AN1950

Configuring and Using the MCP795WXX SPI RTCC for Basic Timekeeping Based on a PIC18

Author: *Alexandru Valeanu*
Microchip Technology Inc.

INTRODUCTION

An increasing number of applications require a Real-Time Clock/Calendar (RTCC) device. MCP795WXX is a feature-rich SPI RTCC that incorporates EEPROM, SRAM, unique ID time-stamp, Watchdog Timer and event-detect module. This application note describes how to configure and use a Microchip SPI RTCC, based on an electronic watch with display and time/date set-up through two push buttons.

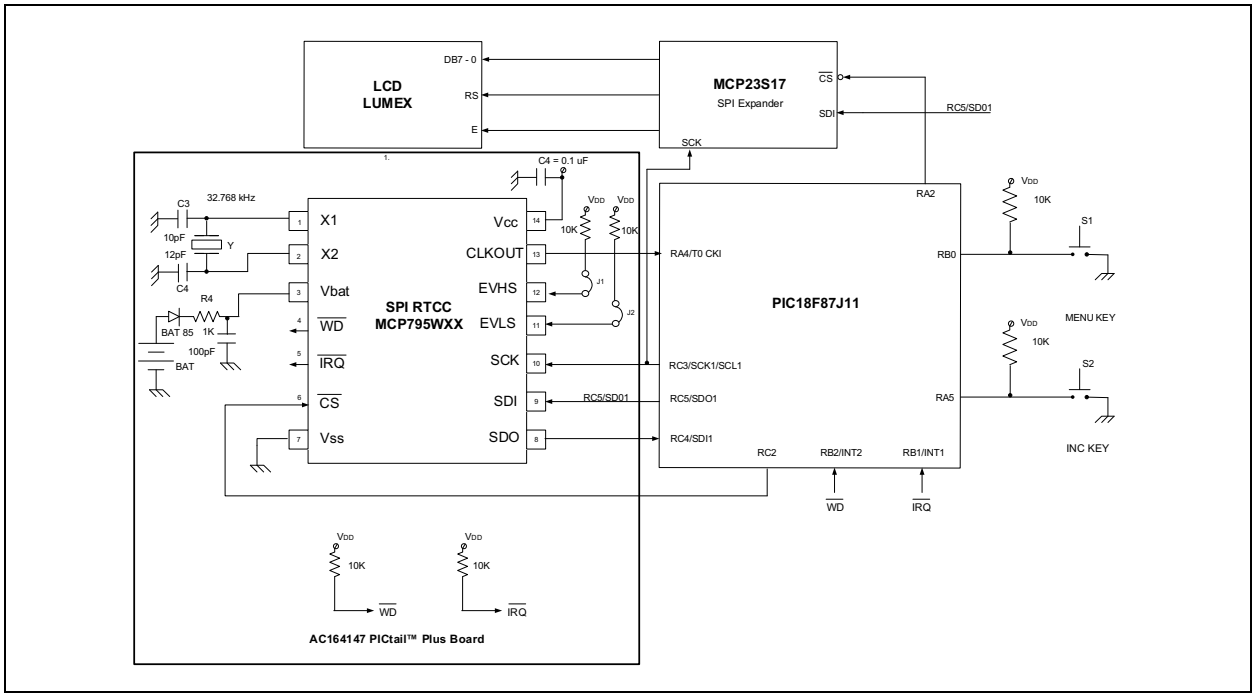
FEATURES OF THE RTCC

- Real-Time Clock/Calendar:
  - Hours, minutes, seconds, hundredths of seconds, day of week, month, and year
  - Support for leap year
- Leap Year Calculation up to 2399
- Time-Stamp Function
- 2 Kbit (256 x 8) EEPROM Memory
- Low-Power CMOS Technology
- 64-Byte x 8 Organization Battery Backed SRAM
- Input for External Battery Backup
- On-Board Crystal Oscillator for RTCC Functions:
  - Battery operated when VCC removed
- Programmable Clock-out Function
- Two Programmable Alarms
- 64-Bit Unique ID in Protected Area:
  - Support EUI-48/64
- Programmable Watchdog Timer
- On-Board Event Detection:
  - Dual configurable inputs
  - High-speed digital event detection on the 1st, 4th, 16th or 32nd event (glitch filter)
  - Low-speed detection with programmable debounce time
- On-Chip Digital Trimming/Calibration

SCHEMATIC

The schematic includes a PIC18 Explorer demo board and the AC164147 SPI RTCC PICtail™ daughter board, as shown in Figure 1.

