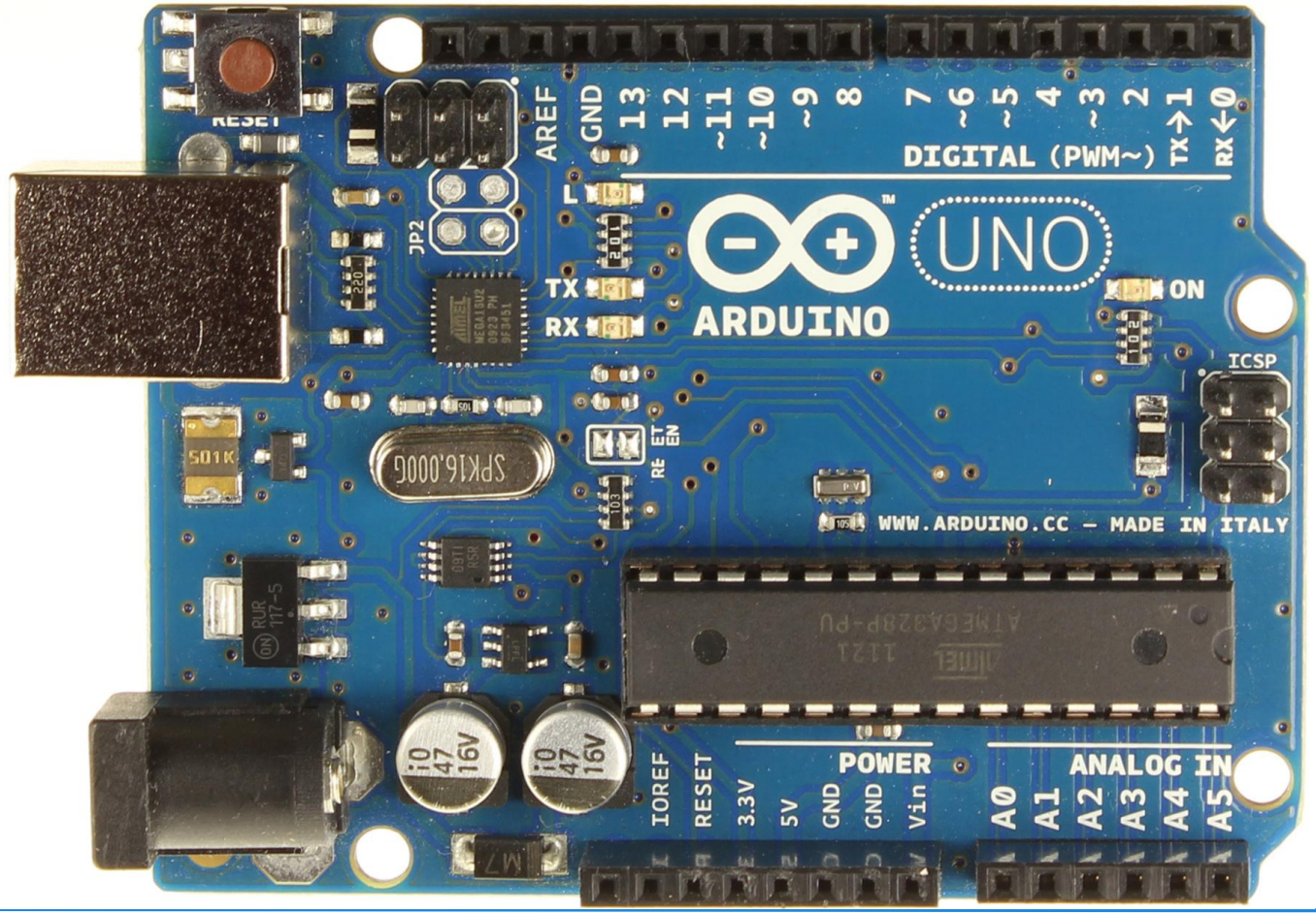
Curso avanzado sobre Arduino: Interrupciones Software

elCacharreo.com

Arduino Intermedio: Presente



Arduino Intermedio: Presente

José Antonio Vacas Martínez

blog

javacasm@elcacharreo.com

twitter
linkedin

Arduino Intermedio: Interrupciones

Utilidad

Cuidados (algo ~~puede~~ dejará de funcionar)

Teoría: timers

Ejemplo http://real2electronics.blogspot.com.es/2011/01/timer-2.html

Tiempo, alarmas http://playground.arduino.cc/Code/time

RTos: http://arduino.cc/forum/index.php?topic=138643.0

Interrupciones

Una interrupción es una parada en la ejecución del programa principal. Un salto a otra función (instrucción en general) y la vuelta al punto por el que iba el programa.

Esto nos permite trabajar en entornos de ejecución más complejos y parecidos a un entorno multitarea.

