Section 29. Real-Time Clock and Calendar

HIGHLIGHTS

This section of the manual contains the following topics:

29.1 INTRODUCTION

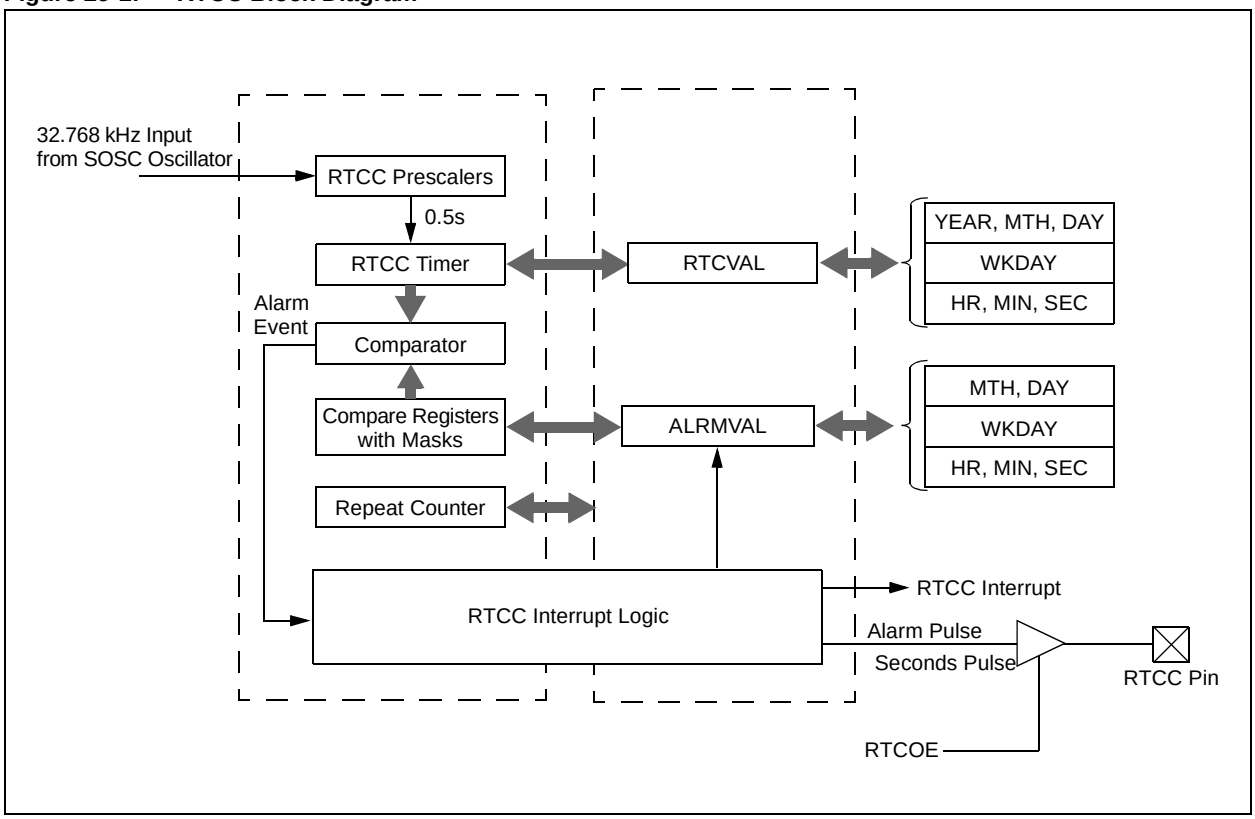
This section discusses the Real-Time Clock and Calendar (RTCC) hardware module, available on PIC32MX devices, and its operation. Listed below are some of the key features of this module:

- Time: Hours, Minutes and Seconds
- 24-hour format (military time)
- Visibility of one-half second period
- Provides calendar: Weekday, Date, Month and Year
- Alarm configurable for half a second, one second, 10 seconds, one minute, 10 minutes, one hour, one day, one week, one month, one year
- Alarm repeat with decrementing counter
- Alarm with indefinite repeat: chime
- Year Range: 2000 to 2099
- Leap Year Correction
- BCD format for smaller firmware overhead
- Optimized for long term battery operation
- Fractional second synchronization
- User calibration of the clock crystal frequency with auto-adjust
- Calibration range: ±0.66 seconds error per month
- Calibrates up to 260 ppm of crystal error
- Requirements: external 32.768 kHz Clock Crystal
- Alarm Pulse or Seconds Clock Output on the RTCC pin

This module provides real-time clock and calendar functions. RTCC is intended for applications where accurate time must be maintained for extended periods of time with minimum-to-no intervention from the CPU. The module is optimized for low-power usage in order to provide extended battery lifetime while keeping track of time.

The RTCC module is a 100-year clock and calendar with automatic leap year detection. The range of the clock is from 00:00:00 (midnight) on January 1, 2000 to 23:59:59 on December 31, 2099. The hours are available in 24-hour (military time) format. The clock provides a granularity of one second with half-second visibility to the user.

Figure 29-1: RTCC Block Diagram



29.2 STATUS AND CONTROL REGISTERS

The RTCC module registers includes the following Special Function Registers (SFRs):

The RTCCON and RTCALRM registers control the operation of the RTCC module.

- RTCCON: Control Register for the RTCC Module
  RTCCONCLR, RTCCONSET, RTCCONINV: Atomic Bit Manipulation, Write-only Registers for RTCCON
- RTCALRM: Control Register for the Alarm Functions of the RTCC Module
  RTCALRMCLR, RTCALRMSET, RTCALRMINV: Atomic Bit Manipulation, Write-only Registers for RTCALRM
- RTCTIME: RTCC Time Register, including Hour, Minutes and Seconds Fields.
  RTCTIMECLR, RTCTIMESET, RTCTIMEINV: Atomic Bit Manipulation, Write-only Registers for RTCTIME
- RTCDATE: RTCC Date Register, including Year, Month, Day and Weekday Fields.
  RTCDATECLR, RTCDATESET, RTCDATEINV: Atomic Bit Manipulation, Write-only Registers for RTCDATE
- ALRMTIME: RTCC Alarm Time Register, including Alarm Hour, Minutes and Seconds Fields
  ALRMTIMECLR, ALRMTIMESET, ALRMTIMEINV: Atomic Bit Manipulation, Write-only Registers for ALRMTIME
- ALRMDATE: RTCC Alarm Date Register, including Alarm Month, Day and Weekday Fields
  ALRMDATECLR, ALRMDATESET,ALRMDATEINV: Atomic Bit Manipulation, Write-only Registers for ALRMDATE
- IFS1: INT Controller Register signalling an Active RTCC Interrupt
  IFS1CLR, IFS1SET, IFS1INV: Atomic Bit Manipulation, Write-only Registers for IFS1
- IEC1: INT Controller Register enabling the RTCC Interrupt
  IEC1CLR, IEC1SET, IEC1INV: Atomic Bit Manipulation, Write-only Registers for IEC1
- IPC8: INT Controller Register for programming the RTCC Interrupt Priority and Subpriority
  IPC8CLR, IPC8SET, IPC8INV: Atomic Bit Manipulation, Write-only Registers for IPC8

The following table summarizes all RTCC-related registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

Table 29-1: RTCC SFR Summary

Name		Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
RTCCON	31:24	—	—	—	—	—	—	CAL9:8>	
	23:16	CAL<7:0>							
	15:8	ON	FRZ	SIDL	—	—	—	—	—
	7:0	RTSECSEL	RTCCLKON	—	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE
RTCCONCLR	31:0	Write clears selected bits in RTCCON, read yields undefined value							
RTCCONSET	31:0	Write sets selected bits in RTCCON, read yields undefined value							
RTCCONINV	31:0	Write inverts selected bits in RTCCON, read yields undefined value							
RTCALRM	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	ALRMEN	CHIME	PIV	ALRMSYNC	AMASK<3:0>			
	7:0	ARPT<7:0>							
RTCALRMCLR	31:0	Write clears selected bits in RTCALRM, read yields undefined value							
RTCALRMSET	31:0	Write sets selected bits in RTCALRM, read yields undefined value							
RTCALRMINV	31:0	Write inverts selected bits in RTCALRM, read yields undefined value							
RTCTIME	31:24	HR10<3:0>				HR01<3:0>			
	23:16	MIN10<3:0>				MIN01<3:0>			
	15:8	SEC10<3:0>				SEC01<3:0>			
	7:0	—	—	—	—	—	—	—	—
RTCTIMECLR	31:0	Write clears selected bits in RTCTIME, read yields undefined value							
RTCTIMESET	31:0	Write sets selected bits in RTCTIME, read yields undefined value							
RTCTIMEINV	31:0	Write inverts selected bits in RTCTIME, read yields undefined value							
RTCDATE	31:24	YEAR10<3:0>				YEAR01<3:0>			
	23:16	MONTH10<3:0>				MONTH01<3:0>			
	15:8	DAY10<3:0>				DAY01<3:0>			
	7:0	—	—	—	—	WDAY01<3:0>			
RTCDATECLR	31:0	Write clears selected bits in RTCDATE, read yields undefined value							
RTCDATESET	31:0	Write sets selected bits in RTCDATE, read yields undefined value							
RTCDATEINV	31:0	Write inverts selected bits in RTCDATE, read yields undefined value							
ALRMTIME	31:24	HR10<3:0>				HR01<3:0>			
	23:16	MIN10<3:0>				MIN01<3:0>			
	15:8	SEC10<3:0>				SEC01<3:0>			
	7:0	—	—	—	—	—	—	—	—
ALRMTIMECLR	31:0	Write clears selected bits in ALRMTIME, read yields undefined value							
ALRMTIMESET	31:0	Write sets selected bits in ALRMTIME, read yields undefined value							
ALRMTIMEINV	31:0	Write inverts selected bits in ALRMTIME, read yields undefined value							
ALRMDATE	31:24	—	—	—	—	—	—	—	—
	23:16	MONTH10<3:0>				MONTH01<3:0>			
	15:8	DAY10<3:0>				DAY01<3:0>			
	7:0	—	—	—	—	WDAY01<3:0>			
ALRMDATECLR	31:0	Write clears selected bits in ALRMDATE, read yields undefined value							
ALRMDATESET	31:0	Write sets selected bits in ALRMDATE, read yields undefined value							
ALRMDATEINV	31:0	Write inverts selected bits in ALRMDATE, read yields undefined value							

Table 29-1: RTCC SFR Summary (Continued)

