqEnd

OneShot1
           bsf     flags,OUTON     ;set timer output on flag
           call    IsTgtZero       ;is target timer = 0?
           SEQ                     ;
           goto    seqEnd          ;if not, go to seqEnd
           clrf    modest          ;otherwise, move to mode state 0
           goto    seqEnd          ;go to seqEnd

ROneShot1
           bsf     flags,OUTON     ;set timer output on flag
           btfsc   flags,EDGON     ;have we been retriggered?
           goto    ROneShot1a      ;yes we have - go to ROneShot1a
           call    IsTgtZero       ;is target timer = 0?
           SEQ                     ;
           goto    seqEnd          ;if not, go to seqEnd
           clrf    modest          ;otherwise, move to mode state 0
           goto    seqEnd          ;go to seqEnd

ROneShot1a
           call    MoveP1ToTgt     ;initialize target counter value
           goto    seqEnd          ;go to seqEnd

Astable0
           bcf     flags,OUTON     ;clear timer output on flag
           incf    modest          ;move to mode state 1
           call    MoveP1ToTgt     ;initialize target counter value
           goto    seqEnd          ;go to seqEnd

Astable1
           bsf     flags,OUTON     ;set timer output on flag
           call    IsTgtZero       ;is target timer = 0?
           SEQ                     ;
           goto    seqEnd          ;if not, go to seqEnd
           incf    modest          ;otherwise, move to mode state 2
           call    MoveP2ToTgt     ;initialize target counter value
           goto    seqEnd          ;go to seqEnd

Astable2
           bcf     flags,OUTON     ;clear timer output on flag
           call    IsTgtZero       ;is target timer = 0?
           SEQ                     ;
           goto    seqEnd          ;if not, go to seqEnd
           movlw   1               ;otherwise, move
           movwf   modest          ; to mode state 0
           call    MoveP1ToTgt     ;initialize target counter value
```

```
           goto    seqEnd          ;go to seqEnd

EAstable0
           bcf     flags,OUTON     ;clear timer output on flag
           call    MoveP1ToTgt     ;initialize target counter value
           btfsc   flags,INPON     ;is timer input on?
           incf    modest          ;if yes - move to mode state 1
           goto    seqEnd          ;go to seqEnd

EAstable1
           btfss   flags,INPON     ;is timer input still on?
           goto    EAstable1a      ;if not - go to EAstable1a
           bsf     flags,OUTON     ;set timer output on flag
           call    IsTgtZero       ;is target timer = 0?
           SEQ                     ;
           goto    seqEnd          ;if not, go to seqEnd
           incf    modest          ;otherwise, move to mode state 2
           call    MoveP2ToTgt     ;initialize target counter value
           goto    seqEnd          ;go to seqEnd

EAstable1a
           bcf     flags,OUTON     ;clear timer output on flag
           clrf    modest          ;go back to mode state 0
           goto    seqEnd          ;

EAstable2
           bcf     flags,OUTON     ;clear timer output on flag
           btfss   flags,INPON     ;is timer input still on?
           goto    EAstable2a      ;if not - go to EAstable2a
           call    IsTgtZero       ;is target timer = 0?
           SEQ                     ;
           goto    seqEnd          ;if not, go to seqEnd
           movlw   1               ;otherwise, move
           movwf   modest          ; to mode state 0
           call    MoveP1ToTgt     ;initialize target counter value
           goto    seqEnd          ;go to seqEnd

EAstable2a
           clrf    modest          ;go back to mode state 0
           goto    seqEnd          ;go to seqEnd

Counter0
           bcf     flags,OUTON     ;clear timer output on flag
           call    MoveP1ToTgt     ;initialize target counter value
           incf    modest          ;move to state 1
           goto    seqEnd          ;go to seqEnd

Counter1
           bcf     flags,OUTON     ;clear timer output on flag
           btfsc   flags,RSTON     ;is reset input active?
           goto    Counter1a       ;yes it is - go to Counter1a
           btfss   flags,EDGON     ;have we detected an active input edge?
           goto    Counter1b       ;no we have not - go to Counter1b
           call    DecrTgt         ;decrement the target counter
           call    IsTgtZero       ;is target counter = 0?
           SEQ                     ;
           goto    Counter1b       ;nope - go to Counter1b
           incf    modest          ;move to mode state 2
           goto    seqEnd          ;go to seqEnd

Counter1a
           call    MoveP1ToTgt     ;re-initialize target counter value

Counter1b
           goto    seqEnd          ;go to seqEnd
```

```
Counter2
          bsf     flags,OUTON     ;set timer output on flag
          btfss   flags,RSTON     ;is reset input active?
          goto    seqEnd          ;if not, go to seqEnd
          call    MoveP1ToTgt     ;initialize target counter value
          decf    modest          ;move back to mode state 1
          goto    seqEnd          ;go to seqEnd