Para poder trabajar con interrupciones el microprocesador o microcontrolador tiene que tener esa capacidad de parar lo que está haciendo, guardar el estado, hacer otra cosa y volver al punto por el que iba.

Las interrupciones se pueden activar y exite una máscara que nos dicen qué tipo está activo y cual no

interrupts() / noInterrupts(). Activan/Desactivan (internamente TIMSKx).

When an interrupt occurs, a flag in the interrupt flag register (TIFRx) is been set. This interrupt will be automatically cleared when entering the ISR or by manually clearing the bit in the interrupt flag register.

Interrupciones por tiempo

Un timer puede generar distintos tipos de interrupciones.
Los registros tiene un sufijo según el timer (TIMSK1 timer1 interrupt mask register, con x entre 0 y 5) y
Las outputs (A,B,C), por ejemplo OCR2A (timer2 output compare register A).

Timer Overflow:
Timer overflow means the timer has reached is limit value. When a timer overflow interrupt occurs, the timer overflow bit TOVx will be set in the interrupt flag register TIFRx. When the timer overflow interrupt enable bit TOIEx in the interrupt mask register TIMSKx is set, the timer overflow interrupt service routine ISR(TIMERx_OVF_vect) will be called.

Timers

¿Qué es un timer? : es un hardware independiente al microprocesador que puede ejecutar tareas paralelas sin detener el flujo del programa.

El microcontrolador de las placas Arduino dispone de 3 módulos Contador/Timers (dos de 8 bits y uno de 16 bits).

Los Timers son usados internamente por el core de Arduino para sus funciones internas, como pueden ser millis, delay, analogWrite,..

En el Mega hay timer3, timer4 y timer5 todos de 16bit timers

Todos los timers dependen del system clock de Arduino. Normalmente es 16MHz, salvo Arduino Pro 3,3V con 8Mhz.

Existen registros que modifican el escalado de los timers con el relj: se conoce como el preescaler. En Arduino todos están configurados a 1kHz con las interrupciones activadas.

Timers arduino

Timer0:

Timer0 es un timer de 8bit.

En Arduino se usa para delay(), millis() y micros().

Timer1:

Timer1 es un timer de 16bit.
En Arduino se usa para Servo library en Arduino Uno (timer5 en Arduino Mega).

Timer2:

Timer2 is a 8bit timer like timer0.
Se usa para tone()

Timer Register

Se puede cambiar su comportamiento usando los registros adecuados

TCCRx - Timer/Counter Control Register. The prescaler can be configured here.

http://letsmakerobots.com/node/28278 http://real2electronics.blogspot.com.es/2011/01/timer-2.html

Timers arduino

Muchos registros son utilizados para controlar cada timer. el Controlador de Registros del Timer/Contador TCCRnA yTCCRnB mantiene el control de los bits principales del timer. (Notese que TCCRnA y TCRnB no corresponden a las salidas A y B). Estos registros mantienen varios grupos de bits:

- Waveform Generation Mode bits (WGM): Estos controlan el timer de modo general.
- (Estos bits se dividen entre TCCRnA y TCCRnB.)
- Clock Select bits (CS): Controla el reloj predivisor
- Compare Match Output A Mode bits (COMnA): Habilita/deshabilita/invierte la salida A
- Compare Match Output B Mode bits (COMnB): Habilita/deshabilita/invierte la salida B

Los Registros de Salida Comparada OCRnA y OCRnB ajustan los niveles en los que las salidas A y B serán afectadas. Cuando el valor del timer coincida con el valor del registro, la correspondiente salida será modificada como se especifique por el modo.

http://arduino.cc/es/Tutorial/SecretsOfArduinoPWM

Librería Timer1

http://playground.arduino.cc/code/timer1

La velocidad del Timer1's la define el prescaler, o divisor. Este se puede establecer en 1, 8, 64, 256 or 1024.

- Max Period = (Prescale)*(1/Frequency)*(2^17)
- Time per Tick = (Prescale)*(1/Frequency)

void initialize(long microseconds=1000000); //**breaks analogWrite() for digital pins 9 and 10**

void start();

void stop();

void restart();

unsigned long read();

void setPeriod(long microseconds);

void pwm(char pin, int duty, long microseconds=-1); pin: 9 o 10, duty: 10 bit

void setPwmDuty(char pin, int duty);

void disablePwm(char pin);

void attachInterrupt(void (*isr)(), long microseconds=-1);

void detachInterrupt();

Librería Timer1 http://playground.arduino.cc/code/timer1

```
#include <TimerOne.h>

void setup() {
   pinMode(13, OUTPUT);
  Timer1.initialize(100000);
// set a timer of length 100000 microseconds (or 0.1 sec - or 10Hz => the led will blink 5 times, 5 cycles of on-and-off, per second)
  Timer1.attachInterrupt( timerIsr ); // attach the service routine here}

void loop()
{  // Main code loop
  // TODO: Put your regular (non-ISR) logic here
}

void timerIsr()
{
    digitalWrite( 13, digitalRead( 13 ) ^ 1 );     // Toggle LED
 }
```