Name		Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
IFS1	31:24	—	—	—	—	—	—	USBIF	FCEIF
	23:16	—	—	—	—	DMA3IF	DMA2IF	DMA1IF	DMA0IF
	15:8	RTCCIF	FSCMIF	I2C2MIF	I2C2SIF	I2C2BIF	U2TXIF	U2RXIF	U2EIF
	7:0	SPI2RXIF	SPI2TXIF	SPI2EIF	CMP2IF	CMP1IF	PMPIF	AD1IF	CNIF
IFS1CLR	31:0	Write clears selected bits in IFS1, read yields undefined value							
IFS1SET	31:0	Write sets selected bits in IFS1, read yields undefined value							
IFS1INV	31:0	Write inverts selected bits in IFS1, read yields undefined value							
IEC1	31:24	—	—	—	—	—	—	USBIE	FCEIE
	23:16	—	—	—	—	DMA3IE	DMA2IE	DMA1IE	DMA0IE
	15:8	RTCCIE	FSCMIE	I2C2MIE	I2C2SIE	I2C2BIE	U2TXIE	U2RXIE	U2EIE
	7:0	SPI2RXIE	SPI2TXIE	SPI2EIE	CMP2IE	CMP1IE	PMPIE	AD1IE	CNIE
IEC1CLR	31:0	Write clears selected bits in IEC1, read yields undefined value							
IEC1SET	31:0	Write sets selected bits in IEC1, read yields undefined value							
IEC1INV	31:0	Write inverts selected bits in IEC1, read yields undefined value							
IPC8	31:24	—	—	—	RTCCIP<2:0>			RTCCIS<1:0>	
	23:16	—	—	—	FSCMIP<2:0>			FSCMIS<1:0>	
	15:8	—	—	—	I2C2IP<2:0>			I2C2IS<1:0>	
	7:0	—	—	—	U2IP<2:0>			U2IS<1:0>	
IPC8CLR	31:0	Write clears the selected bits in IPC8, read yields undefined value							
IPC8SET	31:0	Write sets the selected bits in IPC8, read yields undefined value							
IPC8INV	31:0	Write inverts the selected bits in IPC8, read yields undefined value							

Register 29-1: RTCCON: RTC Control Register(1)

r-x	r-x	r-x	r-x	r-x	r-x	R/W-0	R/W-0
—	—	—	—	—	—	CAL<9:8>	
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CAL<7:0>							
bit 23							bit 16

R/W-0	R/W-0	R/W-0	r-x	r-x	r-x	r-x	r-x
ON	FRZ	SIDL	—	—	—	—	—
bit 15							bit 8

R/W-0	R-0	r-x	r-x	R/W-0	R-0	R-0	R/W-0
RTSECSEL	RTCCLKON	—	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-26 **Reserved:** Write ‘0’; ignore read

bit 25-16 **CAL<9:0>:** RTC Drift Calibration bits, contains a signed 10-bit integer value
0111111111= Maximum positive adjustment, adds 511 RTC clock pulses every one minute
...
0000000001= Minimum positive adjustment, adds 1 RTC clock pulse every one minute
0000000000= No adjustment
1111111111= Minimum negative adjustment, subtracts 1 RTC clock pulse every one minute
...
1000000000= Minimum negative adjustment, subtracts 512 clock pulses every one minute

bit 15 **ON:** RTCC On bit
1 = RTCC module is enabled
0 = RTCC module is disabled
Note 1: The ON bit is only writable when RTCWREN = 1.
2: When using 1:1 PBCLK divisor, the user’s software should not read/write the peripheral’s SFRs in the SYSCLK cycle immediately following the instruction that clears the module’s ON bit.

bit 14 **FRZ:** Freeze in DEBUG Mode bit
1 = When emulator is in DEBUG mode, module freezes operation
0 = When emulator is in DEBUG mode, module continues operation
Note: FRZ is writable in Debug Exception mode only, it is forced to ‘0’ in normal mode.

bit 13 **SIDL:** Stop in IDLE Mode bit
1 = Disables the PBCLK to the RTCC when CPU enters in IDLE mode
0 = Continue normal operation in IDLE mode

bit 12-8 **Reserved:** Write ‘0’; ignore read

bit 7 **RTSECSEL:** RTCC Seconds Clock Output Select
1 = RTCC Seconds Clock is selected for the RTCC pin
0 = RTCC Alarm Pulse is selected for the RTCC pin
Note: Requires RTCOE == 1 (RTCCON<0>) for the output to be active.

Register 29-1: RTCCON: RTC Control Register(1) (Continued)

bit 6 **RTCCLKON:** Status of the RTCC clock enable

1 = RTCC Clock is actively running
0 = RTCC Clock is not running

bit 5-4 **Reserved:** Write '0'; ignore read

bit 3 **RTCWREN:** RTC Value Registers Write Enable bit

1 = RTC Value registers can be written to by the user
0 = RTC Value registers are locked out from being written to by the user

Note: The RTCWREN bit can be set only when write sequence enabled. The register can be written to a '0' at any time.

bit 2 **RTCSYNC:** RTCC Value Registers Read Synchronization bit

1 = RTC Value registers can change while reading, due to a roll-over ripple that results in an invalid data read
If the register is read twice and results in the same data, the data can be assumed to be valid
0 = RTC Value registers can be read without concern about a roll-over ripple

bit 1 **HALFSEC:** Half-Second Status bit

1 = Second half period of a second
0 = First half period of a second

Note: This bit is read-only. It is cleared to '0' on a write to the SECONDS register.

bit 0 **RTCOE:** RTCC Output Enable bit

1 = RTCC clock output enabled – clock presented onto an I/O
0 = RTCC clock output disabled

Note: This bit is ANDed with ON (RTCCON<15>) to produce the effective RTCC output enable.

Note 1: This register is only reset by Power-on Reset (POR).

Register 29-2: RTCCONCLR: RTCCON Clear Register

Write clears selected bits in RTCCON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in RTCCON**

A write of '1' in one or more bit positions clears the corresponding bit(s) in RTCCON register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `RTCCONCLR = 0x00008001` clears bits 15 and 0 in RTCCON register.

Register 29-3: RTCCONSET: RTCCON Set Register

Write sets selected bits in RTCCON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in RTCCON**

A write of '1' in one or more bit positions sets the corresponding bit(s) in RTCCON register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `RTCCONSET = 0x00008001` sets bits 15 and 0 in RTCCON register.

Register 29-4: RTCCONINV: RTCCON Invert Register

Write inverts selected bits in RTCCON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in RTCCON**

A write of '1' in one or more bit positions inverts the corresponding bit(s) in RTCCON register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `RTCCONINV = 0x00008001` inverts bits 15 and 0 in RTCCON register.

Register 29-5: RTCALRM: RTC ALARM Control Register(1)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
ALRMEN	CHIME	PIV	ALRMSYNC	AMASK<3:0>			
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ARPT<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Write ‘0’; ignore read

bit 15 **ALRMEN:** Alarm Enable bit

1 = Alarm is enabled
0 = Alarm is disabled

Note: Hardware clears ALRMEN anytime the alarm event occurs, when ARPT<7:0> = 00 and CHIME = 0.
This field should not be written when RTCC ON = 1 (RTCCON<15>) and ALRMSYNC = 1.

bit 14 **CHIME:** Chime Enable bit

1 = Chime is enabled – ARPT<7:0> is allowed to roll over from 00 to FF
0 = Chime is disabled – ARPT<7:0> stops once it reaches 00

Note: This field should not be written when RTCC ON = 1 (RTCCON<15>) and ALRMSYNC = 1.

bit 13 **PIV:** Alarm Pulse Initial Value bit

When ALRMEN = 0, PIV is writable and determines the initial value of the Alarm Pulse.
When ALRMEN = 1, PIV is read-only and returns the state of the Alarm Pulse.

Note: This field should not be written when RTCC ON = 1 (RTCCON<15>) and ALRMSYNC = 1.

bit 12 **ALRMSYNC:** Alarm Sync bit

1 = ARPT<7:0> and ALRMEN may change as a result of a half second rollover during a read. The ARPT must be read repeatedly until the same value is read twice. This must be done since multiple bits may be changing, which are then synchronized to the PB clock domain
0 = ARPT<7:0> and ALRMEN can be read without concerns of rollover because prescaler is > 32 RTC clock away from a half second rollover.

Note: This assumes a CPU read will execute in less than 32 PBCLKs.

Register 29-5: RTCALRM: RTC ALARM Control Register(1) (Continued)

bit 11-8 **AMASK<3:0>:** Alarm Mask Configuration bits

0000 = Every half second
0001 = Every second
0010 = Every 10 seconds
0011 = Every minute
0100 = Every 10 minutes
0101 = Every hour
0110 = Once a day
0111 = Once a week
1000 = Once a month
1001 = Once a year (except when configured for February 29th, once every 4 years)
1010 = Reserved – do not use
1011 = Reserved – do not use
11XX = Reserved – do not use

Note: This field should not be written when RTCC ON = 1 (RTCCON<15>) and ALRMSYNC = 1.

bit 7-0 **ARPT<7:0>:** Alarm Repeat Counter Value bits

11111111 = Alarm will trigger 256 times
...
00000000 = Alarm will trigger 1 time

The counter decrements on any alarm event. The counter only rolls over from 00 to FF if CHIME = 1.

Note: This field should not be written when RTCC ON = 1 (RTCCON<15>) and ALRMSYNC = 1.

Note 1: This register is only reset by POR.

Register 29-6: RTCALRMCLR: RTCALRM Clear Register

Write clears selected bits in RTCALRM, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in RTCALRM**

A write of '1' in one or more bit positions clears the corresponding bit(s) in RTCALRM register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `RTCALRMCLR = 0x0000c000` clears bits 15 and 14 in RTCALRM register.

Register 29-7: RTCALRMSET: RTCALRM Set Register

Write sets selected bits in RTCALRM, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in RTCALRM**

A write of '1' in one or more bit positions sets the corresponding bit(s) in RTCALRM register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `RTCALRMSET = 0x0000c000` sets bits 15 and 14 in RTCALRM register.

Register 29-8: RTCALRMINV: RTCALRM Invert Register

Write inverts selected bits in RTCALRM, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in RTCALRM**

A write of '1' in one or more bit positions inverts the corresponding bit(s) in RTCALRM register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `RTCALRMINV = 0x0000c000` inverts bits 15 and 14 in RTCALRM register.

Register 29-9: RTCTIME: RTC Time Value Register(1)

R-0	R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
HR10<3:0>				HR01<3:0>			
bit 31							bit 24

R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
MIN10<3:0>				MIN01<3:0>			
bit 23							bit 16

R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SEC10<3:0>				SEC01<3:0>			
bit 15							bit 8

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-28 **HR10<3:0>:** Binary-Coded Decimal Value of Hours bits, 10 digits; contains a value from 0 to 2
Note: HR10<3:2> bits are always read ‘0’.

bit 27-24 **HR01<3:0>:** Binary-Coded Decimal Value of Hours bits, 1 digit; contains a value from 0 to 9

bit 23-20 **MIN10<3:0>:** Binary-Coded Decimal Value of Minutes bits, 10 digits; contains a value from 0 to 5
Note: MIN10<3> bit is always read ‘0’.

bit 19-16 **MIN01<3:0>:** Binary-Coded Decimal Value of Minutes bits, 1 digit; contains a value from 0 to 9

bit 15-12 **SEC10<3:0>:** Binary-Coded Decimal Value of Seconds bits, 10 digits; contains a value from 0 to 5
Note: SEC10<3> bit is always read ‘0’.

bit 11-8 **SEC01<3:0>:** Binary-Coded Decimal Value of Seconds bits, 1 digit; contains a value from 0 to 9

bit 7-0 **Reserved:** Write ‘0’; ignore read

Note 1: This register is only writable when RTCWREN = 1 (RTCCON<3>).

