tronixstuff.com

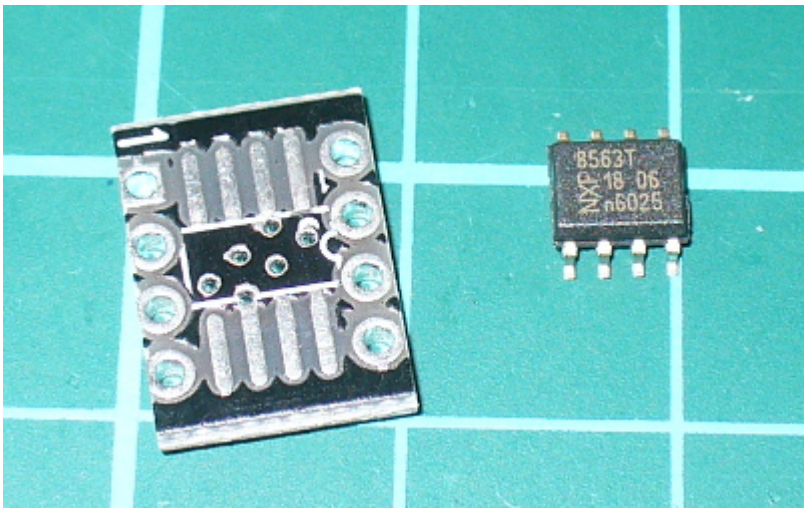
fun and learning with electronics since 2010

Tutorial – Arduino and PCF8563 real time clock IC

Another option for real-time clock ICs is the PCF8563 real-time clock IC from NXP – so this is a tutorial on how to use it for time, date, alarm clock and square-wave generation purposes.

The PCF8563 is another inexpensive RTC that can be used with an Arduino or other platforms due to the wide operating voltage (1 to 5.5V DC), I2C interface, and very low power consumption (when powered by a backup battery it only draws 0.25 µA). If you aren't up to speed on the I2C interface, please review the I2C tutorials before moving forward. And please download the data sheet (.pdf).

The PCF8563 is available in various chip packages, for the curious we're using the TSSOP8 version mounted on a breakout board:



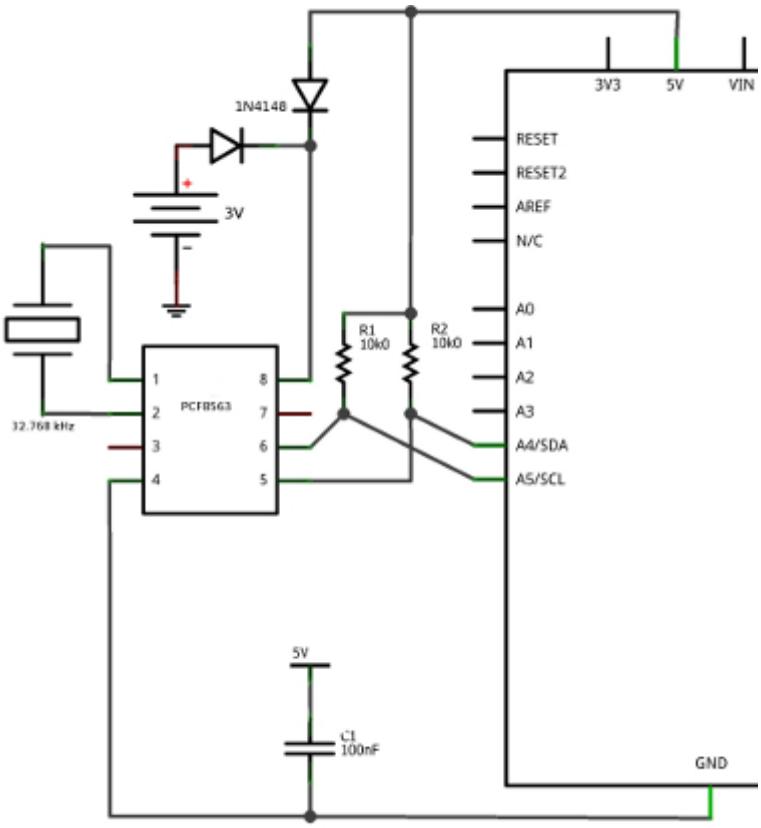
Don't panic – you can also get it in a breadboard-friendly DIP (through-hole) package as well, and also on a pre-built module from the usual suspects.