Librería Timer1 http://playground.arduino.cc/code/timer1

```
#include "TimerOne.h"

void setup()
{
  pinMode(10, OUTPUT);
  Timer1.initialize(500000);         // initialize timer1, and set a 1/2 second period
  Timer1.pwm(9, 512);                // setup pwm on pin 9, 50% duty cycle
  Timer1.attachInterrupt(callback);  // attaches callback() as a timer overflow interrupt
}

void callback()
{
  digitalWrite(10, digitalRead(10) ^ 1);
}

void loop()
{}
```

Librería timer2

http://playground.arduino.cc//Code/FrequencyTimer2

FrequencyTimer2 utiliza el timer2 que normalmente controla el PWMs en **pin 11 and 3**

http://www.pjrc.com/teensy/td_libs_FrequencyTimer2.html

Librería timer2

http://playground.arduino.cc//Code/FrequencyTimer2

```
#include <FrequencyTimer2.h>
void setup() {
  pinMode(11,OUTPUT);
  Serial.begin(19200);
  delay(2);
  Serial.print("Ready");
  FrequencyTimer2::setPeriod(200);
  FrequencyTimer2::enable();}

long burpCount = 0;
extern "C" void Burp(void) {
  burpCount++; }

void loop() {
  static unsigned long v = 0;
  if ( Serial.available()) {
    char ch = Serial.read();
    switch(ch) {
      case '0'...'9': v = v * 10 + ch - '0'; break;
      case 'p':       FrequencyTimer2::setPeriod(v);
       Serial.print("set ");       Serial.print((long)v, DEC);       Serial.print(" = ");
       Serial.print((long)FrequencyTimer2::getPeriod(), DEC); Serial.println();  v = 0;
break;
      case 'e':       FrequencyTimer2::enable();       break;
      case 'd':       FrequencyTimer2::disable();       break;
      case 'o':       FrequencyTimer2::setOnOverflow( Burp);       break;
      case 'n':       FrequencyTimer2::setOnOverflow(0);       break;
      case 'b':       Serial.println(burpCount,DEC);       break;
      case 'x':
#if defined(__AVR_ATmega168__)
       Serial.print("TCCR2A:");Serial.println(TCCR2A,HEX);       Serial.print("TCCR2B:");
      Serial.println(TCCR2B,HEX);Serial.print("OCR2A:");       Serial.println(OCR2A,HEX);
#else
       Serial.print("TCCR2:");Serial.println(TCCR2,HEX);
       Serial.print("OCR2:");Serial.println(OCR2,HEX);
#endif
       Serial.print("TCNT2:");
       Serial.println(TCNT2,HEX);
       break;
    }
  }
}
```

Arduino Intermedio: Interrupciones

Ejemplo de semáforos hecho de esta otra forma

Secretos del PWM

http://arduino.cc/es/Tutorial/SecretsOfArduinoPWM

```
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delayMicroseconds(100); // Aproximadamente 10% de ciclo de trabajo a 1KHz
  digitalWrite(13, LOW);
  delayMicroseconds(1000 - 100); // Aproximadamente 90% de ciclo de trabajo restante apagado
}
```

Secretos del PWM

http://arduino.cc/es/Tutorial/SecretsOfArduinoPWM

El siguiente fragmento de código configura el PWM rápido en los pines 3 y 11 (Timer 2). Para resumir el registro de configuración, poniendo los bits del Modo de generación de Ondas (WGM - waveform generation mode) a 011 selecciona PWM rápido. Poniendo los bits COM2A y COM2B a 10 asigna PWM no-invertido (non-inverted PWM) para las salidas A y B. Poniendo el CS a 100 ajusta el predivisor a dividir el reloj por 64. (Como los bits son diferentes para los diferentes timers, consulta la ficha para ver los valores correctos). Los registros comparados de salida son arbritariamente ajustados a 180 y 50 para controlar el ciclo de trabajo del PWM de las salidas A y B. (Por supuesto puedes modificar directamente los registros en vez de utilizar pinMode, pero necesitas configurar los pins a OUTPUT).

```
pinMode(3, OUTPUT);
 pinMode(11, OUTPUT);
 TCCR2A = _BV(COM2A1) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);
 TCCR2B = _BV(CS22);
 OCR2A = 180;
 OCR2B = 50;
```

Enlaces

Para saber mas:

http://real2electronics.blogspot.com.es/2011/01/timer-2.html

http://letsmakerobots.com/node/28278

Conclusiones

Gracias por vuestra atención