FIGURE 1: SCHEMATIC



The hardware modules used on the demo board are:

- LCD character module
- Two push buttons
- AC164147 SPI RTCC PICtail™ daughter board

To access the LCD through a minimum of pins, the SPI on the MSSP1 module is used, in conjunction with a 16-bit I/O expander with SPI interface (MCP23S17). The two on-board push buttons are S1 and S2, connected to RB0, RA5 GPIOs. The SPI RTCC is part of the RTCC PICtail evaluation board and is directly connected to the MSSP1 module of the MCU. Another necessary connection is between the CLKOUT signal of the RTCC and RA4 (T0CKI), the clock input of TMR0. The RTCC is programmed to offer a square wave of 1 Hz on CLKOUT. TMR0 is programmed as counter and is initialized at 0xFFFF, in order to give a software interrupt at every second. The SPI connections between the SPI RTCC and the MCU (SDI, SDO, SCK, $\overline{CS}$) are not open-drain and, accordingly, do not use pull-up resistors. Secondary connections are: $\overline{WD}$, $\overline{IRQ}$, EVHS and EVLS. They are open-drain outputs or inputs and need related pull-up resistors. The CLKOUT signal goes to RA4/T0CKI without a pull-up and can be programmed to offer several frequencies: 1 Hz, 4 kHz, 16 kHz and 32 kHz.

The AC164147 RTCC PICtail daughter board has two other components:

- a 32.768 Hz crystal driving the internal clock of the RTCC
- a 3-volt battery sustaining the RTCC when V_{DD} is not present on the demo board

DETAILS ABOUT IMPLEMENTATION

The application is performed on a PIC18 Explorer demo board on which a PIC18F87J11 MCU is mounted. The code is written in C using MPLAB® X V3.55 and the XC8 compiler v1.34.

It implements an electronic watch (based on the MCP795WXX SPI RTCC), displaying the six basic time/date variables on the on-board LCD. It includes a setup sequence, which sets the same six time/date variables, using the two push buttons of the evaluation board (S1 = MENU key, S2 = INCREMENT key). At the same time, the code shows the customers how to configure and use the timekeeping registers.

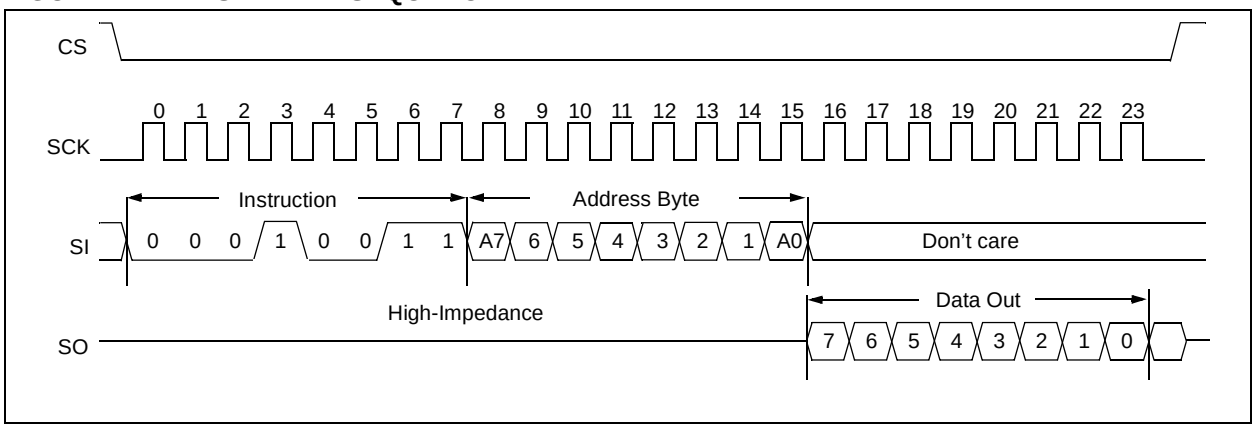
FUNCTIONAL DESCRIPTION

MCP795WXX is an SPI slave device, connected to the SPI bus of the PIC18 MCU (MSSP1 module). The Chip Select of the RTCC ($\overline{CS}$ = pin 6) is controlled by the RC2 GPO pin.

