Interrupts

An interrupt is a signal that can interrupt the normal execution flow to take care of an immediate event. Arduino Leonardo allows for as many as 5 interrupt pins. Interrupts can be triggered on 4 different situations:

* When the pin is LOW
* Whenever the pin changes value
* When the pin goes from LOW to HIGH
* When the pin goes from HIGH to LOW

Additionally, timed interrupts can be set. Arduino Leonardo

Pin 2

sei();                    // Enable global interrupts  
EIMSK |= (1 << 1);     // Enable external interrupt INT1

EICRA |= (1 << 2);    // Trigger INT1 on change

EICRA |= (1<<ISC31) | (1<<ISC21) | ((1<<ISC11)|(1<<ISC10)) | (1<<ISC00)

ISR(EXT\_INT1\_vect)

ICR1 = 15625 /timer limit

TCCR1B |= (1<<CS12) | (1<<WGM13) //100 -> 256 scaling and wgm13-> ICR1 is limit

TIMSK1 |= 1 timer overflow

ISR(TIMER1\_OVF\_vect)

Pin 11 input no pullup

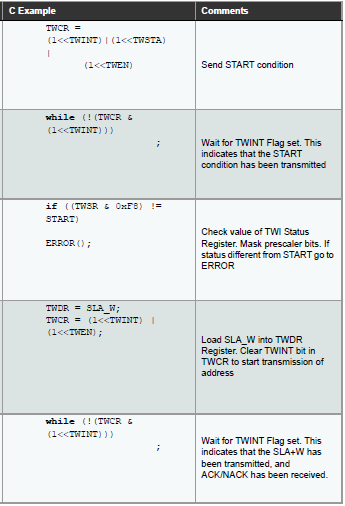
Value = PINB & (1<<PINB7) // read value

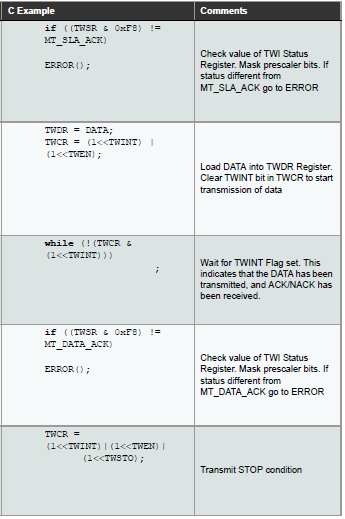
Pin 2 pullup

PORTD |= (1<<1) //set pullup

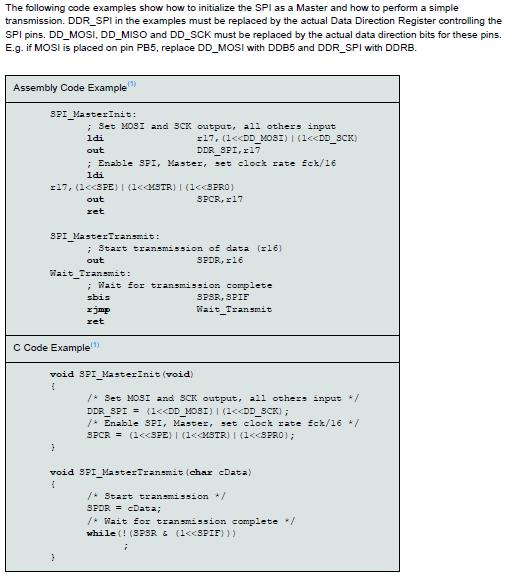
Value = PIND & (1<<1)

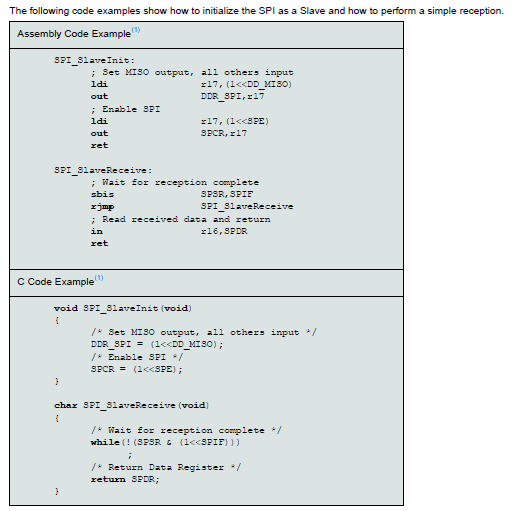
I2c





Spi





PWM

void NewTone(uint8\_t pin, unsigned long frequency, unsigned long length) {

uint8\_t prescaler = \_BV(CS10); // Try using prescaler 1 first.

unsigned long top = F\_CPU / frequency / 4 - 1; // Calculate the top.

if (top > 65535) { // If not in the range for prescaler 1, use prescaler 256 (61 Hz and lower @ 16 MHz).

prescaler = \_BV(CS12); // Set the 256 prescaler bit.

top = top / 256 - 1; // Calculate the top using prescaler 256.

}

if (length > 0) \_nt\_time = millis() + length; else \_nt\_time = 0xFFFFFFFF; // Set when the note should end, or play "forever".

if (\_pinMask == 0) {

\_pinMask = digitalPinToBitMask(pin); // Get the port register bitmask for the pin.

\_pinOutput = portOutputRegister(digitalPinToPort(pin)); // Get the output port register for the pin.

uint8\_t \*\_pinMode = (uint8\_t \*) portModeRegister(digitalPinToPort(pin)); // Get the port mode register for the pin.

\*\_pinMode |= \_pinMask; // Set the pin to Output mode.

}

ICR1 = top; // Set the top.

if (TCNT1 > top) TCNT1 = top; // Counter over the top, put within range.

TCCR1B = \_BV(WGM13) | prescaler; // Set PWM, phase and frequency corrected (ICR1) and prescaler.

TCCR1A = \_BV(COM1B0);

TIMSK1 |= \_BV(OCIE1A); // Activate the timer interrupt.

}

Analog read

int ADCsingleREAD(uint8\_t adctouse)

{

    int ADCval;

    ADMUX |= (1<<MUX2);        // ADC4

    ADMUX |= (1 << REFS0);   // use AVcc as the reference

// 128 prescale for 8Mhz

ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);

    ADCSRA |= (1 << ADEN);    // Enable the ADC

    ADCSRA |= (1 << ADSC);    // Start the ADC conversion

    while(ADCSRA & (1 << ADSC)); // waits for the ADC to finish

    ADCval = ADCL;

    ADCval = (ADCH << 8) + ADCval; // ADCH is read so ADC can be updated again

    return ADCval;

}

Tone

<https://github.com/arduino/ArduinoCore-avr/blob/master/cores/arduino/Tone.cpp>