ME 4370 - Mechatronics - Spring 2016

It is all about it Timing - PCs vs. MCUs

Overview :

A microcontroller, or MCU is fundamentally different from a personal computer. MCUs are capable of operating on *precise* timing rountines while personal computers run on a *non-deterministic* timing. Therefore, one specialized capability of an MCU is precise timing of input and output as well as internal events. Personal computers cannot do this.

Measure Time :

How could we make a timer? Lets talk about some ideas.

1. moon and stars (sun)
2. seasons
3. sand, hourglass
4. pendulum, time based on ω_n powered by weight or springs
5. quartz or other crystall based timing
6. whats another oscillator you can think of?

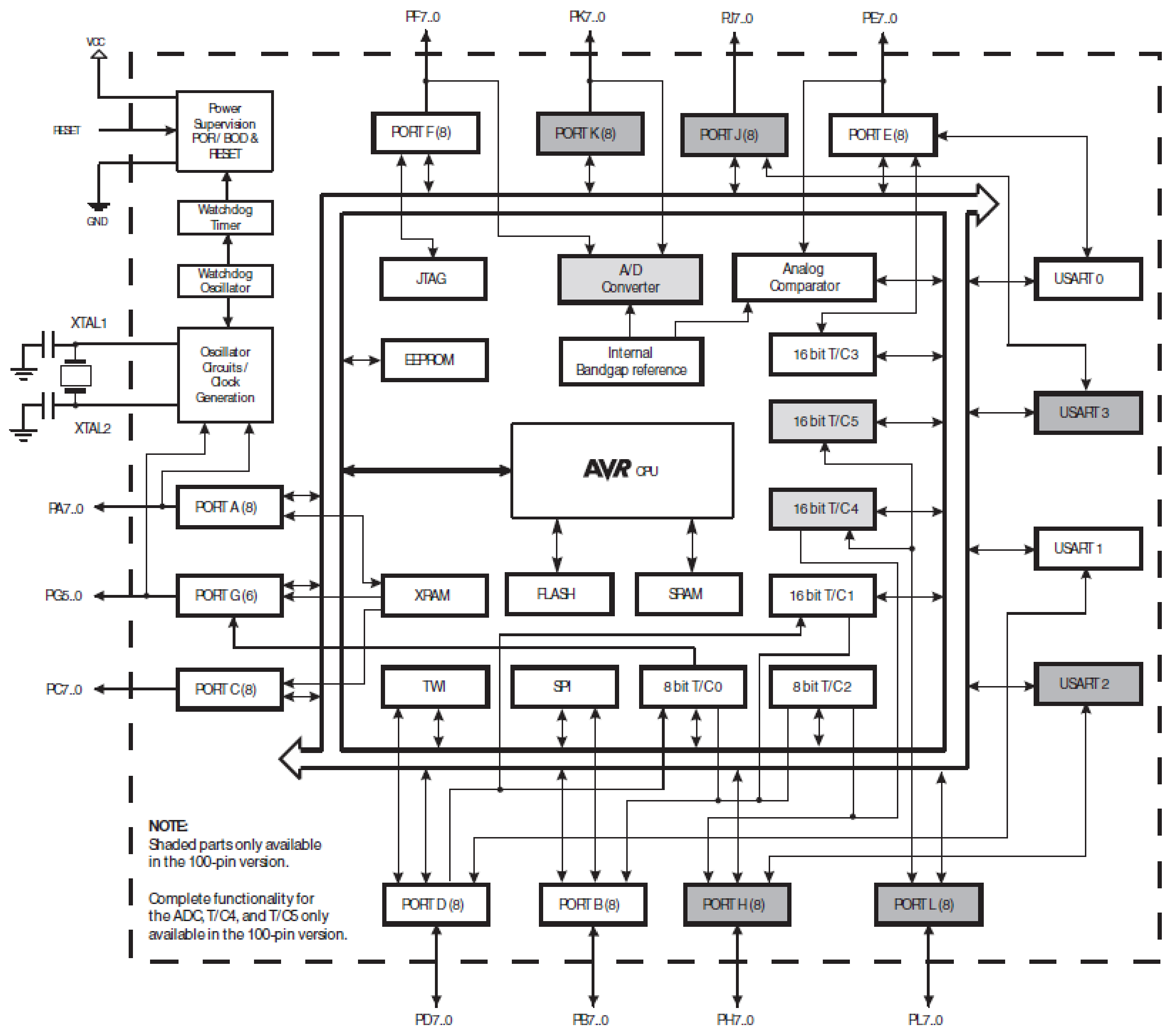
The System Clock :