Register 29-10: RTCTIMECLR: RTCTIME Clear Register

Write clears selected bits in RTCTIME, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in RTCTIME**

A write of '1' in one or more bit positions clears the corresponding bit(s) in RTCTIME register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `RTCTIMECLR = 0x0000ff00` clears bits 15:8 in RTCTIME register.

Register 29-11: RTCTIMESET: RTCTIME Set Register

Write sets selected bits in RTCTIME, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in RTCTIME**

A write of '1' in one or more bit positions sets the corresponding bit(s) in RTCTIME register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `RTCTIMESET = 0x00005900` sets seconds to 59 in RTCTIME register.

Register 29-12: RTCTIMEINV: RTCTIME Invert Register

Write inverts selected bits in RTCTIME, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in RTCTIME**

A write of '1' in one or more bit positions inverts the corresponding bit(s) in RTCTIME register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `RTCTIMEINV = 0x00000300` inverts bits 9 and 8 in RTCTIME register.

Register 29-13: RTCDATE: RTC Date Value Register(1)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
YEAR10<3:0>				YEAR01<3:0>			
bit 31							bit 24

R-0	R-0	R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
MONTH10<3:0>				MONTH01<3:0>			
bit 23							bit 16

R-0	R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
DAY10<3:0>				DAY01<3:0>			
bit 15							bit 8

r-x	r-x	r-x	r-x	R-0	R/W-x	R/W-x	R/W-x
—	—	—	—	WDAY01<3:0>			
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-28 **YEAR10<3:0>:** Binary-Coded Decimal Value of Years bits, 10 digits

bit 27-24 **YEAR01<3:0>:** Binary-Coded Decimal Value of Years bits, 1 digit

bit 23-20 **MONTH10<3:0>:** Binary-Coded Decimal Value of Months bits, 10 digits; contains a value from 0 to 1
Note: MONTH10<3:1> bits are always read ‘0’.

bit 19-16 **MONTH01<3:0>:** Binary-Coded Decimal Value of Months bits, 1 digit; contains a value from 0 to 9

bit 15-12 **DAY10<3:0>:** Binary-Coded Decimal Value of Days bits, 10 digits; contains a value from 0 to 3
Note: DAY10<3:2> bits are always read ‘0’.

bit 11-8 **DAY01<3:0>:** Binary-Coded Decimal Value of Days bits, 1 digit; contains a value from 0 to 9

bit 7-4 **Reserved:** Write ‘0’; ignore read

bit 3-0 **WDAY01<3:0>:** Binary-Coded Decimal Value of Weekdays bits,1 digit; contains a value from 0 to 6
Note: WDAY01<3> bit is always read ‘0’.

Note 1: This register is only writable when RTCWREN = 1 (RTCCON<3>).

Register 29-14: RTCDATECLR: RTCDATE Clear Register

Write clears selected bits in RTCDATE, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in RTCDATE**

A write of '1' in one or more bit positions clears the corresponding bit(s) in RTCDATE register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `RTCDATECLR = 0x00000007` will clear bits 2:0 in RTCDATE register.

Register 29-15: RTCDATESET: RTCDATE Set Register

Write sets selected bits in RTCDATE, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in RTCDATE**

A write of '1' in one or more bit positions sets the corresponding bit(s) in RTCDATE register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `RTCDATESET = 0x00000003` sets weekday to 3 in RTCDATE register.

Register 29-16: RTCDATEINV: RTCDATE Invert Register

Write inverts selected bits in RTCDATE, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in RTCDATE**

A write of '1' in one or more bit positions inverts the corresponding bit(s) in RTCDATE register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `RTCDATEINV = 0x00000003` inverts bits 1 and 0 in RTCDATE register.

Register 29-17: ALRMTIME: Alarm Time Value Register

R-0	R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
HR10<3:0>				HR01<3:0>			
bit 31							bit 24

R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
MIN10<3:0>				MIN01<3:0>			
bit 23							bit 16

R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SEC10<3:0>				SEC01<3:0>			
bit 15							bit 8

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-28 **HR10<3:0>:** Binary Coded Decimal value of hours bits, ‘10’ digit; contains a value from 0 to 2
Note: HR10<3:2> bits are always read ‘0’.

bit 27-24 **HR01<3:0>:** Binary Coded Decimal value of hours bits, ‘1’ digit; contains a value from 0 to 9

bit 23-20 **MIN10<3:0>:** Binary Coded Decimal value of minutes bits, ‘10’ digit; contains a value from 0 to 5
Note: MIN10<3> bit is always read ‘0’.

bit 19-16 **MIN01<3:0>:** Binary Coded Decimal value of minutes bits, ‘1’ digit; contains a value from 0 to 9

bit 15-12 **SEC10<3:0>:** Binary Coded Decimal value of seconds bits, ‘10’ digit; contains a value from 0 to 5
Note: SEC10<3> bit is always read ‘0’.

bit 11-8 **SEC01<3:0>:** Binary Coded Decimal value of seconds bits, ‘1’ digit; contains a value from 0 to 9

bit 7-0 **Reserved:** Write ‘0’; ignore read

Register 29-18: ALRMTIMECLR: ALRMTIME Clear Register

Write clears selected bits in ALRMTIME, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in ALRMTIME**

A write of '1' in one or more bit positions clears the corresponding bit(s) in ALRMTIME register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `ALRMTIMECLR = 0x0000ff00` clears bits 15:8 in ALRMTIME register.

Register 29-19: ALRMTIMESET: ALRMTIME Set Register

Write sets selected bits in ALRMTIME, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in ALRMTIME**

A write of '1' in one or more bit positions sets the corresponding bit(s) in ALRMTIME register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `ALRMTIMESET = 0x00330000` sets alarm minutes to 33 in ALRMTIME register.

Register 29-20: ALRMTIMEINV: ALRMTIME Invert Register

Write inverts selected bits in ALRMTIME, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in ALRMTIME**

A write of '1' in one or more bit positions inverts the corresponding bit(s) in ALRMTIME register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `ALRMTIMEINV = 0x00000300` inverts bits 9 and 8 in ALRMTIME register.

Register 29-21: ALRMDATE: Alarm Date Value Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

R-0	R-0	R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
MONTH10<3:0>				MONTH01<3:0>			
bit 23							bit 16

R-0	R-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
DAY10<3:0>				DAY01<3:0>			
bit 15							bit 8

r-x	r-x	r-x	r-x	R-0	R/W-x	R/W-x	R/W-x
—	—	—	—	WDAY01<3:0>			
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-24 **Reserved:** Write ‘0’; ignore read

bit 23-20 **MONTH10<3:0>:** Binary Coded Decimal value of months bits, ‘10’ digit; contains a value from 0 to 1
Note: MONTH10<3:1> bits are always read ‘0’.

bit 19-16 **MONTH01<3:0>:** Binary Coded Decimal value of months bits, ‘1’ digit; contains a value from 0 to 9

bit 15-12 **DAY10<3:0>:** Binary Coded Decimal value of days bits, ‘10’ digit; contains a value from 0 to 3
Note: DAY10<3:2> bits are always read ‘0’.

bit 11-8 **DAY01<3:0>:** Binary Coded Decimal value of days bits, ‘1’ digit; contains a value from 0 to 9

bit 7-4 **Reserved:** Write ‘0’; ignore read

bit 3-0 **WDAY01<3:0>:** Binary Coded Decimal value of weekdays bits, ‘1’ digit; contains a value from 0 to 6
Note: WDAY01<3> bit is always read ‘0’.

Register 29-22: ALRMDATECLR: ALRMDATE Clear Register

Write clears selected bits in ALRMDATE, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in ALRMDATE**

A write of '1' in one or more bit positions clears the corresponding bit(s) in ALRMDATE register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `ALRMDATECLR = 0x00000007` clears bits 2:0 in ALRMDATE register.

Register 29-23: ALRMDATESET: ALRMDATE Set Register

Write sets selected bits in ALRMDATE, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in ALRMDATE**

A write of '1' in one or more bit positions sets the corresponding bit(s) in ALRMDATE register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `ALRMDATESET = 0x00003100` sets alarm day to 31 in ALRMDATE register.

Register 29-24: ALRMDATEINV: ALRMDATE Invert Register

Write inverts selected bits in ALRMDATE, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in ALRMDATE**

A write of '1' in one or more bit positions inverts the corresponding bit(s) in ALRMDATE register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `ALRMDATEINV = 0x00000003` inverts bits 1 and 0 in ALRMDATE register.

Register 29-25: IFS1: Interrupt Flag Status Register 1(1)

r-x	r-x	r-x	r-x	r-x	r-x	R/W-0	R/W-0
—	—	—	—	—	—	USBIF	FCEIF
bit 31							bit 24

r-0	r-0	r-0	r-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	DMA3IF	DMA2IF	DMA1IF	DMA0IF
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RTCCIF	FSCMIF	I2C2MIF	I2C2SIF	I2C2BIF	U2TXIF	U2RXIF	U2EIF
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-0
SPI2RXIF	SPI2TXIF	SPI2EIF	CMP2IF	CMP1IF	PMPIF	AD1IF	CNIF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-26 **Reserved:** Write ‘0’; ignore read

bit 25-24 Interrupt flags for other peripheral devices

bit 23-20 **Reserved:** Write ‘0’; ignore read

bit 19-16 Interrupt flags for other peripheral devices

bit 15 **RTCCIF**: RTCC Interrupt Flag bit
1 = RTCC interrupt pending
0 = No RTCC interrupt pending
Set by the hardware when an event is generated by the RTCC.
Cleared by the software, usually in the ISR.

bit 14-0 Interrupt flags for other peripheral devices

Note 1: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the RTCC.

Register 29-26: IEC1: Interrupt Enable Control Register 1(1)

r-x	r-x	r-x	r-x	r-x	r-x	R/W-0	R/W-0
—	—	—	—	—	—	USBIE	FCEIE
bit 31							bit 24

r-0	r-0	r-0	r-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	DMA3IE	DMA2IE	DMA1IE	DMA0IE
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RTCCIE	FSCMIE	I2C2MIE	I2C2SIE	I2C2BIE	U2TXIE	U2RXIE	U2EIE
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-0
SPI2RXIE	SPI2TXIE	SPI2EIE	CMP2IE	CMP1IE	PMPIE	AD1IE	CNIE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-26 Unimplemented: Read as ‘0’

bit 25-24 Interrupt flags for other peripheral devices

bit 23-20 **Reserved:** Write ‘0’; ignore read

bit 19-16 Interrupt flags for other peripheral devices

bit 15 **RTCCIE:** RTCC interrupt enable.
1 = RTCC interrupt enabled
0 = RTCC interrupt disabled
Set/cleared by the software to enable/disable the interrupt when an event is generated by the RTCC.

bit 14-0 Interrupt enable bits for other peripheral devices

Note 1: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the RTCC.