Demonstration Circuit

If you have a pre-made module, you can skip to the next section. However if you're making up the circuit yourself, you will need:

- One 0.1 uF capacitor

And here's the schematic:



* You can skip the diodes and battery if you don't want a backup power supply when the main power is turned off or removed. Pin 3 is for the interrupt output (we'll consider that later) and pin 7 is for the square-wave oscillator output.

Communicating with the PCF8563

Now to get down into the land of I2C once more. When looking through the data sheet NXP mentions two bus addresses, which have the same 7-bits finished with either a 1 for read or 0 for write. However you can just bitshift it over one bit as we don't need the R/W bit – which gives you a bus address of 0x51.

Next you need to know which registers store the time and date – check the register map (table 4) on page 7 of the data sheet:

Address	Register name	Bit							
		7	6	5	4	3	2	1	0
Control and status registers									
00h	Control_status_1	TEST1	N	STOP	N	TESTC	N	N	N
01h	Control_status_2	N	N	N	TI_TP	AF	TF	AIE	TIE
Time and date registers									
02h	VL_seconds	VL	SECONDS (0 to 59)						
03h	Minutes	x	MINUTES (0 to 59)						
04h	Hours	x	x	HOURS (0 to 23)					

and date start from 0x02. And one more thing – data is stored in the BCD (binary-coded-decimal) format.

But don't panic, we have a couple of functions to convert numbers between BCD and decimal.

Writing the time and date is a simple matter of collating the seconds, minutes, hours, day of week, day of month, month and year into bytes, converting to BCD then sending them to the PCF8563 with seven *Wire.write()* functions. Reading the data is also easy, just set the pointer to 0x02 and request seven bytes of data – then run them through a BCD to decimal conversion. With a catch.

And that catch is the need to sort out unwanted bits. Revisit table 4 in the data sheet – if you see an *x* that's an unused bit. If any of them are a 1 they will mess up the BCD-decimal conversion when reading the register, so they need to be eliminated just like a whack-a-mole. To do this, we perform an & (bitwise AND) operation on the returned byte and mask out the unwanted bits with a zero. How does that work?

Example – the byte for dayOfMonth is returned – we only need bits 5 to 0. So 6 and 7 are superfluous. If you use (dayOfMonth & B00111111) the & function will set bits 6 and 7 to zero, and leave the other bits as they were.