ACounter2
          bsf     flags,OUTON     ;set timer output on flag
          btfsc   flags,EDGON     ;have we detected an active input edge?
          goto    ACounter2a      ;yes we have - go to ACounter2a
          btfss   flags,RSTON     ;is reset input active?
          goto    seqEnd          ;if not, go to seqEnd
          call    MoveP1ToTgt     ;initialize target counter value
          decf    modest          ;move back to mode state 1
          goto    seqEnd          ;go to seqEnd

; auto reset handler (make sure configuration program does not allow terminal
; count value less than 2)
ACounter2a
          bcf     flags,OUTON     ;clear timer output on flag
          call    MoveP1ToTgt     ;initialize target counter
          call    DecrTgt         ;decrement target counter
          decf    modest          ;move back to mode state 1
          goto    seqEnd          ;go to seqEnd

;********************
;* Start of program *
;********************
start     movlw     11001111b     ;disable wake-up on GPIO pin change, disable weak
                                  ; pullups, use internal clock for RTCC (positive edge),
          option                  ;  assign prescaler to WDT, set prescaler to 128
          movlw     GPTRIS        ;set GPIO
          tris      GPIO          ; TRIS register
          movf      RTCC,w        ;initialize
          movwf     oldRTCC       ; oldRTCC value
          call      InitRollOver  ;initialize rollover counter
          clrf      tgtL          ;initialize
          clrf      tgtH          ; target count register
          movlw     DBNCTM        ;initialize
          movwf     dbncTmr       ; debounce timer
          clrf     inDbnc         ;set initial state of timer input debounce
                                  ; state machine
          clrf     rstDbnc        ;set initial state of reset input debouce
                                  ; state machine
          clrf     flags          ;clear flags register
          clrf     modest         ;clear timer mode state

;read the timer configuration from the EEPROM and copy it to the
;block of file registers used to hold the configuration data
readEE
          movlw     6
          movwf     bcount        ;set number of bytes to read as 6
          movlw     mode          ;set FSR to point to 1st address into
          movwf     FSR           ; configuration memory block
          call      bStart        ;generate start bit
          movlw     10100000b     ;set slave address and write mode and
          movwf     txbuf         ; copy into transmit buffer
          call      tx            ;  and send it
          movlw     0             ;copy read start address
          movwf     txbuf         ; into transmit buffer
          call      tx            ;  and send it
          call      bStart        ;generate start bit
          movlw     10100001b     ;get slave address and read mode
          movwf     txbuf         ;into transmit buffer
```

```
          call      tx            ;and transmit it
                                  ;
rbyte     call      rx            ;read 1 byte from device
          movf      datai,w       ;copy byte into register
          movwf     IND0          ; pointed to by FSR
          incf      FSR           ;increment the FSR
          decfsz    bcount        ;are all 6 bytes read?
          goto      lowack        ;no, send low ack and do another
          bsf       eeprom,do     ;yes, send high ack bit
          call      bitOut        ;to stop transmission
          call      bStop         ;and send a stop bit
          vgoto     infLoop       ;all done - go to the main program loop

lowack    bcf       eeprom,do     ;send low ack bit
          call      bitOut        ;to continue transmission
          goto      rbyte         ;and read another byte

;*********************************************
;* Start of main program loop                *
;*********************************************
infLoop

;handle the timer
HandleTimer
          movf      RTCC,w        ;get the current RTCC counter value
          movwf     temp          ; and temporarily save it
          subwf     oldRTCC,w     ;has the RTCC
          SLT                     ; rolled over?
          goto      decRoll       ;if yes - decrement the rollover counter
          goto      tmrEnd        ;else goto tmrEnd

decRoll
          decf      dbncTmr       ;decrement the debounce timer
          movlw      1            ;subtract 1
          subwf      rollL        ; from rollL (low byte)
          SGE                     ;skip if borrow did not occur
          subwf      rollH        ;otherwise, subtract 1 from rollH (high byte)
          movlw      0            ;is rollover
          iorwf      rollL,w      ; counter
          SNE                     ;  =
          iorwf      rollH,w      ;   0?
          SEQ                     ;
          goto      tmrEnd        ;if no - goto tmrEnd
          call      InitRollOver  ;re-initialize rollover counter

;handle .1 second target counter if not in an event counter mode
          movlw      COUNTR       ;are we in an
          subwf      mode,w       ; event counter
          SLT                     ;  mode?
          goto       tmrEnd       ;if yes - goto tmrEnd
          call       IsTgtZero    ;otherwise, is target counter = 0?
          SEQ                     ;if it is, skip
          call       DecrTgt      ;otherwise, decrement the target counter

tmrEnd
          movf       temp,w       ;update old RTCC value
          movwf      oldRTCC      ; from temp register

;handle inputs
handleIn
          bcf        flags,EDGON  ;reset the input edge flag
; handle debouncing of the inputs
          clrw                    ;is debounce timer
          iorwf      dbncTmr,w    ; = 0?
          SEQ                     ;if yes, skip next instruction
          goto       doSequencer  ;otherwise, go to doSequencer
```

```
; handle TimerIn input debounce state machine
          call      doTmrInState  ;run timer input debounce state machine
          movlw     2             ;is current state
          subwf     inDbnc,w      ; >= 2?
          SGE                     ;if so, skip next instruction
          goto      TInEnd3       ;goto TInEnd3 (input low)