Register 29-27: IPC8: Interrupt Priority Control Register 8(1)

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RTCCIP<2:0>			RTCCIS<1:0>	
bit 31							bit 24

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FCSMIP<2:0>			FCSMIS<1:0>	
bit 23							bit 16

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	I2C2IP<2:0>			I2C2IS<1:0>	
bit 15							bit 8

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	U2IP<2:0>			U2IS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-29 **Reserved:** Write ‘0’; ignore read

bit 28-26 **RTCCIP<2:0>:** RTCC Interrupt Vector Priority bits
111 = RTCC interrupts have priority 7 (highest priority)
•
•
•
001 = RTCC interrupts have priority 1
000 = RTCC interrupts are disabled

bit 25-24 **RTCCIS<1:0>:** RTCC Interrupt Vector Subpriority bits
11 = RTCC interrupts have subpriority 3 (highest subpriority)
•
•
00 = RTCC interrupts have subpriority 0 (lowest subpriority)

bit 23-21 **Reserved:** Write ‘0’; ignore read

bit 20-16 **Interrupt priority control for other peripheral devices**

bit 15-13 **Reserved:** Write ‘0’; ignore read

bit 12-8 **Interrupt priority control for other peripheral devices**

bit 7-5 **Reserved:** Write ‘0’; ignore read

bit 4-0 **Interrupt priority control for other peripheral devices**

Note 1: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the RTCC.

29.3 MODES OF OPERATION

The RTCC module offers the following operating modes:

- Real-Time Clock and Calendar (RTCC) function
- Alarm function

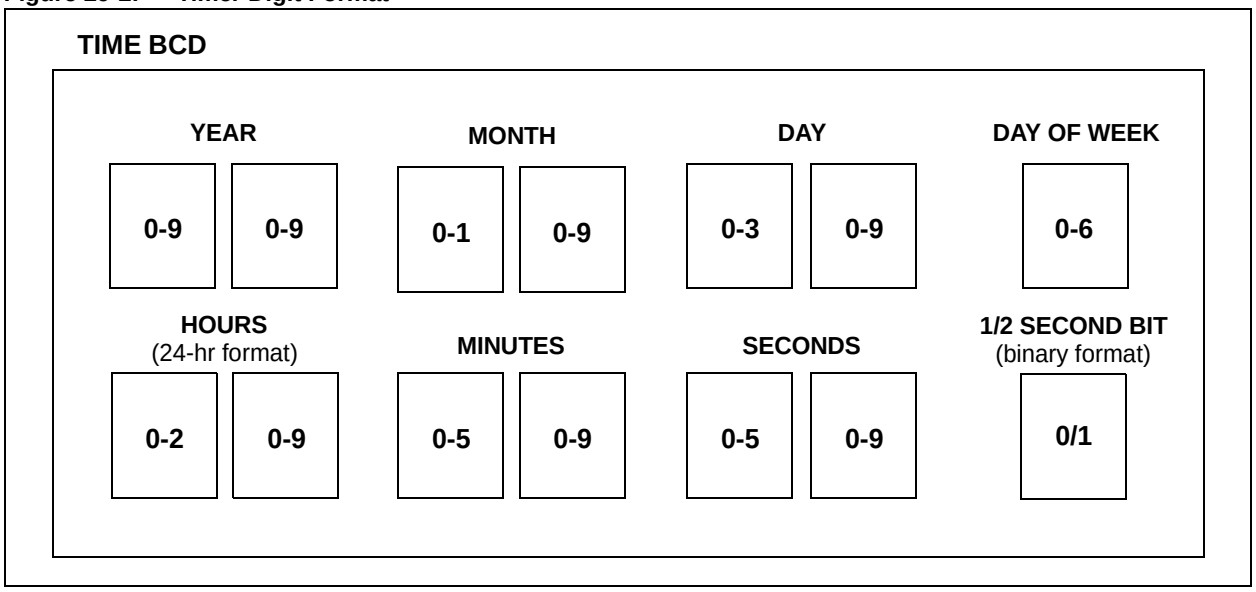
29.3.1 RTCC Operation

The RTCC is a 100-year clock and calendar with automatic leap year detection. The range of the clock is from 00:00:00 (midnight) on January 1, 2000, to 23:59:59 on December 31, 2099. The hours use the 24-hour time format (military time) with no hardware provisions for regular time format (AM/PM).

The RTCC provides a programming granularity of 1 second but has visibility of the half-second field.

The register interface for the RTCC values (RTCTIME and RTCDATE) is implemented using the Binary Coded Decimal (BCD) format. This simplifies the firmware, when using the module, as each of the digit values is contained within its own 4-bit value (see Figure 29-2).

Figure 29-2: Timer Digit Format



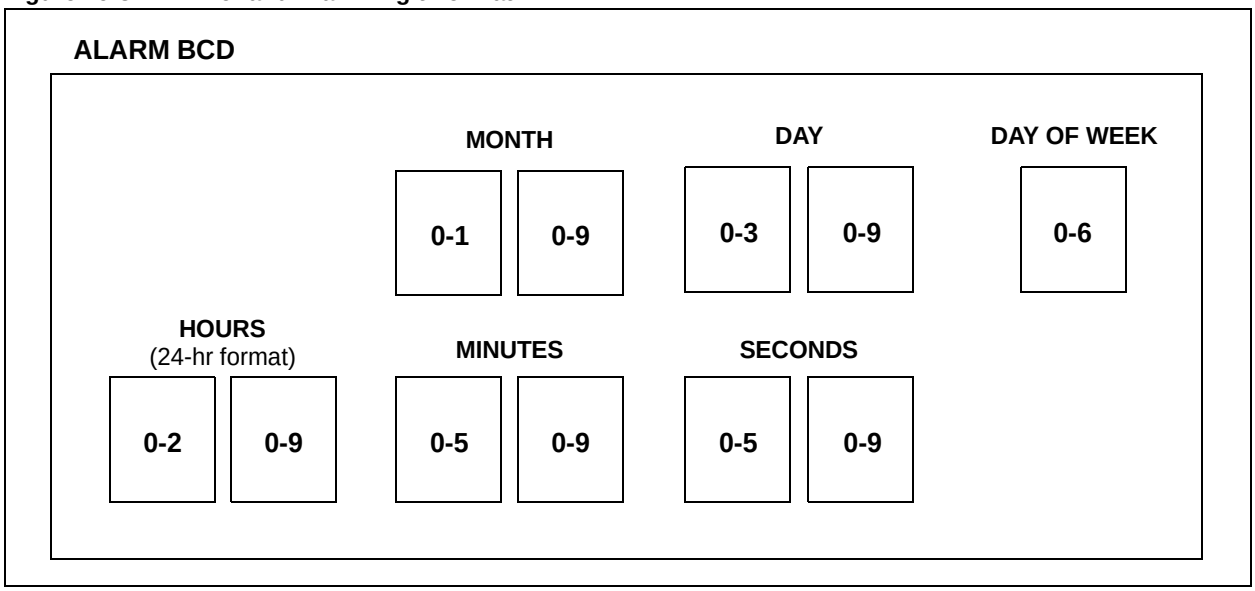
29.3.2 Alarm Operation

The module provides an alarm function configurable anywhere from a half-second to one year. However, only the half-second alarm has the half-second resolution. The module can be configured to repeat the alarm at pre-configured intervals, after the alarm is enabled. The indefinite repetition of the alarm is provided through the Chime feature.

The module provides an interrupt at every alarm pulse event. In addition to the alarm interrupt, an alarm pulse output is provided that operates at half the frequency of the alarm (the alarm pulse toggles at every alarm match). This output is completely synchronous with the RTCC clock and can be used to provide a trigger clock to other devices. The initial value of this output pin is controlled by PIV (RTCALRM<13>). See Register 29-5 for more information.

The register interface for the Alarm values (ALRMTIME and ALRMDATE) is implemented using the BCD format. This simplifies the firmware, when using the module, as each of the digit values is contained within its own 4-bit value (see Figure 29-3).

Figure 29-3: Timer and Alarm Digit Format

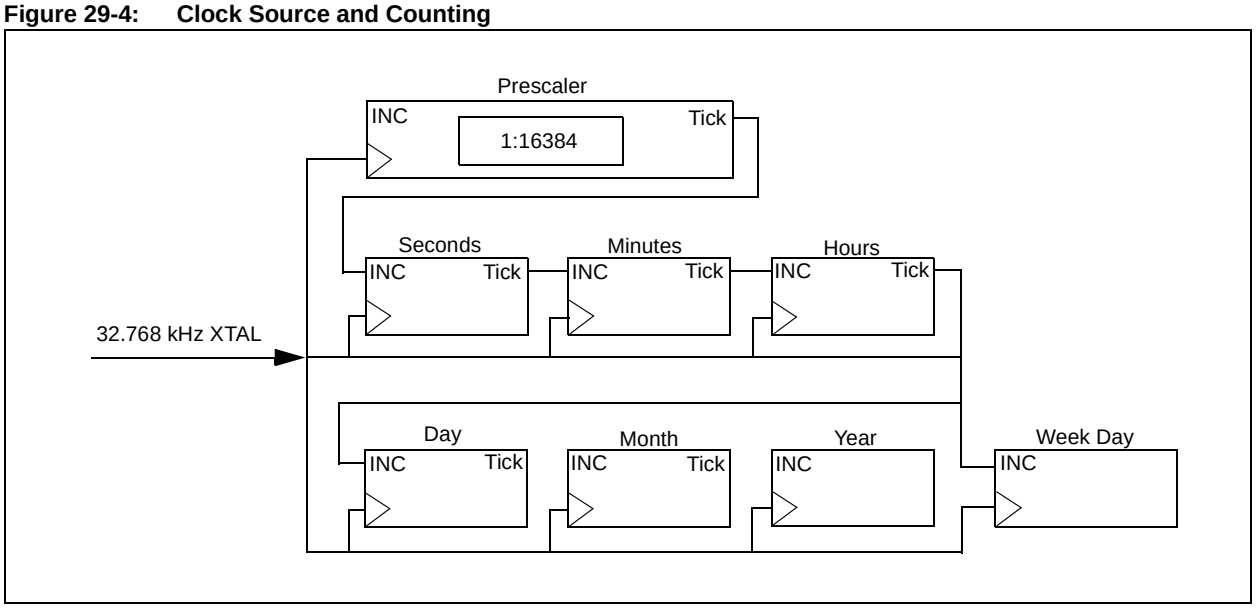


29.3.3 Clock Source

The RTCC module is intended to be clocked by an external real-time clock crystal that is oscillating at 32.768 kHz. Calibration of the crystal can be accomplished through this module, yielding an accuracy of +/-0.66 seconds per month (see **Section 29.3.10 "Calibration"** for further details).