As stated in the MCP795XXX data sheet (DS20002280), for reads, the part is selected by pulling $\overline{CS}$ low, then the 8-bit READ instruction (13h) is transmitted to the MCP795WXX followed by the 8-bit address (A7 through A0). After the correct READ instruction and address are sent, the data stored in the memory at the selected address is shifted out on the SO pin. The data stored in the memory at the next address can be read sequentially by continuing to provide clock pulses. The internal Address Pointer is automatically incremented to the next higher address after each byte of data is shifted out.

As the RTCC registers are separate from the SRAM array, when reading the RTCC registers set, the address will wrap back to the start of the RTCC registers. Also when an address within the SRAM array is loaded, the internal Address Pointer will wrap back to the start of the SRAM array. The READ instruction can be used to read the arrays indefinitely by continuing to clock the device. The read operation is terminated by raising the $\overline{CS}$ pin (Figure 2).

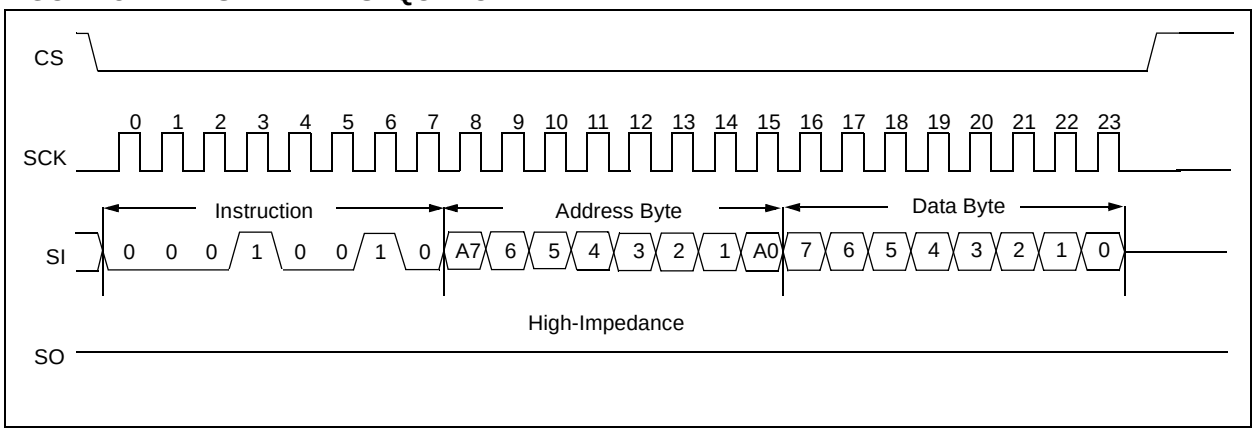
FIGURE 2: SPI READ SEQUENCE



For writes, as the RTCC and SRAM registers do not require the WREN sequence like the EEPROM, the user may proceed by setting the $\overline{CS}$ low, issuing the `WRITE` instruction (12h), followed by the address, and then the data to be written. As no write cycle is required for the RTCC and SRAM registers, the entire array can be written in a single command.

For the data to be actually written to the array, the $\overline{CS}$ must be brought high after a whole byte has been clocked in. If $\overline{CS}$ is brought high at any other time, the last byte will not be written. Refer to Figure 3 for more detailed illustrations on the write sequence.

FIGURE 3: SPI WRITE SEQUENCE



APPLICATION DESCRIPTION

This application performs an electronic watch. Its two main functions are:

- display of the six time/date variables (year, month, date, hour, minutes, seconds) using the interrupts of the microcontroller (this operation is performed on the on-board LCD; the format is 24 hours).
- setup of the above variables using the two on-board push buttons: S1 = MENU key, S2 = INCREMENT key. The real-time display of the time/date variables is performed as long as the MENU key (S1) is not pressed (the action of the INCREMENT key (S2) has no effect on the watch continuously displaying the time and the date).