;input high
          btfsc     flags,INPHI   ;was timer input previously low?
          goto      TInEnd1       ;if not - go to TInEnd1
          btfsc     tmropt,TRLEDG ;are we configured for rising edge detect?
          goto      TInEnd1       ;if not - go to TInEnd1
          bsf       flags,EDGON   ;set the edge detect flag

TInEnd1   bsf       flags,INPHI   ;set timer input high flag
          btfsc     tmropt,LOWIN  ;is input configured to be active low?
          goto      TInEnd2       ;if yes - go to TInEnd2
          bsf       flags,INPON   ;if no - set timer input active flag
          goto      doRstIn       ;go to doRstIn
TInEnd2   bcf       flags,INPON   ;clear timer input active flag
          goto      doRstIn       ;go to doRstIn

;input low
TInEnd3   btfss     flags,INPHI   ;was timer input previously high?
          goto      TInEnd4       ;if not - go to TInEnd3
          btfss     tmropt,TRLEDG ;are we configured for trailing edge detect?
          goto      TInEnd4       ;if not - go to TInEnd3
          bsf       flags,EDGON   ;set the edge detect flag
TInEnd4   bcf       flags,INPHI   ;clear timer input high flag
          btfss     tmropt,LOWIN  ;is input configured to be active low?
          goto      TInEnd5       ;if no - go to TInEnd5
          bsf       flags,INPON   ;if yes - set timer input active flag
          goto      doRstIn       ;go to doRstIn
TInEnd5   bcf       flags,INPON   ;clear timer input active flag

;handle TimerReset input
doRstIn
          call      doRstInState  ;run reset input state machine
          movlw     2             ;is current state
          subwf     rstDbnc,w     ; >= 2?
          SGE                     ;if so, skip next instruction
          goto      RstEnd2       ;if no - go to RstEnd2

;reset input high
          bsf       flags,RSTHI   ;clear Timer Reset high flag
          btfsc     tmropt,LOWRST ;is reset input configured to be active low?
          goto      RstEnd1       ;if yes - go to RstEnd1
          bsf       flags,RSTON   ;set reset input on flag
          goto      dbncEnd       ;go to dbncEnd
RstEnd1   bcf       flags,RSTON   ;clear reset input on flag
          goto      dbncEnd       ;go to dbncEnd

;reset input low
RstEnd2   bcf       flags,RSTHI   ;clear Timer Reset high flag
          btfss     tmropt,LOWRST ;is reset input configured to be active low?
          goto      RstEnd3       ;if no - go to RstEnd3
          bsf       flags,RSTON   ;set reset input on flag
          goto      dbncEnd       ;go to dbncEnd
RstEnd3   bcf       flags,RSTON   ;clear reset input on flag

dbncEnd   movlw     DBNCTM        ;reset
          movwf     dbncTmr       ; debounce timer

;handle sequencer state
doSequencer
```

```
          call     doSeqState     ;run mode sequencer state machine
          btfss    flags,OUTON    ;is timer output on?
          goto     outOff         ;nope - go to outOff

; timer output on
outOn
          btfsc     tmropt,LOWOUT ;is timer output configured to be active low?
          goto      outOn1        ;if yes - go to outOn1
          bsf       GPIO,TMROUT   ;set timer output high
          goto      outEnd        ;go to dbncEnd
outOn1    bcf       GPIO,TMROUT   ;set timer output low
          goto      outEnd        ;go to outEnd

; timer outout off
outOff    btfss     tmropt,LOWOUT ;is timer output configured to be active low?
          goto      outOff1       ;if no - go to outOff1
          bsf       GPIO,TMROUT   ;set timer output high
          goto      outEnd        ;go to dbncEnd
outOff1   bcf       GPIO,TMROUT   ;set timer output low

outEnd
          clrwdt                   ;reset the watchdog timer
          goto     infLoop         ;do it again! and again! and again!


;****************
;* Reset vector *
;****************
; For 12C508, this location contains movlw XX, where XX is the calibration value
; for the on-board oscillator - thus the real reset vector is at address 0
     org      0x1ff        ;location of "reset" vector


;******************
;* End of program *
;******************
     end
```

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602-786-7200 Fax: 602-786-7277
Technical Support: 602 786-7627
Web: http://www.microchip.com

Atlanta
Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 972-991-7177 Fax: 972-991-8588

Dayton
Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 714-263-1888 Fax: 714-263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516-273-5305 Fax: 516-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong
Microchip Asia Pacific
RM 3801B, Tower Two
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

India
Microchip Technology Inc.
India Liaison Office
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-4036 Fax: 91-80-559-9840

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Shanghai
Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hong Qiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700
Fax: 86 21-6275-5060

Singapore
Microchip Technology Taiwan
Singapore Branch
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan, R.O.C
Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2-717-7175 Fax: 886-2-545-0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44-1628-851077 Fax: 44-1628-850259

France
Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
2 Rue du Buisson aux Fraises
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 Müchen, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-39-6899939 Fax: 39-39-6899883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa 222 Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

7/29/97