Now to put all that together in a demonstration sketch. It puts everything mentioned to work and simply sets the time to the PCF8563, and then returns it to the serial monitor. The data is kept in global variables declared at the start of the sketch, and the conversions between BCD and decimal are done "on the fly" in the functions used to send or retrieve data from the PCF8563. Read through the following sketch and see how it works for yourself:

```
// Example 54.1 - PCF8563 RTC write/read demonstration

#include "Wire.h"
#define PCF8563address 0x51

byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
String days[] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"
};

byte bcdToDec(byte value)
{
  return ((value / 16) * 10 + value % 16);
}

byte decToBcd(byte value){
  return (value / 10 * 16 + value % 10);
}

void setPCF8563()
// this sets the time and date to the PCF8563
```

```
  Wire.write(decToBcd(minute));
  Wire.write(decToBcd(hour));
  Wire.write(decToBcd(dayOfMonth));
  Wire.write(decToBcd(dayOfWeek));
  Wire.write(decToBcd(month));
  Wire.write(decToBcd(year));
  Wire.endTransmission();
}

void readPCF8563()
// this gets the time and date from the PCF8563
{
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x02);
  Wire.endTransmission();
  Wire.requestFrom(PCF8563address, 7);
  second     = bcdToDec(Wire.read() & B01111111); // remove VL error bit
  minute     = bcdToDec(Wire.read() & B01111111); // remove unwanted bits from MSB
  hour       = bcdToDec(Wire.read() & B00111111);
  dayOfMonth = bcdToDec(Wire.read() & B00111111);
  dayOfWeek  = bcdToDec(Wire.read() & B00000111);
  month      = bcdToDec(Wire.read() & B00011111);  // remove century bit, 1999 is over
  year       = bcdToDec(Wire.read());
}

void setup()
{
  Wire.begin();
  Serial.begin(9600);
  // change the following to set your initial time
  second = 0;
  minute = 28;
  hour = 9;
  dayOfWeek = 2;
  dayOfMonth = 13;
  month = 8;
  year = 13;
  // comment out the next line and upload again to set and keep the time from resetting every
reset
  setPCF8563();
}

void loop()
{
  readPCF8563();
  Serial.print(days[dayOfWeek]);
```

```
    Serial.print("/20");
    Serial.print(year, DEC);
    Serial.print(" - ");
    Serial.print(hour, DEC);
    Serial.print(":");
    if (minute < 10)
    {
      Serial.print("0");
    }
    Serial.print(minute, DEC);
    Serial.print(":");
    if (second < 10)
    {
      Serial.print("0");
    }
    Serial.println(second, DEC);
    delay(1000);
  }
```

And a quick video of this in operation:

If all you need to do is write and read the time with the PCF8563, you're ready to go. However there's a few more features of this unassuming little part which you might find useful, so at least keep reading...

Square-wave output

the LED, then connect the cathode to pin 7 of the PCF8563.

The frequency is controlled from the register at 0x0D. Simply write one of the following values for the respective frequencies:

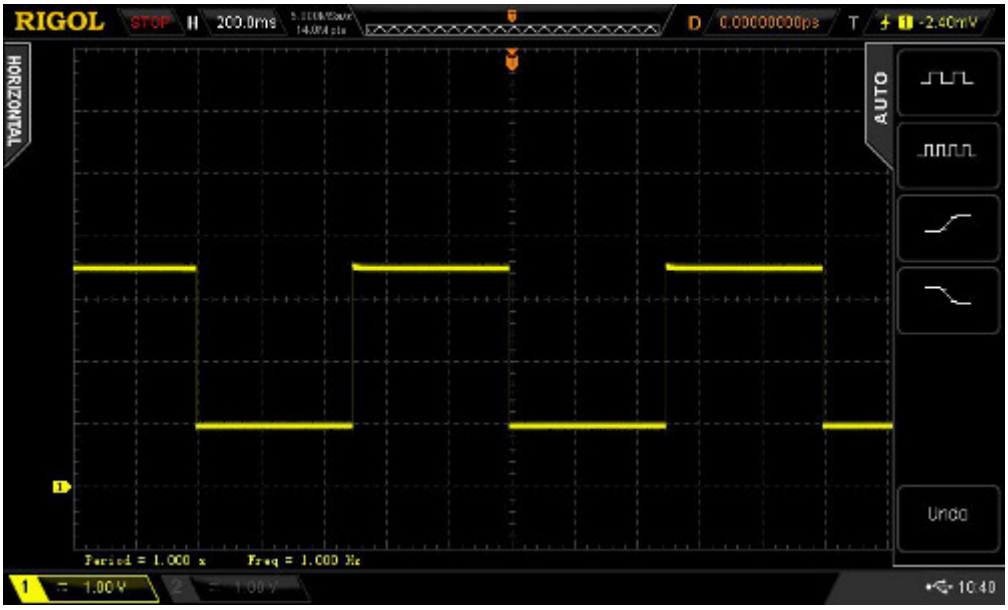
- 10000000 for 32.768 kHz;
- 10000001 for 1.024 kHz;
- 10000010 for 32 kHz;
- 10000011 for 1 Hz;
- 0 turns the output off and sets it to high impedance.