Pressing the MENU key will start the setup menu, disabling the interrupts. The menu is covered once in the following order: year, month, date, hour, minutes and seconds. Going from one variable to another is performed through the MENU key, and incrementing a variable is performed through the INCREMENT key. The last action of the MENU key exits the setup menu. Accordingly, to correct a possible setup error, the setup menu must be re-entered. The upper limits of every variable are:

- year = (23) 99
- month = 12
- date = (always) 31
- hour = 23 (24 hours format)
- minutes = 59
- seconds = 59

Entering the setup menu will not stop the oscillator of the RTCC. At the end of the setup, the time/date variables are updated. If the user enters the Time Setup mode, all variables are written to the RTCC in the end of the sequence, even if no variables are changed. In this case, when exiting the menu, the watch will resume counting from the point where the setup was entered.

FIRMWARE DESCRIPTION

The project follows the standard multi-file philosophy. All necessary drivers can be found in the related libraries.

Delay Drivers (`delay_drivers.h`)

- LCD functions – Since the controller of the LCD needs some delays to process commands, a few auxiliary delays were created based on TMR1:
  - dly39us()
  - dly43us()
  - dly1_5ms()
- Long delays – Used for the keyboard debounce or as general purpose. They are based on TMR3 and include:
  - dly5ms()
  - dly100ms()
  - dly1s()

LCD Drivers (`lcd_drivers.h`)

Basic LCD function – They handle data, commands and strings written into the LCD. The three drivers used are defined below:

Basic LCD Functions

```
void wrcmnd_lcd (unsigned char cmnd_lcd)            ;   // writes a command in the LCD
void wrdata_lcd (unsigned char data_lcd)            ;   // writes a data byte in the LCD
void wrstr_lcd (const unsigned char *str_lcd)       ;   // writes a string
                                                        // the string is stored in the Flash
```

High-level LCD functions – They initialize or print date/time to the LCD.

The library also includes time and date global variables: `sec, min, hr, day, dat, mon, yr`.

High-Level LCD Functions

```
void clr_lcd(void)                ; void ini_lcd(void)      ; // initialization of the LCD
void sec_to_lcd(void)             ; void min_to_lcd(void)   ;
void hr_to_lcd(void)            ; // time printed to the LCD
void dat_to_lcd(void)             ; void mon_to_lcd(void)   ;
void  yr_to_lcd(void)           ; // date printed to the LCD
void back_lcd(unsigned char pos); // turns back cursor with 'pos' positions
void del_lcd(unsigned char pos) ; // deletes back 'pos' characters
void incr_yr(void)              ; // increments YEARS value (2 digits), used in the setup menu
void incr_mon(void)             ; // increments MONTHS value (2 digits), used in the setup menu
void incr_dat(void)             ; // increments DATE value (2 digits), used in the setup menu
void incr_hr(void)              ; // increments HOURS value (2 digits), used in the setup menu
void incr_min(void)             ; // increments MINUTES ('minutes' have no aux flags)
void incr_sec(void)             ; // increments SECONDS ; bit7 = ST = 1, used in the setup menu
```

RTCC Drivers (`spi_rtcc_drivers.h`)

Represent the medium-level communication between the MSSP1 module of the PIC18 and the SPI RTCC.

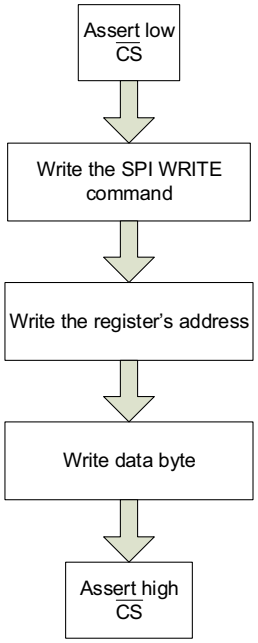
The related functions call the SPI drivers, as described below. Moreover, the library defines all necessary constants, as: registers, addresses and masks.