To allow the RTCC to be clocked by an external 32.768 kHz crystal, the SOSCEN bit (OSCCON<1>) must be set (see **Section 6. "Oscillators"**, Register 6-1). This is the only bit outside of the RTCC module with which the user must be concerned for enabling the RTCC. The Status bit SOSCRDY (OSCCON<22>) can be used to check that the secondary oscillator is running.

Figure 29-4: Clock Source and Counting



29.3.4 Digit Carry Rules

This section explains which timer values are affected when there is a rollover.

- Time of Day – from 23:59:59 to 00:00:00, with a carry to the Day field
- Day – the carry from the day field to the month field is dependent on the current month (Refer to Table 29-3 for the day to month rollover schedule.)
- Month – from 12/31 to 01/01, with a carry to the Year field
- Day of Week – from 6 to 0, without a carry (refer to Table 29-2)
- Year – from 99 to 00, without a carry (this surpasses the intended use of the RTCC)

Considering that the following values are in BCD format, the carry to the upper BCD digit will occur at a count of 10, and not a count of 16 (SECONDS, MINUTES, HOURS, WEEKDAY, DAYS, MONTHS).

Table 29-2: Day of Week Schedule

Day of Week	
Sunday	0
Monday	1
Tuesday	2
Wednesday	3
Thursday	4
Friday	5
Saturday	6

Table 29-3: Day to Month Rollover Schedule

Month	Maximum Day Field
01 (January)	31
02 (February)	28 or 29(1)
03 (March)	31
04 (April)	30
05 (May)	31
06 (June)	30
07 (July)	31
08 (August)	31
09 (September)	30
10 (October)	31
11 (November)	30
12 (December)	31

Note 1: See **Section 29.3.5 “Leap Year”**.

29.3.5 Leap Year

Since the year range on the RTCC module is 2000 to 2099, the leap calculation is determined by any year divisible by 4 in the above range. The only month to be affected in a leap year is February. (February has 29 days in a leap year, but only 28 days in all other years.)

29.3.6 RTCC General Functionality

All timer registers containing a time value of seconds, or greater, are writable. The user can configure the current time by simply writing to these registers the desired year, month, day, hour, minutes and seconds. The timer will then use the newly written values to proceed with the count from the desired starting point.

Note that if the RTCC is enabled having ON = 1 (RTCCON<15>), the timer will continue incrementing even while the registers are being adjusted. However, any time the SECONDS register (RTCTIME<15:8>) is written, the Prescaler is reset to '0'. This provides a known Prescaler value after timer adjustments.

If an update (CPU write) of the timer register occurs, it is the user's responsibility to ensure that when ON = 1 (RTCCON<15>), a timer increment will not occur to the registers that are being updated. This can be done by observing the value of the RTCSYNC bit (RTCCON<2>), or the preceding digits from which a carry can occur, or by only updating the registers immediately following the seconds pulse (or alarm interrupt). Note that the corresponding counters are clocked based on their defined intervals, i.e., the DAYS register is clocked once a day, the MONTHS register is only clocked once a month, etc. This leaves large windows of time in which registers can be safely updated.

The timer also provides visibility into the half-second field of the counter. However, this value is read-only and can only be reset by writing to the SECONDS register (RTCTIME<15:8>).

29.3.7 Safety Window for Register Reads and Writes

The RTCSYNC bit (RTCCON<2>) indicates a time window during which: an update to the RTCC time registers (RTCTIME, RTCDATE) is not imminent, and the registers can be safely read and written. When RTCSYNC = 0, the registers can be safely accessed by the CPU. When RTCSYNC = 1, the user must employ a firmware solution to assure that the data read did not fall on an update boundary, resulting in an invalid or partial read.

The RTCSYNC bit is set 32 RTCC clock edges before an update is about to occur. It is cleared one clock later, after the update occurs (thus RTCSYNC is asserted for a total of 33 clocks). At worst case, when the CPU core uses the 32.768 kHz oscillator as its clock and the PBCLK is SYSCLK/8, there are at least 22 CPU clocks in which to execute instructions after a read of the RTCSYNC = 0 (some clock cycles might be lost due to synchronization and bus delays).

Note that, independent of the RTCSYNC value – the user can, by reading and comparing a timer register value twice, ensure in code that the register read did not span an RTCC clock update.

Writes to the Time and Date registers should not be performed when RTCSYNC = 1. This restriction exists for two reasons:

1. A write could cause a timing violation in the Alarm Match logic, leading to an invalid alarm event and a corruption of the ARPT register. This event can occur during the low time of an RTCC clock, following a rollover event.
2. A write during a rollover event, when the RTCC clock is high, will be ignored by H/W.

Example 29-1: Updating the RTCC Time and Date

```
/*
    The following code example will update the RTCC time and date.
*/

    assume the secondary oscillator is enabled and ready, i.e. OSCCON<1>=1, OSCCON<22>=1, and
    RTCC write is enabled i.e. RTCWREN (RTCCON<3>) =1;

    unsigned long time=0x04153300;// set time to 04 hr, 15 min, 33 sec
    unsigned long date=0x06102705;// set date to Friday 27 Oct 2006

    RTCCONCLR=0x8000;             // turn off the RTCC
    while(RTCCON&0x40);           // wait for clock to be turned off
    RTCTIME=time;                 // safe to update the time
    RTCDATE=date;                 // update the date
    RTCCONSET=0x8000;             // turn on the RTCC
    while(!(RTCCON&0x40));        // wait for clock to be turned on

                                  // can disable the RTCC write
```

Example 29-2: Updating the RTCC Time Using the RTCSYNC Window

```
/*
    The following code example will update the RTCC time and date.
*/

    assume RTCC write is enabled i.e. RTCWREN (RTCCON<3>) =1;

    unsigned long time=0x04153300;// set time to 04 hr, 15 min, 33 sec
    unsigned long date=0x06102705;// set date to Friday 27 Oct 2006

    asm volatile (“di”);          // disable interrupts, critical section follows
    while((RTCCON&0x4)!=0);       // wait for not RTCSYNC
    RTCTIME=time;                 // safe to update the time
    RTCDATE=date;                 // update the date
    asm volatile (“ei”);          // restore interrupts, critical section ended

                                  // can disable the RTCC write
```

29.3.8 Synchronization

The module provides a single RTCSYNC bit (RTCCON<2>) that the user must use to determine when it is safe to read and update the time and date registers. In addition, the module provides synchronization for Reset conditions (i.e., a write to the seconds register), and for ON (RTCCON<15>).

29.3.8.1 RTCSYNC Bit Generation

The RTCSYNC bit is a read-only bit that is set when ON = 1 and the RTCC Prescaler counter equals 0x7FE0 (32 clocks away from a one-second roll-over). Logic clears the RTCSYNC bit for any of the following conditions:

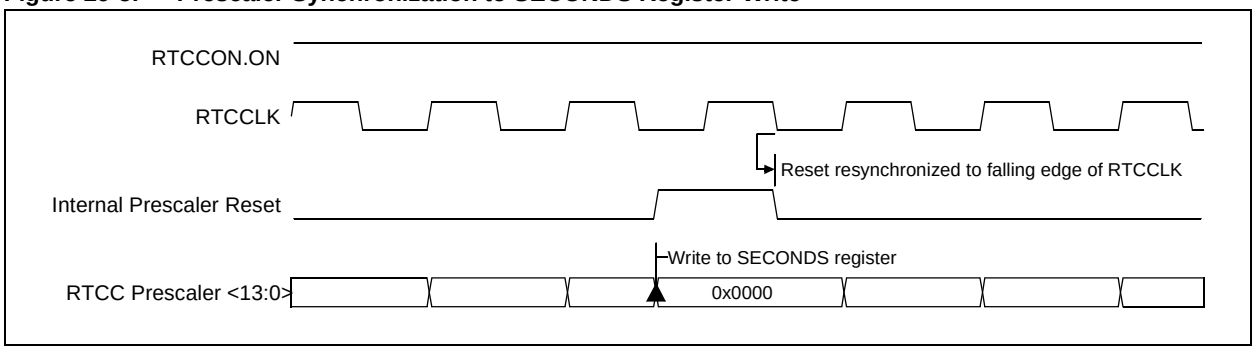
- POR
- Whenever ON = 0
- On a write to the SECONDS (RTCTIME<15:8>) register
- On the rising edge of the RTCC clock, when prescaler is 0x0000

See Figure 29-6 for the RTCSYNC bit timings.

29.3.8.2 Prescaler Reset Synchronization

A write to the SECONDS register (RTCTIME<15:8>) asynchronously resets the RTCC Prescaler (including the HALFSEC register). The Reset remains active until a falling edge of RTCCLK is detected (see Figure 29-5).

Figure 29-5: Prescaler Synchronization to SECONDS Register Write



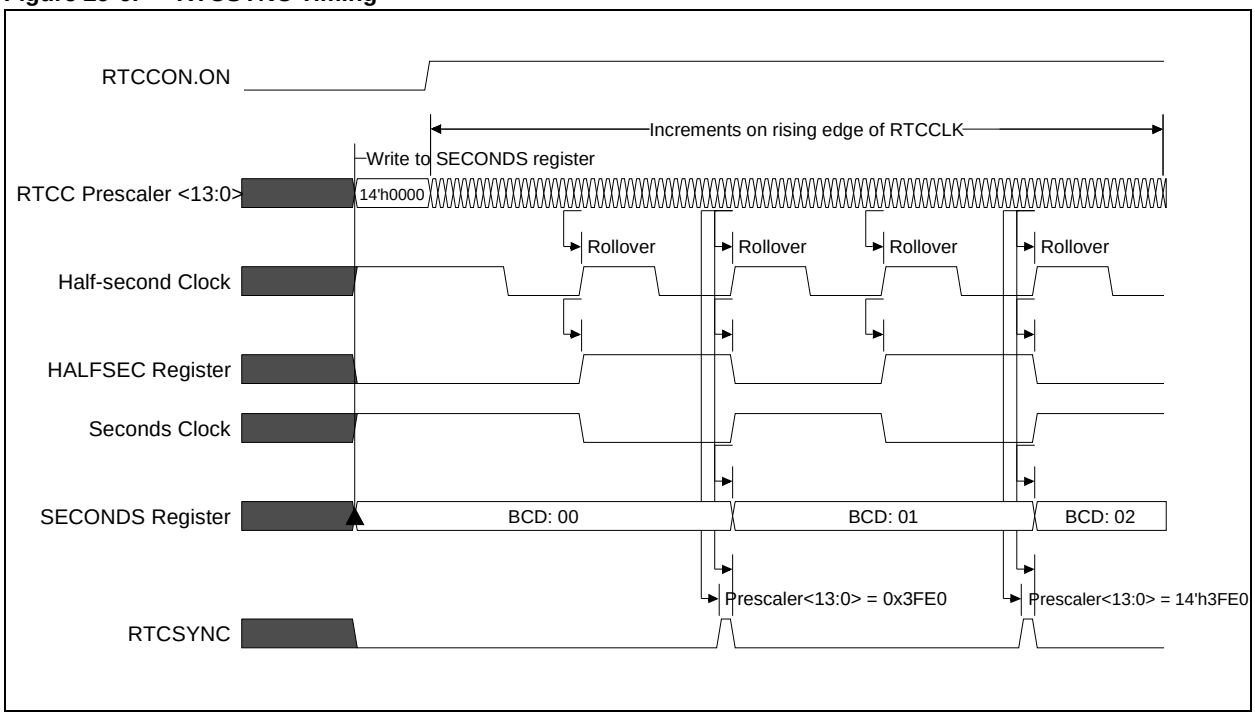
29.3.8.3 Gating Off the RTCC Clock

There are two conditions for which the internal RTCC clock is held off:

- ON (RTCCON<15>) = 0
- The device is in the DEBUG mode and FRZ (RTCCON<14>) = 1.