Most MCUs contain an internal oscillator based on a quartz crystal (wikipedia). This oscillator is used as the *system clock* which schedules program execution and allows for advanced timing capabilties. The speed of this system is often measured in MIPS (Million Instructions Per Second) not to be confused with (Microprocessor without Interlocked Pipeline Stages).

We generally do not interface the system clock directly, however there is a specialized register that we can access called the timer counter TCNT. Our MCU (ATMEGA2560) has several of these available. If the timer counter is enabled it acts as a *free running timer*. It is really just a register that increments itself in time with the system clock.

Consider the 8bit timer counter TCNT0 on your 16MHz ATMEGA2560. What does this mean? This means that on every cycle of the clock TCNT0 = TCNT0 + 1. However we do not see this in code. This happens very quickly! So to use it to measure events that that more than a very short time we have to manage the *timer overflow*. It turns out there is a register for that.

Figure 2-1. Block Diagram



Timing on a PC :

Why does it take so long for a to PC application to load? PCs also operate on a very similar system clock. The main difference is the fact an OS level computer is doing many things at once and the multiple routines and programs must share the system resources. So scheduling these events in true real time is a very difficult task but it can be done (RTOS). Alternatively, most OS have *non-deterministic* timing routines.

Precise Timing Capabilities :

1. Scheduling Internal Events
2. Timing Internal Events
3. Scheduling External Events - Outputs
4. Timing External Events - Inputs

Scheduling Internal Events :

Scheduling internal events using the TCNT register is generally done through *polling* or through the use of *interrupt routines*.

Polling is act of accessing the TCNT register, and this is generally in the main code, *loop*. The example shows a while loop used to wait until the TCNT register reaches a certain value. This may be the easiest way the do internal timing but it is not the best.

Polling example 2

```
void loop() {

  c_time=TCNT1L;

  while (!(TCNT0<200);
  c_cnt++;

  Serial.println(c_cnt);


}
```

Polling example 2

```
void loop() {

  c_time=TCNT1L;

  while (!(TIFR1&B00000001));
  c_cnt++;

  Serial.println(c_cnt);


}
```

Interrupt Service Routines encompass are a very important capability of the MCU. An ISR or referred to as an *Interrupt* is a special block of code, i.e. a function, that executes exactly when we need it to. This can be based on different things, but in this example the function *fires* whe TCNT reaches a certain value. This is commonly referred to as a *Real Time Interrupt.*
ISR example

```
void setup()
{
pinMode(13,OUTPUT);
/* or use:
DDRB = DDRB | B00100000;  // this sets pin 5  as output
                       // without changing the value of the other pins
*/
// Disable interrupts while loading registers
cli();
// Set the registers
TCCR1A = 0; //Timer Counter Control register
// Set mode
TCCR1B = (1 << WGM12); // turn on CTC mode
// Set prescale values (1024). (Could be done in same statement
// as setting the WGM12 bit.)
TCCR1B |= (1 << CS12) | (1 << CS10);
//Enable timer compare interrupt===> TIMSK1 for ATmega328,
//TIMSK for ATmega8
TIMSK |= (1 << OCIE1A);
// Set OCR1A
OCR1A = 15624;
// Enable global interrupts
sei();
}
void loop(){}

ISR (TIMER1_COMPA_vect) {
   digitalWrite(13, !digitalRead(13));
   //PORTB ^= _BV(PB5); // as digitalWrite(13,x) is an Arduino
   //function, direct writing to the port may be preferable
}
```