Writing a Byte to the SPI RTCC

```
void spi_rtcc_wr (unsigned char rtcc_reg, unsigned char time_var)   {
                                    // SPI write to the SPI RTCC
spi_rtcc_start()                  ;   // start SPI comm with the SPI RTCC, sets CS pin low
spi_wrbyte(SPI_RTCC_WRITE)        ;   // send the SPI WRITE command
spi_wrbyte(rtcc_reg)              ;   // send the register's address
spi_wrbyte(time_var)              ;   // send the SPI data byte
spi_rtcc_stop()                   ;}  // stop SPI comm, set CS pin high
```

The firmware for writes to the RTCC follows in the chart below.

Flowchart for Writes

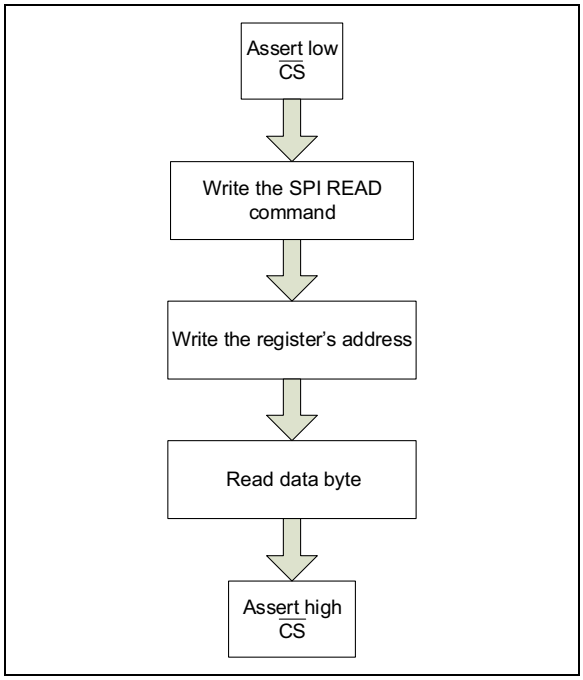


Reading a Byte from the SPI RTCC

```
unsigned char spi_rtcc_rd(unsigned char rtcc_reg){
                                        // SPI read from the SPI RTCC
spi_rtcc_start()                  ;   // start the SPI comm with the SPI RTCC, sets CS
                                        pin low
spi_wrbyte(SPI_RTCC_READ)         ;   // send the SPI READ command
spi_wrbyte(rtcc_reg)              ;   // send the register's address
rtcc_buf = spi_rdbyte()           ;   // read the result and store it
spi_rtcc_stop()                   ;   // stop the SPI comm with the SPI RTCC, sets CS
                                        pin high
return rtcc_buf                   ;}  // return the read result
```

The firmware for reads from the RTCC is shown in the flowchart below.

Flowchart for Reads



SPI Drivers (`spi_drivers.h`)

SPI drivers provide the low-level SPI communication with the RTCC. Its functions are called by the `spi_rtcc_drivers`, as depicted in the previous paragraph.

Keyboard Drivers (2 keys polling)

The set of keyboard drivers has only one function: `keyb_press()`. The `keyb_press()` function awaits the selection of one of the two on-board switches:
S1 (MENU key) or S2 (INCREMENT key). After the selection is made, the firmware updates the code of the pressed key. Upon exiting the function, a value is returned in either `KEYB_MENU` or `KEYB_INCR`. The function performs a key debounce of 2 x 100 msec. The function will exit only after the pressed key is released (deactivated). For more details about the operating system based on the two on-board switches, refer to **Section "Application Description"**.

The Interrupt Function

Interrupts are generated by the TMR0 overflow, which is initialized at 0xFFFF as a counter. TIMER0 is incremented once per second by the CLKOUT signal coming from the RTCC. The interrupt function calls the `display_time()` function, which reads the six related registers of the RTCC and puts them in the six global variables (year, month, date, hour, minute and seconds, found in "`lcd_drivers.h`"). The Random Byte Access mode is used, as some versions of the application can use only a subset of these six variables. In the end, the interrupt function (through the `display_time()` driver) displays these six variables on the on-board LCD, according to the format below:

ROW1: "date" string: year month date

ROW2: "time" string: hour minutes seconds

CONFIGURING THE SPI RTCC

The configuration of the RTCC includes two stages:

- initialization of the RTCC
- setup of timekeeping registers

Initialization of the SPI RTCC

The "spi_rtcc_drivers.h" header includes two initialization functions:

- "void ini_spi_rtcc(void)"
- "void ini_spi_time(void)"

The first function has the role to enable the battery through the VBATEN bit in the RTCWKDAY register and to set the CONTROL register, in order to disable the alarms and to configure the CLKOUT pin as square wave at 1 Hz frequency.