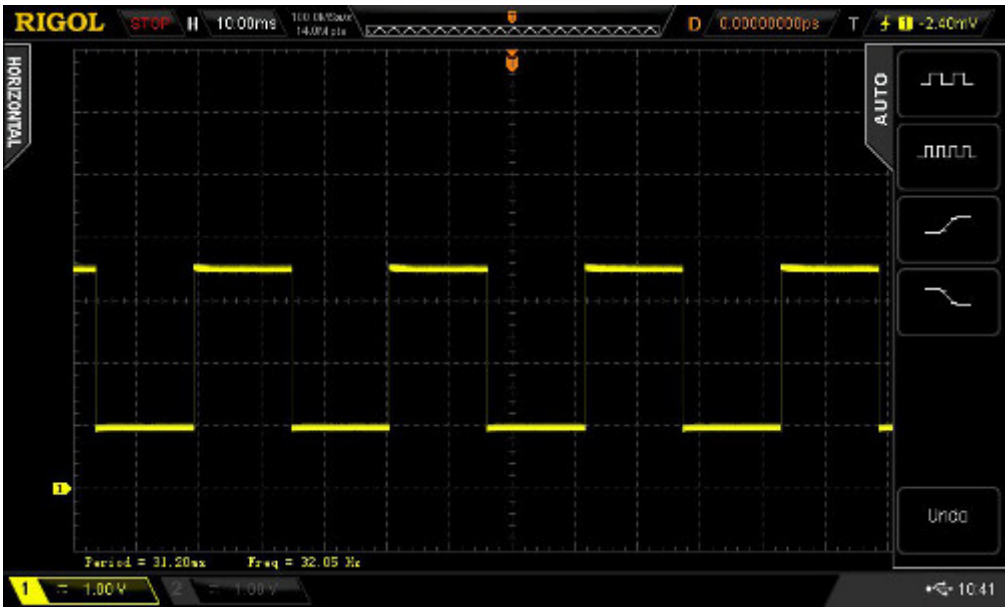
The following is a quick demonstration sketch which runs through the options:

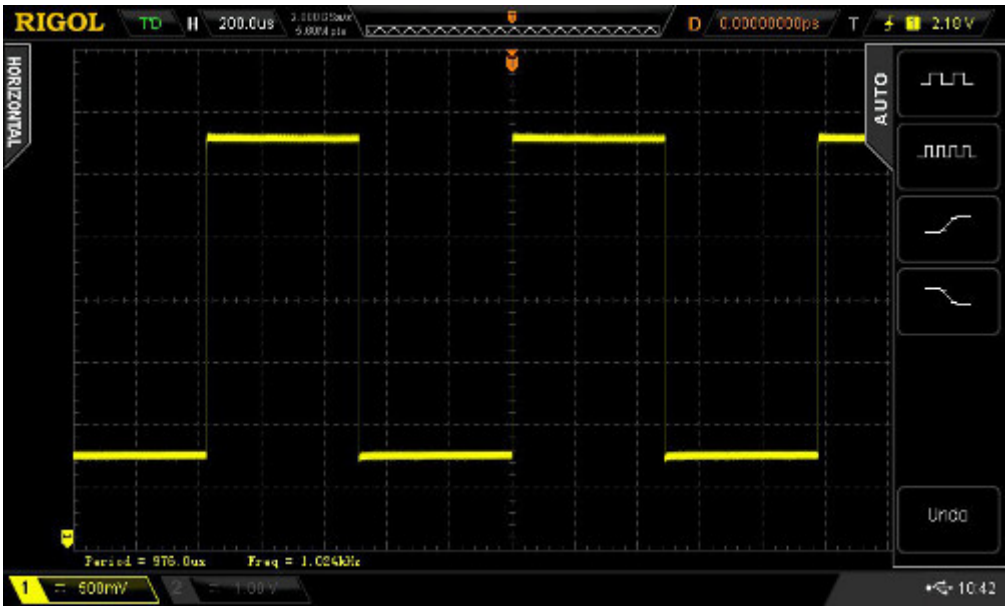
```
// Example 54.2 - PCF8563 square-wave generator (signal from pin 7)

#include "Wire.h"
#define PCF8563address 0x51

void PCF8563oscOFF()
// turns off oscillator
{
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x0D);
  Wire.write(0);
  Wire.endTransmission();
}

void PCF8563osc1Hz()
// sets oscillator to 1 Hz
{
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x0D);
  Wire.write(B10000011);
  Wire.endTransmission();
}

void PCF8563osc32Hz()
// sets oscillator to 32 kHz
{
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x0D);
  Wire.write(B10000010);
  Wire.endTransmission();
}
```

