PCF2127 RTC Ardunio Library

Eric Dinger

November 29, 2012
Version 1.0

1 Overview

This document is the design document for an arduino library for the NXP RTC PFC2127a. It will start with an overview of the device registers and the plan for them in the noted version of the library. After the register there will be an interface definiton for the class, along with requirments and return types, along with a definition of the associated data structure for the RTC. The RTC can do either I^{2}C or SPI communication, but this library will only implement SPI.

2 Device Registers

2.1 Control Registers

This version of the library will need to control some of bits in all of the control registers, so it should go ahead and be able to manage them all, including dealing with the bits that are uneeded due to not implementing other functions (timestamp, etc. . .).

2.1.1 Control 1 Address: 0x00

Bit 7: External test mode
Bit 6: Unused
Bit 5: RTC Source clock running or not, clkout unaffected
Bit 4: Timestamp interrupt 1
Bit 3: Power on reset override
Bit 2: 12/24 hour mode
Bit 1: Minute interrupt
Bit 0: Second interrupt

2.1.2 Control 2 Address: 0x01

Bit 7: Minute or second interrupt generated
Bit 6: Watchdog flag
Bit 5: Timestamp interrupt 2
Bit 4: Alarm interrupt, set when the alarm conditions are met.
Bit 3: Countdown timer interrupt
Bit 2: Timestamp interrupt enable
Bit 1: Alarm interrupt enable
Bit 0: Countdown timer interupt enable

2.1.3 Control 3

Bit [7-5]: Power managment control
Bit 4: Battery switch over timestamp
Bit 3: Battery switch over interrupt
Bit 2: Battery status
Bit 1: Battery flag interrupt enable
Bit 0: Battery low interrupt enable

Unsure if the standard power managment will be correct, this decision hinges on the extra power fail detection function. Need to get the actual hardware and test on it.

2.2 Time and Date Registers

The ability read all of the time and date registers will be included in the library. Special attention needs to be payed to Seconds and the oscilator stop flag bit in it.

2.2.1 Seconds Address: 0x03

Bit 7: Oscilator stop flag
Bit [6-0] Seconds

2.2.2 MinutesAddress: 0x04

Nothing unusual

2.2.3 Hours Address: 0x05

Bit [7-6]: Unused
12 Hour Mode
Bit 5: AM/PM
Bit [4-0]: Hours
24 Hour Mode
Bit [5-0]: Hours

2.2.4 Days Address: 0x06

Nothing unusual

2.2.5 Weekdays Address: 0x07

Nothing unusual

2.2.6 Months Address: 0x08

Nothing unusual

2.2.7 Years Address: 0x09

2.3 Alarm Registers

For all Alarm registers bit 7 is the enable bit for the alarm.

2.3.1 Second Alarm Address: 0x0A

2.3.2 Minute Alarm Address: 0x0B

2.3.3 Hour Alarm Address: 0x0C

Varies depending on 12/24 hour mode, same as the hour register.

2.3.4 Day Alarm Address: 0x0D

2.3.5 Weekday Alarm Address: 0x0E

2.4 CLKOUT Control Register Address: 0x0F

Bit [7-6]: Temperature measurement period
Bit [5-3]: Unused
Bit [2-0]: CLKOUT frequency selection

This will be included in this version of the library. See the tables on pages 12 and 13 of the data sheet for the values.

2.5 Watchdog Registers

Watchdog will not be implemented in this version of the library.

2.6 Timestamp Registers

Timestamp will not be implemented in this version of library.

2.7 Aging Offset Register Address: 0x19

Bit [7-4]: Unused
Bit [3-0]: Againg offset value
Adjusting the aging offset will not be included in this version of the library.

2.8 Ram Register

Ram writing will not be in this version of this library. I know this a big draw to this RTC chip, but this is orthoginal to the rest of the design work, and can easily be added later.

3 Class Design

3.1 Data Structure

To conserve memory only the time and date will be kept in memory. All of the control registers will be read from the device and modified. More to come.

3.2 Public interface

Note: All of the setter methods will have the same operational behavior. They will all read the current value from the RTC then clear and set the desired bits and write out the new value.

3.2.1 Constructor

Type: Void
Arguments: CS (Chip select) pin.
Return: Void

As much as I would like the constructor to be able to read the RTC's status when the class is initialized, I can't do this because it is common for classes to be declared globaly in the Arduino enviroment, and it is quite likely that SPI won't be up and running at this time. So the constructor will zero out the time and date.

3.2.2 Begin

Type: Enum
Arguments: None
Return: Enum of an error value
Here we read the control registers and the seconds register, we check the battery status bit and the OSF bit, and the stop bit. If any are set, we return the error, otherwise we return an all clear.

3.2.3 GetTime

Type:
Arguments:
Return:

3.2.4 SetTime

Type:
Arguments:
Return:

3.2.5 GetDate

Type:
Arguments:
Return:

3.2.6 SetDate

Type:
Arguments:

Return:

3.2.7 SetTempPeriod

Type: void
Arguments: enum period for temp measuremnt
Return: void
Will take an enum, see table on page 12 of datasheet for values.

3.2.8 SetCLKOUT

Type: void
Arguments: enum of CLKOUT frequency. Return: void
Will take an enum, see table on page 13 of datasheet.

3.2.9 GetRegister

Type: uint8_t
Arguments: Register address
Return: uint8_t containg the requested register contents
The reason this is made public is in this version we aren't going to have getters for every thing, so if we wanted to do one of those operations they would use this.

3.2.10 SetRegister

Type: void
Arguments: address, uint8_t with register contents
Return: void
The reason this is made public is in this version we aren't going to have setters for every thing, so if we wanted to do one of those operations they would use this.

3.3 Private interface