Stopping the RTCC clock does not affect reading and writing registers from the peripheral bus interface.

Figure 29-6: RTCSYNC Timing



29.3.9 Write Lock

In order to perform a write to any of the RTCC timer registers, the RTCWREN bit (RTCCON<3>) must be set. Setting of the RTCWREN bit is only allowed once the device level unlocking sequence has been executed. The unlocking sequence is as follows:

1. Load 0xAA996655 to CPU register X.
2. Load 0x556699AA to CPU register Y.
3. Load 0x00000008 to CPU register Z (the RTCWREN bit number).
4. Suspend or disable all Initiators that can access the Peripheral Bus and interrupt the unlock sequence. (i.e., DMA and Interrupts).
5. Store CPU register X to SYSKEY.
6. Store CPU register Y to SYSKEY.
7. Store CPU register Z to RTCCONSET.
8. Re-enable DMA and interrupts.

Note that steps 5 through 7 must be followed exactly to unlock RTCC write operations. If the sequence is not followed exactly, the RTCWREN bit will not be set.

Refer to Figure 29-3 for an assembly language implementation of the Write Unlock operation.

Example 29-3: Write Unlock Sequence

```
# assume interrupts are disabled
# assume the DMA controller is suspended
# assume the device is locked

#starting critical sequence
SYSKEY = 0xaa996655;                 // write first unlock key to SYSKEY
SYSKEY = 0x556699aa;                 // write second unlock key to SYSKEY
RTCCONSET = 0x8;                     // set RTCWREN in RTCCONSET
#end critical sequence

# re-enable interrupts
# re-enable the DMA controller
```

Note: To avoid accidental writes to the RTCC time values, it is recommended that the RTCWREN bit (RTCCON<3>) is kept clear at any other time. For RTCWREN to be set, there is only 1 instruction cycle time window allowed between the key1, key2 sequence and the setting of RTCWREN. Therefore, it is recommended that the code example in Example 29-3 be followed.

29.3.10 Calibration

The real-time crystal input can be calibrated using the periodic auto-adjust feature. When properly calibrated, the RTCC can provide an error of less than 0.66 seconds per month. Calibration has the ability to eliminate an error of up to 260 ppm.

The calibration is accomplished by finding the number of error clock pulses and writing this value into the CAL field of the RTCCON register (RTCCON<9:0>). This 10-bit signed value will be either added or subtracted from the RTCC timer once every minute. Refer to the steps below for RTCC calibration:

1. Using another timer resource on the device, the user must find the error of the 32.768 kHz crystal.
2. Once the error is known, it must be converted to the number of error clock pulses per minute.

Equation 29-1: Formula Box:

(Ideal Frequency (32,758) – Measured Frequency) * 60 = Error Clocks per Minute

3. a) If the oscillator is *faster* than ideal (negative result from step 2), the CAL register value needs to be negative. This causes the specified number of clock pulses to be subtracted from the timer counter once every minute.
   b) If the oscillator is *slower* than ideal (positive result from step 2), the CAL register value needs to be positive. This causes the specified number of clock pulses to be added to the timer counter once every minute.
4. Load the CAL register (RTCCON<9:0>) with the correct value.

Writes to the CAL register should only occur when the timer is turned off, or immediately after the rising edge of the seconds pulse (except when SECONDS (RTCTIME<15:8>) field is 00, due to the possibility of the auto-adjust event).

Notes:	It is up to the user to include in the error value the initial error of the crystal, drift due to temperature, and drift due to crystal aging. A write to the SECONDS register resets the state of calibration (not its value). If an adjustment just occurred, it will occur again because of the minute roll-over.

Example 29-4: Updating the RTCC Calibration Value

```
/*
    The following code example will update the RTCC calibration.
*/



    int cal=0x3FD;                // 10 bits adjustment, -3 in value

    if(RTCCON&0x8000)
    {                             // RTCC is ON
        unsigned intt0, t1;
        do
        {
            t0=RTCTIME;
            t1=RTCTIME;
        }while(t0!=t1);           // read valid time value
        if((t0&0xFF)==00)
        {                         // we're at second 00, wait auto-adjust to be performed
            while(!(RTCCON&0x2)); // wait until second half...
        }
    }

    RTCCONCLR=0x03FF0000;         // clear the calibration
    RTCCONSET=cal;
```


29.4 ALARM

The RTCC module provides an alarm function with the following features:

- Configurable from a half-second to one year
- Enabled using the ALRMEN bit (RTCALRM<15>)
- One-time alarm, repeat alarms, and indefinite repetition of the alarm available

29.4.1 Configuring the Alarm

The alarm feature is enabled using the ALRMEN bit.

The interval selection is made based on the settings of the alarm mask, AMASK (RTCALRM<11:8>). The AMASK bits determine which and how many digits of the alarm must match the clock value for the alarm to occur (see Figure 29-7).

Note:	Once the timer value reaches the alarm setting, one RTCC clock period will elapse prior to setting the alarm interrupt. The result is that, for a short period, the user will see the timer value at the alarm setting without the interrupt having gone off.

29.4.1.1 Configuring the One-Time Alarm

When the alarm is issued, with ARPT (RTCALRM<7:0>) = 0 and CHIME (RTCALRM<14>) = 0, the ALRMEN bit automatically clears.

Example 29-5: Configuring the RTCC for a One-Time One-Per-Day Alarm

```
/*
    The following code example will update the RTCC one-time alarm.
    Assumes the interrupts are disabled.
*/


    unsigned long alTime=0x16153300;// set time to 04 hr, 15 min, 33 sec
    unsigned long alDate=0x06102705;// set date to Friday 27 Oct 2006

                                    // turn off the alarm, chime and alarm repeats; clear
                                    // the alarm mask

    while(RTCALRM&0x1000);          // wait ALRMSYNC to be off
    RTCALRMCLR=0xCFFF;              // clear ALRMEN, CHIME, AMASK and ARPT;
    ALRMTIME=alTime;
    ALRMDATE=alDate;                // update the alarm time and date

    RTCALRMSET=0x8000|0x00000600;   // re-enable the alarm, set alarm mask at once per day
```

29.4.1.2 Configuring the Repeat Alarm

In addition to providing a one-time alarm, the module can be configured to repeat the alarm at a pre-configured interval. The ARPT register contains the number of times the alarm repeats after the alarm is enabled. When ARPT = 0 and CHIME = 0, the repeat function is disabled and only a single alarm pulse will be produced. The alarm can be generated up to 256 times by setting ARPT = 0xFF.

Each time, after the alarm is issued, the ARPT register is decremented by one. Once the register reaches 0, the alarm will be generated one last time; after which point, ALRMEN bit is cleared automatically and the alarm will turn off.

Example 29-6: Configuring the RTCC for a Ten-Times One-Per-Hour Alarm

```
/*
    The following code example will update the RTCC repeat alarm.
    Assumes the interrupts are disabled.
*/


    unsigned long alTime=0x23352300;  // set time to 23hr, 35 min, 23 sec
    unsigned long alDate=0x06111301;  // set date to Monday 13 Nov 2006

                                      // turn off the alarm, chime and alarm repeats; clear
                                      // the alarm mask
    while(RTCALRM&0x1000);            // wait ALRMSYNC to be off
    RTCALRMCLR=0xCFFF;                // clear the ALRMEN, CHIME, AMASK and ARPT;
    ALRMTIME=alTime;
    ALRMDATE=alDate;                  // update the alarm time and date
    RTCALRMSET=0x8000|0x0509;         // re-enable the alarm, set alarm mask at once per hour
                                      // for 10 times repeat
```

29.4.1.3 Configuring the indefinite alarm

To provide an indefinite repetition of the alarm, the Chime feature can be enabled using the CHIME (RTCALRM<14>) bit. When CHIME = 1, rather than disabling the alarm when the last repeat has been performed, the ARPT rolls over from 0x00 to 0xFF and continues counting indefinitely.

Example 29-7: Configuring the RTCC for Indefinite One-Per-Day Alarm

```
/*
    The following code example will update the RTCC indefinite alarm.
    Assumes the interrupts are disabled.
*/


    unsigned long alTime=0x23352300;  // set time to 23hr, 35 min, 23 sec
    unsigned long alDate=0x06111301;  // set date to Monday 13 Nov 2006

                                      // turn off the alarm, chime and alarm repeats; clear
                                      // the alarm mask
    while(RTCALRM&0x1000);            // wait ALRMSYNC to be off
    RTCALRMCLR=0xCFFF;                // clear ALRMEN, CHIME, AMASK, ARPT;
    ALRMTIME=alTime;
    ALRMDATE=alDate;                  // update the alarm time and date
    RTCALRMSET=0xC600;                // re-enable the alarm, set alarm mask at once per
                                      // hour, enable CHIME
```


Figure 29-7: Alarm Mask Settings

Alarm Mask Setting AMASK<3:0>	Day of the Week	Month	Day	Hours	Minutes	Seconds
0000 – Every half second 0001 – Every second						
0010 – Every 10 seconds						s
0011 – Every minute						s s
0100 – Every 10 minutes					m	s s
0101 – Every hour					m m	s s
0110 – Every day				h h	m m	s s
0111 – Every week	d			h h	m m	s s
1000 – Every month			d d	h h	m m	s s
1001 – Every year(1)		m m	d d	h h	m m	s s

Note 1: Annually, except when configured for February 29.

29.4.2 Alarm Interrupt

The alarm event is generated when the RTCC timer matches the alarm registers. The match must only occur on the unmasked portion of the time/date registers, according to the AMASK (RTCALRM<11:8>) bit settings (see Figure 29-7).