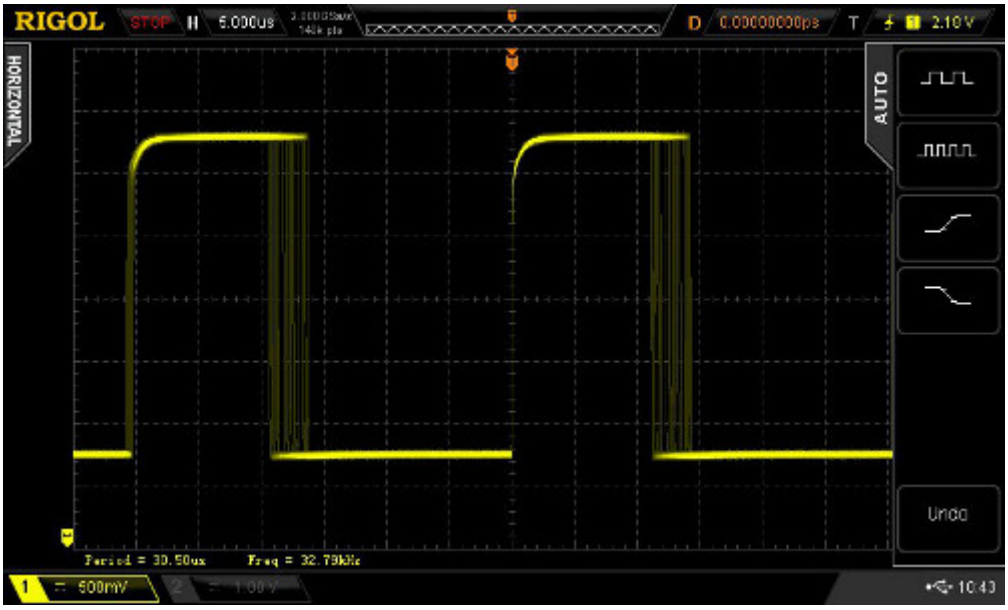
```
  Wire.write(B10000001);
  Wire.endTransmission();
}

void PCF8563osc32768kHz()
// sets oscillator to 32.768 kHz
{
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x0D);
  Wire.write(B10000000);
  Wire.endTransmission();
}

void setup()
{
  Wire.begin();
}

void loop()
{
  PCF8563osc1Hz();
  delay(2000);
  PCF8563osc32Hz();
  delay(2000);
  PCF8563osc1024kHz();
  delay(2000);
  PCF8563osc32768kHz();
  delay(2000);
  PCF8563oscOFF();
  delay(2000);
}
```

And the resulting waveforms from slowest to highest frequency. Note the sample was measured from a point between the LED and resistor, so the oscillations don't vary between the supply voltage and zero:









Self-awareness of clock accuracy

The PCF8563 monitors the oscillator and supply voltage, and if the oscillator stops or the voltage drops below a certain point – the first bit of the seconds register (called the VL bit) is set to 1.