The corresponding code is indicated below:

```
void ini_spi_rtcc(void)                          {   // initialization of the SPI RTCC
day = spi_rtcc_rd(RTCWKDAY)                      ;   // read day register
spi_rtcc_wr(RTCWKDAY,day|VBATEN)                 ;   // enable the battery back-up
spi_rtcc_wr(CONTROL,ALM_NO+SQWEN+CLKO_01H)       ;   // write in the general control register:
                                                 }   // no alarms, CLKOUT = square wave = 1HZ
```

The second function tests the OSCRUN bit (RTCWKDAY). If the oscillator is already started, no action is taken. If the oscillator is not running, time/date are set arbitrary and the oscillator will be started. The code is presented below:

```
void ini_spi_time(void)                             // initialization of time/date vars on the SPI RTCC
                                                  {  // it initializes also START OSC.
    if((day&OSCRUN)==OSCRUN)                     {;}  // if oscillator = already running, do nothing.
    else                                          {  // if oscillator = not running, set time/date(arbitrary)
                                                     // and SART oscillator/ crystal
 spi_rtcc_wr(RTCYEAR,0x10)                        ;  // initialize YEAR register
 spi_rtcc_wr(RTCMTH,0x03)                         ;  // initialize MONTH register
 spi_rtcc_wr(RTCDATE,0x01)                        ;  // initialize DATE register
 spi_rtcc_wr(RTCHOUR,0x00)                        ;  // initialize HOUR register
 spi_rtcc_wr(RTCMIN,0x00)                         ;  // initialize MIN register
 spi_rtcc_wr(RTCSEC,ST)                          ;}  // initialize SEC register, start OSC
                                                  }
```

Setup of timekeeping registers

The sequence can be found in the end of the MAIN function and updates the six time/date registers, through the six time date variables: `yr, mon, dat, hr, min, sec`.

```
spi_rtcc_wr(RTCYEAR,yr)              ;   // update YEAR value in RTCC
spi_rtcc_wr(RTCMTH,mon)              ;   // update MONTH value in RTCC
                                         // LPYR bit is read only; you may clear it.
spi_rtcc_wr(RTCDATE,dat)             ;   // update DATE value in RTCC
spi_rtcc_wr(RTCHOUR,hr)              ;   // update HOUR value in RTCC
spi_rtcc_wr(RTCMIN,min)              ;   // update MINUTES value in RTCC
sec = sec | ST                       ;   // restore oscillator START bit
spi_rtcc_wr(RTCSEC,sec)              ;   // update SECONDS value in RTCC
```

USING THE SPI RTCC FOR BASIC TIMEKEEPING

The usage of timekeeping registers, consists of reading and displaying them on the LCD.

These two operations are performed by the `display_time()` function, once per second, in interrupts.

```
void display_time(void)                {  // displays all time/date variables:
                                          // YEAR, MONTH, DATE, HOUR, MINUTES, SECONDS
     yr = spi_rtcc_rd(RTCYEAR)         ;  // read YEAR
     mon = spi_rtcc_rd(RTCMTH)         ;  // read MONTH
     mon = mon & (~LPYR)               ;  // mask the leap year bit
     dat = spi_rtcc_rd(RTCDATE)        ;  // read DATE
     hr = spi_rtcc_rd(RTCHOUR)         ;  // read HOUR
     min = spi_rtcc_rd(RTCMIN)         ;  // read MIN
     sec = spi_rtcc_rd(RTCSEC)         ;  // read SEC; once finished the RTCC's READ

     wrcmnd_lcd(SET_DDRAM+N2_ROW1+07)  ;  // set 'YEAR' position on the first row
     yr_to_lcd(); wrdata_lcd(' ')      ;  // display YEAR + separator
     mon_to_lcd(); wrdata_lcd(' ')     ;  // display MONTH + separator
     dat_to_lcd()                      ;  // display DATE
     wrcmnd_lcd(SET_DDRAM+N2_ROW2+07)  ;  // set HOUR position on the second row
     hr_to_lcd(); wrdata_lcd(' ')      ;  // display HOUR + separator
     min_to_lcd()                      ;  // display MINUTES
     if((sec&0x7f)%2){ wrdata_lcd(':');} // display separator ':' for odd seconds
     else            { wrdata_lcd(' ');} // display separator ' ' for even seconds
     sec_to_lcd()                      ;} // display SECONDS, end of DISPLAY function
```