At every alarm event, an interrupt is generated. In addition, an alarm pulse output is provided that operates at half the frequency of the alarm. This output is completely synchronous to the RTCC clock and can be used as a trigger clock to other peripherals. This output is available on the RTCC pin. The output pulse is a clock with a 50% duty cycle and a frequency half that of the alarm event (see Figure 29-8). The alarm must be enabled for the pulse to be active, ALRMEN (RTCALRM<15>) = 1. The initial value of the alarm pulse at the RTCC output pin is programmable using the PIV bit (RTCALRM<13>).

The RTCC pin is also capable of outputting the seconds clock. The user can select between the alarm pulse – generated by the RTCC module, or the seconds clock output. The RTSECSEL bit (RTCCON<7>) selects between these two outputs. When RTSECSEL = 0, the alarm pulse is selected. When RTSECSEL = 1, the seconds clock is selected (see Figure 29-9).

Note:	Changing any of the alarm time, date and alarm registers, other then the RTCOE (RTCCON<0>), while the alarm is enabled (ALRMEN = 1), can result in a false alarm event leading to a false alarm interrupt. To avoid a false alarm event and to perform a safe write to the alarm registers, the timer and alarm values should only be changed while the RTCC is disabled (RTCCON<15>=0), or when ALRMSYNC (RTCALRM<12>) = 0.

Figure 29-8: Alarm Event Generation

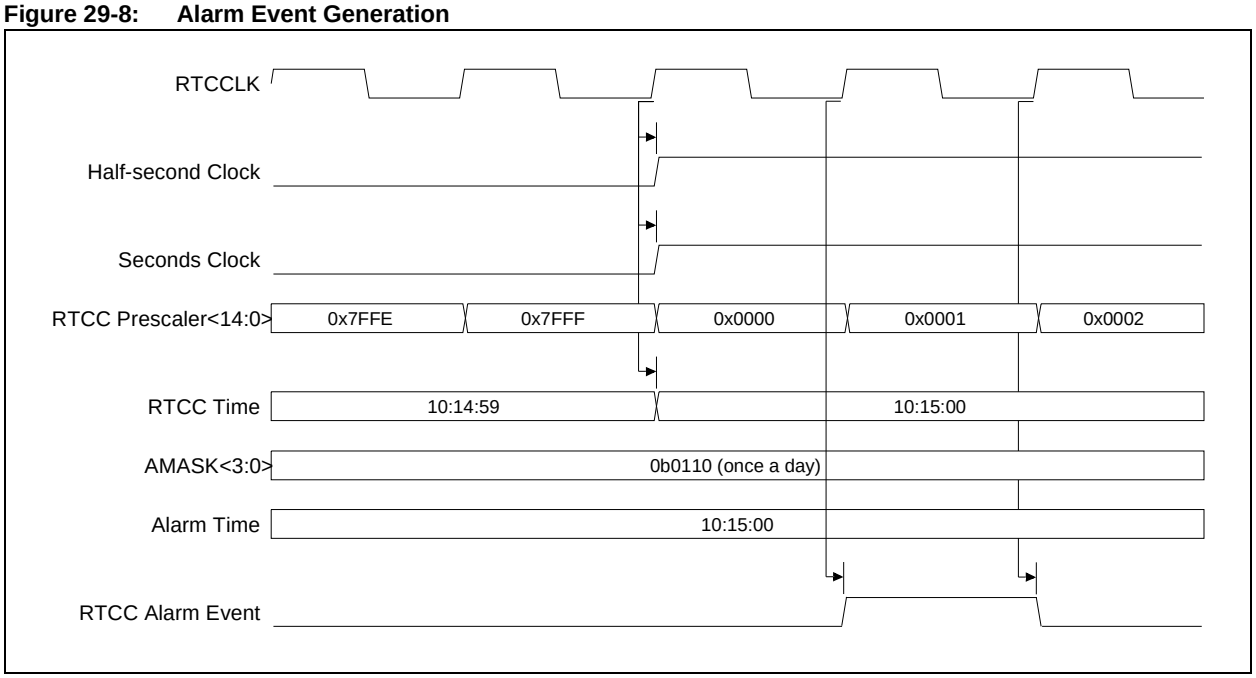
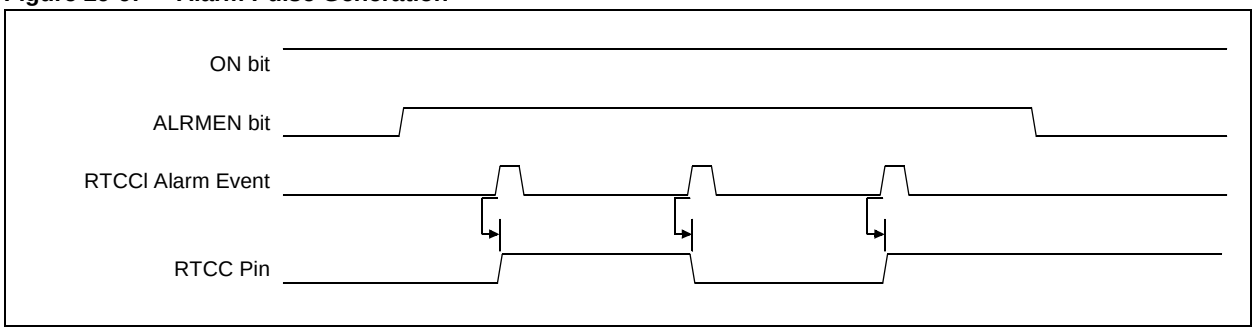


Figure 29-9: Alarm Pulse Generation



29.5 INTERRUPTS

The RTCC device has the ability to generate interrupts reflecting the alarm event that occurs when the RTCC timer matches the alarm registers. The match occurs on the unmasked portion of the time/date registers according to the AMASK (RTCALRM<11:8>) bit settings.

At every alarm event, an interrupt can be generated:

- The alarm interrupt is signalled by RTCCIF (IFS1<15>). This interrupt flag must be cleared in software.

In order to enable the RTCC interrupts, use the respective RTCC interrupt enable bit:

- RTCCIE (IEC1<15>).

The interrupt priority level bit and interrupt subpriority level bit must be also be configured:

- RTCCIP (IPC8<28:26>), RTCCIS (IPC8<25:24>)

Refer to **Section 8. "Interrupts"** for further details.

29.5.1 Interrupt Configuration

The RTCC has one dedicated interrupt flag bit RTCCIF and a corresponding interrupt enable/mask bit RTCCIE. RTCCIE is used to enable or disable the RTCC interrupt source. There is one specific RTCC interrupt vector.

The RTCCIF is set when the RTCC alarm registers match the RTCC time registers.

Note that the RTCCIF bit will be set without regard to the state of the corresponding enable bit. The IF bit can be polled by software if desired.

The RTCCIE bit is used to define the behavior of the Interrupt Controller (INT) when the corresponding RTCCIF bit is set. When the RTCCIE bit is clear, the INT module does not generate a CPU interrupt for the event. If the RTCCIE bit is set, the INT module will generate an interrupt to the CPU when the RTCCIF bit is set (subject to the priority and subpriority as outlined below).

It is the responsibility of the user's software routine that services a particular interrupt to clear the appropriate interrupt flag bit before the service routine is complete.

The priority of the RTCC peripheral can be set with the RTCCIP<2:0> bits. This priority defines the priority group to which the interrupt source will be assigned. The priority groups range from a value of 7 (the highest priority) to a value of 0 (which does not generate an interrupt). An interrupt being serviced will be preempted by an interrupt in a higher priority group.

The subpriority bits allow setting the priority of an interrupt source within a priority group. The values of the subpriority RTCCIS<1:0> range from 3 (the highest priority) to 0, the lowest priority. An interrupt within the same priority group, but having a higher subpriority value, will not preempt a lower subpriority interrupt that is in progress.

The priority group and subpriority bits allow more than one interrupt source to share the same priority and subpriority. If simultaneous interrupts occur in this configuration, the natural order of the interrupt sources within a priority/subpriority group pair determine the interrupt generated. The natural priority is based on the vector numbers of the interrupt sources. The lower the vector number the higher the natural priority of the interrupt. Any interrupts that were overridden by natural order will then generate their respective interrupts based on priority, subpriority, and natural order after the interrupt flag for the current interrupt is cleared.

After an enabled interrupt is generated, the CPU will jump to the vector assigned to that interrupt. The vector number for the interrupt is the same as the natural order number. The CPU will then begin executing code at the vector address. The user's code at this vector address should perform any application specific operations and clear the RTCCIF (IFS1<15>) interrupt flag, and then exit. Refer to the vector address table details in the **Section 8. "Interrupts"** for more information on interrupts.

Table 29-4: RTCC Interrupt Vector for Various Offsets with EBASE = 0x8000:0000

Interrupt	Vector/Natural Order	IRQ Number	Vector Address IntCtl.VS = 0x01	Vector Address IntCtl.VS = 0x02	Vector Address IntCtl.VS = 0x04	Vector Address IntCtl.VS = 0x08	Vector Address IntCtl.VS = 0x10
RTCC Alarm	35	47	8000 0660	8000 0AC0	8000 1380	8000 2500	8000 4800

Example 29-8: RTCC Initialization with Interrupts Enabled Code Example

```
/*
    The following code example illustrates an RTCC initialization with interrupts enabled.
    When the RTCC alarm interrupt is generated, the cpu will jump to the vector assigned to
    RTCC interrupt.
*/
                                  // assume RTCC write is enabled i.e. RTCWREN (RTCCON<3>) =1;
    IEC1CLR=0x00008000;           // disable RTCC interrupts

    RTCCONCLR=0x8000;             // turn off the RTCC
    while(RTCCON&0x40);           // wait for clock to be turned off

    IFS1CLR=0x00008000;           // clear RTCC existing event
    IPC8CLR=0x1f000000;           // clear the priority
    IPC8SET=0x0d000000;           // Set IPL=3, subpriority 1
    IEC1SET=0x00008000;           // Enable RTCC interrupts

    RTCTIME=0x16153300;           // safe to update time to 16 hr, 15 min, 33 sec
    RTCDATE=0x06102705;           // update the date to Friday 27 Oct 2006

    RTCALRMCLR=0xCFFF;            // clear ALRMEN, CHIME, AMASK and ARPT;
    ALRMTIME=0x16154300;          // set alarm time to 16 hr, 15 min, 43 sec
    ALRMDATE=0x06102705;          // set alarm date to Friday 27 Oct 2006

    RTCALRMSET=0x8000|0x00000600; // re-enable the alarm, set alarm mask at once per day

    RTCCONSET=0x8000;             // turn on the RTCC
    while(!(RTCCON&0x40));        // wait for clock to be turned on
```

Example 29-9: RTCC ISR Code Example

```
/*
    The following code example demonstrates a simple interrupt service routine for RTCC
    interrupts. The user's code at this vector should perform any application specific
    operations and must clear the RTCC interrupt flag before exiting.
*/

void__ISR(_RTCC_VECTOR, ipl3) __RTCCInterrupt(void)
{
                                      // ... perform application specific operations
                                      // in response to the interrupt

    IFS1CLR=0x00008000;               // be sure to clear RTCC interrupt flag
                                      // before exiting the service routine.
}
```

Note: The RTCC ISR code example shows MPLAB® C32 C compiler specific syntax. Refer to your compiler manual regarding support for ISRs.