Thus your sketch can tell you if there's a chance of the time not being accurate by reading this bit. The default value is 1 on power-up, so you need to set it back to zero after setting the time in your sketch – which is done when you write seconds using the code in our example sketches. Then from that point it can be monitored by reading the seconds register, isolating the bit and returning the value.

Examine the function *checkVLerror()* in the following example sketch. It reads the seconds byte, isolates the VL bit, then turns on D13 (the onboard LED) if there's a problem. The only way to restore the error bit to "OK" is to re-set the time:

```
// Example 54.3 - PCF8563 RTC write/read demonstration with error-checking

#include "Wire.h"
#define PCF8563address 0x51

byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
String days[] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"
};

byte bcdToDec(byte value)
{
  return ((value / 16) * 10 + value % 16);
}
```

```
void setPCF8563()
// this sets the time and date to the PCF8563
{
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x02);
  Wire.write(decToBcd(second));
  Wire.write(decToBcd(minute));
  Wire.write(decToBcd(hour));
  Wire.write(decToBcd(dayOfMonth));
  Wire.write(decToBcd(dayOfWeek));
  Wire.write(decToBcd(month));
  Wire.write(decToBcd(year));
  Wire.endTransmission();
}

void readPCF8563()
// this gets the time and date from the PCF8563
{
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x02);
  Wire.endTransmission();
  Wire.requestFrom(PCF8563address, 7);
  second     = bcdToDec(Wire.read() & B01111111); // remove VL error bit
  minute     = bcdToDec(Wire.read() & B01111111); // remove unwanted bits from MSB
  hour       = bcdToDec(Wire.read() & B00111111);
  dayOfMonth = bcdToDec(Wire.read() & B00111111);
  dayOfWeek  = bcdToDec(Wire.read() & B00000111);
  month      = bcdToDec(Wire.read() & B00011111);  // remove century bit, 1999 is over
  year       = bcdToDec(Wire.read());
}

void checkVLerror()
// this checks the VL bit in the seconds register
// and turns on D13 if there's a possible accuracy error
{
  byte test;
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x02);
  Wire.endTransmission();
  Wire.requestFrom(PCF8563address, 1);
  test = Wire.read();
  test = test & B10000000;
  if (test == B10000000)
  {
    // error
    digitalWrite(13, HIGH);
```

```
    digitalWrite(13, LOW);
  }
}

void setup()
{
  Wire.begin();
  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);
  Serial.begin(9600);
  // change the following to set your inital time
  second = 0;
  minute = 42;
  hour = 11;
  dayOfWeek = 2;
  dayOfMonth = 13;
  month = 8;
  year = 13;
  // comment out the next line and upload again to set and keep the time from resetting every
reset
  // setPCF8563();
}

void loop()
{
  readPCF8563();
  Serial.print(days[dayOfWeek]);
  Serial.print(" ");
  Serial.print(dayOfMonth, DEC);
  Serial.print("/");
  Serial.print(month, DEC);
  Serial.print("/20");
  Serial.print(year, DEC);
  Serial.print(" - ");
  Serial.print(hour, DEC);
  Serial.print(":");
  if (minute < 10)
  {
    Serial.print("0");
  }
  Serial.print(minute, DEC);
  Serial.print(":");
  if (second < 10)
  {
    Serial.print("0");
```

And now for a demonstration of the error-checking at work. We have the PCF8563 happily returning the data to the serial monitor. Then the power is removed and restored. You see D13 on the Arduino-compatible board turn on and then the error is displayed in the serial monitor:

This function may sound frivolous, however if you're building a real product or serious project using the PCF8563, you can use this feature to add a level of professionalism and instil confidence in the end user.