ACCESSING THE RTCC REGISTERS

There are two basic functions for accessing the RTCC registers: one for writes and one for reads. Each of them was fully described in the above paragraphs. Both use register addresses, inside the SRAM zone of the SPI RTCC.

As described in the MCP795XXX data sheet (DS20002280), the addresses of the RTCC register are shown in Table 1.

TABLE 1: RTCC REGISTER ADDRESSES

Address	Register Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
00h	RTCHSEC	HSECTEN3	HSECTEN2	HSECTEN1	HSECTEN0	HSECONE3	HSECONE2	HSECONE1	HSECONE0
01h	RTCSEC	ST	SECTEN2	SECTEN1	SECTEN0	SECONE3	SECONE2	SECONE1	SECONE0
02h	RTCMIN	—	MINTEN2	MINTEN1	MINTEN0	MINONE3	MINONE2	MINONE1	MINONE0
03h	RTCHOUR	TRIMSIGN	12/$\overline{24}$	$\overline{AM}$/PM HRTEN1	HRTEN0	HRONE3	HRONE2	HRONE1	HRONE0
04h	RTCWKDAY	—	—	OSCRUN	PWRFAIL	VBATEN	WKDAY2	WKDAY1	WKDAY0
05h	RTCDATE	—	—	DATETEN1	DATETEN0	DATEONE3	DATEONE2	DATEONE1	DATEONE0
06h	RTCMTH	—	—	LPYR	MTHTEN0	MTHONE3	MTHONE2	MTHONE1	MTHONE0
07h	RTCYEAR	YRTEN3	YRTEN2	YRTEN1	YRTEN0	YRONE3	YRONE2	YRONE1	YRONE0
08h	CONTROL	OUT	SQWEN	ALM1EN	ALM0EN	EXTOSC	CRSTRIM	SQWFS1	SQWFS0
09h	OSCTRIM	TRIMVAL7	TRIMVAL6	TRIMVAL5	TRIMVAL4	TRIMVAL3	TRIMVAL2	TRIMVAL1	TRIMVAL0

According to these addresses, in the basic read/write functions, only the register's address will differ. Reads are used in the interrupt function (once/second). Writes are used in the initialization function and in the setup sequence (the main function).

An SPI access to the RTCC needs two SPI commands as:

READ	`0001 0011`	Read RTCC/SRAM array beginning at selected address
WRITE	`0001 0010`	Write RTCC/SRAM data to memory array beginning at selected address

All register addresses and flag masks, are defined in "`spi_rtcc_drivers.h`".

CONCLUSION

This application note presents how to control (display and setup) an electronic watch, based on Microchip's SPI RTCC, MCP795WXX. The project is performed on a PIC18 Explorer demo board, using the on-board resources: LCD (accessed through the SPI bus) and push buttons. The code (drivers and main function) is written in C, using MPLAB® X v3.55 and XC8 compiler v1.34. The target microcontroller is the PIC18F87J11.

APPENDIX A: REVISION HISTORY

Revision A (June 2015)

- Initial release of this document.

Revision B (July 2017)

- Updated bit and register names to match new data sheet format.

NOTES:

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Trademarks

ISBN: 978-1-5224-1936-5

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-3326-8000
Fax: 86-21-3326-8021

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

France - Saint Cloud
Tel: 33-1-30-60-70-00

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7289-7561

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820