29.6 OPERATION IN POWER-SAVING AND DEBUG MODES

Note: In this manual, a distinction is made between a power mode as it is used in a specific module, and a power mode as it is used by the device, e.g., sleep mode of the Comparator and SLEEP mode of the CPU. To indicate which type of power mode is intended, uppercase and lowercase letters (Sleep, Idle, Debug) signify a module power mode, and all uppercase letters (SLEEP, IDLE, DEBUG) signify a device power mode.

29.6.1 RTCC Operation in SLEEP Mode

When the device enters SLEEP mode, the system clock is disabled. The RTCC and alarm continue to operate while in SLEEP mode. The operation of the alarm is not affected by SLEEP. An alarm event can wake-up the CPU if the alarm interrupt has a higher priority than the current CPU IPL.

29.6.2 RTCC Operation in IDLE Mode

When the device enters IDLE mode, the system clock sources remain functional. The RTCC and alarm continue to operate while in IDLE mode. The operation of the alarm is not affected by IDLE. An alarm event can wake-up the CPU if the alarm interrupt has a higher priority than the current CPU IPL.

Bit SIDL (RTCCON<13>) selects the behavior of the module IDLE mode.

- If SIDL = 1, the PBCLK to the RTCC will be disabled. The PBCLK is the clock source for the AMASK (RTCALRM<11:8), CHIME (RTCALRM<14>), ALRMTIME, ALRMDATE and all of the synchronizers that provide the read data for RTCTIME, and some other bits like ALRMSYNC (RTCALRM<12>), ALRMEN (RTCALRM<15>) and RTCSYNC (RTCCON<2>). Thus, the SIDL functionality can be used to reduce the RTCC power consumption without affecting the functionality of the RTCC.
- If SIDL = 0, the module will continue normal operation in IDLE mode.

29.6.3 RTCC Operation in DEBUG Mode

The FRZ bit (RTCCON<14>) determines whether the RTCC module will run or stop while the CPU is executing Debug Exception code (i.e., the application is halted) in DEBUG mode. When FRZ = 0, the RTCC module continues to run even when the application is halted in DEBUG mode. When FRZ = 1 and the application is halted in DEBUG mode, the module will freeze its operations and make no changes to the state of the RTCC module. The Prescaler and RTCC timers will not increment. If a Configuration register normally causes the state of the module to change on a read, that functionality is disabled during Freeze. The module will resume its operation after the CPU resumes execution.

Note: The FRZ bit is readable and writable only when the CPU is executing in Debug Exception mode. In all other modes, the FRZ bit reads as '0'. If FRZ bit is changed during DEBUG mode, the new value does not take effect until the current Debug Exception mode is exited and re-entered. During the Debug Exception mode, the FRZ bit reads the state of the peripheral when entering DEBUG mode.

29.7 EFFECTS OF VARIOUS RESETS

29.7.1 Device Reset

When a device Reset occurs, the RTCALRM register is forced to its Reset state, causing the alarm to be disabled (if enabled prior to the Reset). Note, however, that if the RTCC is enabled, it will continue to operate when a device Reset occurs.

29.7.2 Power-on Reset

The RTCTIME and RTCDATE registers are not affected by the POR. A POR forces the device to its inactive state. Once the device exits the POR state, the clock registers should be reloaded with the desired values.

The timer prescaler values can only be reset by writing to the SECONDS register (RTCTIME<15:8>). No device Reset can affect the prescalers.

29.7.3 Watchdog Timer Reset

The Watchdog Timer Reset is equivalent to the device Reset.

29.7.4 Effects of the ON Bit

When ON bit (RTCCON<15>) = 0, the bits RTCSYNC (RTCCON<2>), HALFSEC (RTCCON<1>), and ALRMSYNC (RTCALRM<4>) are asynchronously reset and held in Reset. Also, the RTCC pin output is determined by the RTCOE bit (RTCCON<0>), gated by the ON bit.

29.8 PERIPHERALS USING RTCC MODULE

There are no other peripheral modules using the RTCC device.

29.9 I/O PIN CONTROL

Enabling the RTCC modules configures the I/O pin direction. When the RTCC module is enabled, configured and the output enabled, the I/O pin direction is properly configured as a digital output.

Table 29-5: I/O Pin Configuration for use with RTCC Module

Required Settings for Module Pin Control							
IO Pin Name	**Required**	**Module Control**	**Bit Field**	**TRIS(4)**	**Pin Type**	**Buffer Type**	**Description**
RTCC	Yes(1)	ON and RTCOE(2)	RTSECSEL = 1	X	O	CMOS	RTCC Seconds Clock
RTCC	Yes(1)	ON and RTCOE(2)	RTSECSEL = 0 and ALRMEN and PIV(3)	X	O	CMOS	RTCC Alarm Pulse

Legend: CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output

Note 1: The RTCC pin is only required when Seconds Clock or Alarm Pulse output is needed. Otherwise, this pin can be used for general purpose IO and require the user to set the corresponding TRIS control register bit.

2: The ON (RTCCON<15>) and RTCOE (RTCCON<0>) bits are always required to validate the output function of the RTCC pin, either Seconds Clock or Alarm Pulse.

3: When RTSECSEL (RTCCON<7>) = 0, the RTCC pin output is the Alarm Pulse. If the ALRMEN (RTCALRM<15>) = 0, PIV (RTCALRM<13>) selects the value at the RTCC pin. When the ALRMEN = 1 the RTCC pin reflects the state of the Alarm Pulse.

4: The setting of the TRIS bit is irrelevant.

29.10 DESIGN TIPS

Question 1: ***If I do not use the RTCC output for my RTCC module, is this I/O pin available as a general purpose I/O pin?***

Answer: Yes. If you are not interested in outputting the Seconds Clock or the Alarm Pulse, you can use the RTCC pin as a general I/O pin as long as RTCC output is disabled (RTCCON<0>=0). Note that when used as a general purpose I/O pin, the user is responsible for configuring the respective data direction register (TRIS) for input or output.

Question 2: ***How do I make sure that when reading the RTCC time value I get the proper value, not affected by rollover (seconds to minutes, minutes to hours)?***

Answer: The easiest way to read the proper current time is to perform a double read of the RTCTIME register. The right time value is obtained when two consecutive readings are identical. The same is true when reading the RTCDATE register.

Question 3: ***Is the week of the day automatically calculated by the RTCC device when I set in a specific date, like 18 Jan 2006?***

Answer: No, the device doesn't perform this calculation automatically. When writing the RTCDATE register, you have to provide a valid value for the WDAY01 field (RTCDATE<3:0>), i.e., 3 for Wednesday, 18 Jan. However, from that point on, the RTCC device takes care of updating the day of the week field properly.

Question 4: ***Does the device perform the leap years calculation automatically or do I have to perform some corrections in the RTCC date?***

Answer: The RTCC device automatically performs leap year detection. No updates are necessary in the year range 2000 to 2099. However, when you program the date to the RTCDATE register you must enter a valid date. For example, don't program the RTCC with the date, 29 Feb 2007.

Question 5: ***Can I freely write to the RTCTIME and RTCDATE registers to update the current time or date?***

Answer: In short, no, you cannot write directly to the RTCTIME and RTCDATE registers. Actually, if the RTCC is disabled (RTCCON<15> = 0), you could update the time and date values at any time. However, if the RTCC is ON, further precautions must be taken. There is a safe window when the writes to the Time and Date registers are safely performed. That window is indicated by the RTCSYNC bit (RTCCON<2>). Any update to RTCTIME or RTCDATE registers should occur when RTCSYNC = 0. (The hardware actually ignores a write to the TIME and DATE registers that occurs during a rollover event when the RTCC clock is high, so the write operation could go undetected). Even more, if the alarm is enabled and the AMASK is half-second (RTCALRM<11:8>) any update to the RTCTIME and RTCDATE should occur when the ALRMSYNC is 0 (RTCALRM<12>). This ensures a proper functioning of the alarm trigger mechanism, without spurious alarm events being generated.

Notes:	You have to have the RTCWREN (RTCCON<3>) set in order to be able to update the RTCTIME and RTCDATE registers. Normally you should disable interrupts when trying to update the RTCTIME. If not, you cannot be sure that the write operation to the RTCTIME register occurs when RTCSYNC/ALRMSYNC is deasserted. Another way to perform the update of the system time and date is to turn off the RTCC, perform the write operations and then turn RTCC on again.

Question 6: ***Can I write to the ALRMTIME and ALRMDATE registers freely to update the alarm time or date?***

Answer: In short, no, you cannot update the Alarm Time and Date registers directly. The following steps are necessary:

- If the RTCC is disabled, i.e., ON = 0 (RTCCON<15>), you can perform the write to the ALRMTIME and ALRMDATE at any time.

Else, the write can occur only when ALRMSYNC = 0 (RTCALRM<12>).

Notes:	Normally you should disable interrupts when trying to update the alarm time and date. If not, you cannot be sure the write operation to the ALRMTIME/ALRMDATE registers occurs when ALRMSYNC is deasserted. Another way to perform the update of the alarm time and date is to turn off the RTCC, perform the write operations and then turn RTCC on again. Note that this approach has an impact on the timing accuracy, since the RTCC will not count while it is stopped. The exact same approach applies to the writable fields of the RTCALRM register: CHIME (RTCALRM<14>), AMASK (RTCALRM<11:8>), ALRMEN (RTCALRM<15>), ARPT (RTCALRM<7:0>) and PIV (RTCALRM<13>). If the RTCC is ON, the write should occur only when ALRMSYNC = 0.

Question 7: ***Can I toggle freely the RTCWREN bit in the RTCCON register?***

Answer: You can always clear the RTCWREN bit (RTCCON<3>). However, in order to enable the write to the RTCCTIME and RTCCDATE register, a proper sequence of operations must be performed. Refer to the **Section 29.3.9 "Write Lock"** for details.

29.11 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the PIC32MX device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Real Time Clock and Calendar (RTCC)) module are:

Title	Application Note #
No related application notes at this time	N/A

Note: Visit the Microchip web site (www.microchip.com) for additional application notes and code examples for the PIC32MX family of devices.

29.12 REVISION HISTORY

Revision A (October 2007)

This is the initial released version of this document.

Revision B (October 2007)

Updated document to remove Confidential status.

Revision C (April 2008)

Revised status to Preliminary; Revised U-0 to r-x.

Revision D (June 2008)

Revised Registers 29-1, bit 14; Revised Registers 29-26, 29-27, Footnote; Revised Examples 29-1 and 29-9; Change Reserved bits from “Maintain as” to “Write”; Added Note to ON bit (RTCCON Register).