Alarm Clock

You can use the PCF8563 as an alarm clock, that is be notified of a certain time, day and/or day of the week – at which point an action can take place. For example, trigger an interrupt or turn on a digital output pin for an external siren. Etcetera. Using the alarm in the sketch is quite similar to reading and writing the time, the data is stored in certain registers – as shown in the following table from page seven of the data sheet:

Alarm registers							
09h	Minute_alarm	AE_M	MINUTE_ALARM (0 to 59)				
0Ah	Hour_alarm	AE_H	x	HOUR_ALARM (0 to 23)			
0Bh	Day_alarm	AE_D	x	DAY_ALARM (1 to 31)			
0Ch	Weekday_alarm	AE_W	x	x	x	x	WEEKDAY_ALARM (0 to 6)

However there is a catch – the MSB (most significant bit, 7) in the registers above is used to determine whether that particular register plays a part in the alarm. For example, if you want your alarm to include hours and minutes, bit 7 needs to be set to 1 for the hour and minute alarm register. Don't panic – you can easily set that bit by using a bitwise OR ("|") and B10000000 to set the bit on with the matching data before writing it to the register.

alarm off by setting that bit to zero. Using hardware, first set bit 1 of register 0x01 to 1 – then whenever an alarm occurs, current can flow into pin 3 of the PCF8563.

Yes – it's an open-drain output – which means current flows from the supply voltage into pin 3. For example if you want to turn on an LED, connect a 560Ω resistor between 5V and the anode of the LED, then connect the cathode to pin 3 of the PCF8563. To turn off this current, you need to turn off the alarm flag bit as mentioned earlier.

Now let's put all that into a demonstration sketch. It's documented and if you've been following along it shouldn't be difficult at all:

```
// Example 54.4 - PCF8563 alarm clock demonstration

#include "Wire.h"
#define PCF8563address 0x51

byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
byte alarmMinute, alarmHour, alarmDay, alarmDayOfWeek;
String days[] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"
};

byte bcdToDec(byte value)
{
  return ((value / 16) * 10 + value % 16);
}

byte decToBcd(byte value){
  return (value / 10 * 16 + value % 10);
}

void setPCF8563alarm()
// this sets the alarm data to the PCF8563
{
  byte am, ah, ad, adow;
  am = decToBcd(alarmMinute);
  am = am | 100000000; // set minute enable bit to on
  ah = decToBcd(alarmHour);
  ah = ah | 100000000; // set hour enable bit to on
  ad = decToBcd(alarmDay);
  ad = ad | 100000000; // set day of week alarm enable bit on
  adow = decToBcd(alarmDayOfWeek);
  adow = ad | 100000000; // set day of week alarm enable bit on
```

```
  // optional day of month and day of week (0~6 Sunday - Saturday)
  /*
  Wire.write(ad);
  Wire.write(adow);
  */
  Wire.endTransmission();

  // optional - turns on INT_ pin when alarm activated
  // will turn off once you run void PCF8563alarmOff()
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x01);
  Wire.write(B00000010);
  Wire.endTransmission();
}

void PCF8563alarmOff()
// turns off alarm enable bits and wipes alarm registers.
{
  byte test;
  // first retrieve the value of control register 2
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x01);
  Wire.endTransmission();
  Wire.requestFrom(PCF8563address, 1);
  test = Wire.read();

  // set bit 3 "alarm flag" to 0
  test = test - B00001000;

  // now write new control register 2
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x01);
  Wire.write(test);
  Wire.endTransmission();
}

void checkPCF8563alarm()
// checks if the alarm has been activated
{
  byte test;
  // get the contents from control register #2 and place in byte test;
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x01);
  Wire.endTransmission();
  Wire.requestFrom(PCF8563address, 1);
```

```
    // alarm! Do something to tell the user
    Serial.println("** alarm **");
    delay(2000);

    // turn off the alarm
    PCF8563alarmOff();
  }
}

void setPCF8563()
// this sets the time and date to the PCF8563
{
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x02);
  Wire.write(decToBcd(second));
  Wire.write(decToBcd(minute));
  Wire.write(decToBcd(hour));
  Wire.write(decToBcd(dayOfMonth));
  Wire.write(decToBcd(dayOfWeek));
  Wire.write(decToBcd(month));
  Wire.write(decToBcd(year));
  Wire.endTransmission();
}

void readPCF8563()
// this gets the time and date from the PCF8563
{
  Wire.beginTransmission(PCF8563address);
  Wire.write(0x02);
  Wire.endTransmission();
  Wire.requestFrom(PCF8563address, 7);
  second     = bcdToDec(Wire.read() & B01111111); // remove VL error bit
  minute     = bcdToDec(Wire.read() & B01111111); // remove unwanted bits from MSB
  hour       = bcdToDec(Wire.read() & B00111111);
  dayOfMonth = bcdToDec(Wire.read() & B00111111);
  dayOfWeek  = bcdToDec(Wire.read() & B00000111);
  month      = bcdToDec(Wire.read() & B00011111);  // remove century bit, 1999 is over
  year       = bcdToDec(Wire.read());
}

void setup()
{
  Wire.begin();
  Serial.begin(9600);
  // change the following to set your initial time
  second = 50;
```

```
  month = 8;
  year = 13;
  // comment out the next line and upload again to set and keep the time from resetting every
reset
  setPCF8563();

  alarmMinute = 45;
  alarmHour = 13;
  // comment out the next line and upload again to set and keep the alarm from resetting ev-
ery reset
  setPCF8563alarm();
}

void loop()
{
  readPCF8563();
  Serial.print(days[dayOfWeek]);
  Serial.print(" ");
  Serial.print(dayOfMonth, DEC);
  Serial.print("/");
  Serial.print(month, DEC);
  Serial.print("/20");
  Serial.print(year, DEC);
  Serial.print(" - ");
  Serial.print(hour, DEC);
  Serial.print(":");
  if (minute < 10)
  {
    Serial.print("0");
  }
  Serial.print(minute, DEC);
  Serial.print(":");
  if (second < 10)
  {
    Serial.print("0");
  }
  Serial.println(second, DEC);
  delay(1000);

  // alarm?
  checkPCF8563alarm();
}
```

This is the same as the example 54.1, however we've added the required functions to use the alarm. The re-

and is written to the PCF8563 using the function:

```
void setPCF8563alarm()
```

Note the use of bitwise OR ("|") to add the enable bit 7 to the data before writing to the register. The interrupt pin is also set to activate at the end of this function, however you can remove that part of the code if unnecessary. We also demonstrate checking the alarm status via software using the function:

```
void checkPCF8563alarm()
```

which simply reads the AF bit in the register at 0x01 and let's us know if the alarm has occurred via the Serial Monitor. In this function you can add code to take action for your required needs. It also calls the function:

```
void PCF8563alarmOff()
```

which retrieves the contents of the register at 0x01, sets the AF bit to zero and writes it back. We do this to preserve the status of the other bits in that register. For the curious and non-believers you can see this sketch in action through the following video, first the software and then the hardware interrupt pin method (an LED comes on at the alarm time and is then turned off:

Conclusion

Hopefully you found this tutorial useful and now have the confidence to use the PCF8563 in your own projects. Furthermore I hope you learned something about the I2C bus and can have satisfaction in that you didn't take the lazy option of using the library.

People often say to us "Oh, there's a library for that", however if you used every library – you'd never learn how to interface things for yourself. One day there might not be a library! And then where would you be? So learning the hard way is better for you in the long run.

This post is brought to you by pmdway.com – everything for makers and electronics enthusiasts, with free delivery worldwide.

To keep up to date with new posts at tronixstuff.com, please subscribe to the mailing list in the box on the right, or follow us on twitter @tronixstuff.

This entry was posted in arduino, I2C, PCF8563, real time clock, tronixstuff, tutorial and tagged arduino, clock, IC, NXP, PCF8563, PMDway, real, RTC, time, tronixstuff, tutorial on August 13, 2013 [https://tronixstuff.com/2013/08/13/tutorial-arduino-and-pcf8563-real-time-clock-ic/] by John Boxall.

About John Boxall

https://nostarch.com/arduino-workshop-2nd-edition

View all posts by John Boxall →
