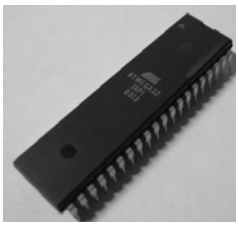
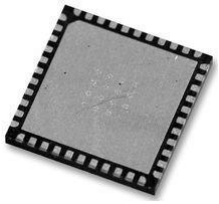
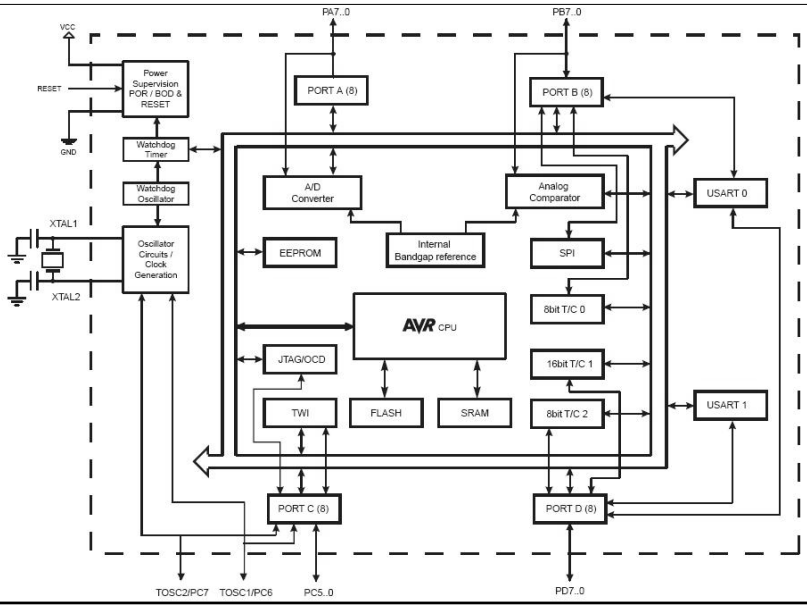


Atmel AVR

Subrat Nayak
EEL 4924



Microcontroller Vs Microprocessor

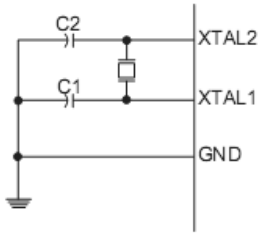


Choosing an Atmel AVR

- http://www.atmel.com/AVR
- Footprint
  - DIP
  - SOIC/ SSOP/TQFP
  - MLF/BGA

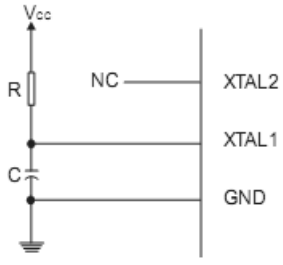
CLOCK

- External Crystal/Ceramic Resonator



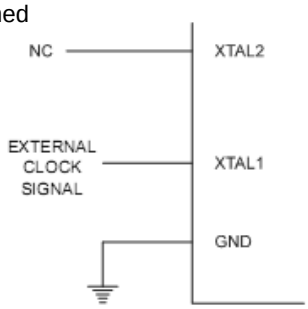
- External Low-frequency Crystal -
  32.768 kHz watch crystal

- External RC Oscillator –
  - CKOPT fuse –
    sets internal 36pF bw xtal1 and Gnd
  - f = 1/(3RC)



CLOCK

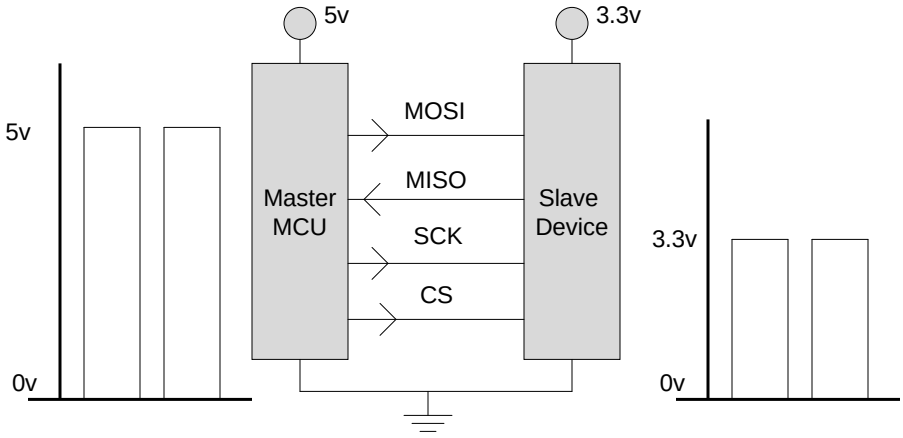
- Calibrated Internal RC Oscillator
  - Fixed 1.0, 2.0, 4.0, or 8.0 MHz clock
  - CKOPT Fuse should always be un-programmed
- External Clock - EASY
  - CKOPT fuse –
  - sets internal 36pF bw xtal1 and Gnd
  - Be careful about voltage of Oscillator – 3.3v / 5v



- Real Time Clock
  - Timer/Counter Oscillator pins (TOSC1 and TOSC2)
  - Optimized for use with a 32.768 kHz watch crystal

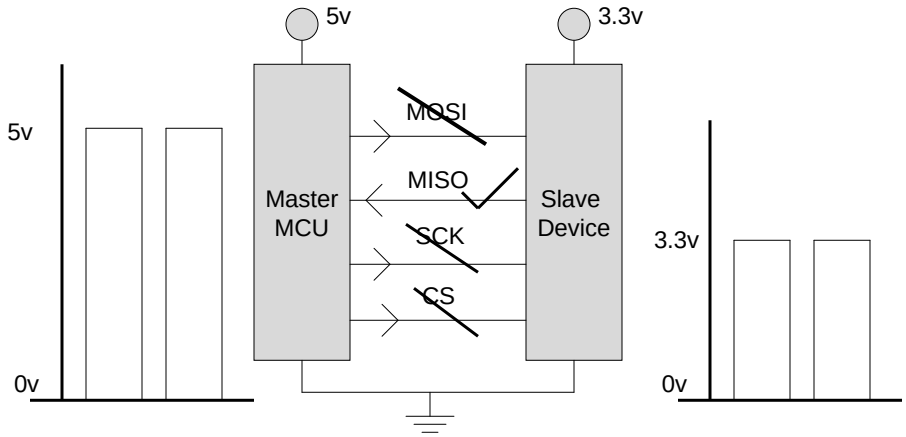
I/O

- Electrical Characteristics
  - Current Sink/Source
  - Max voltage output or Input



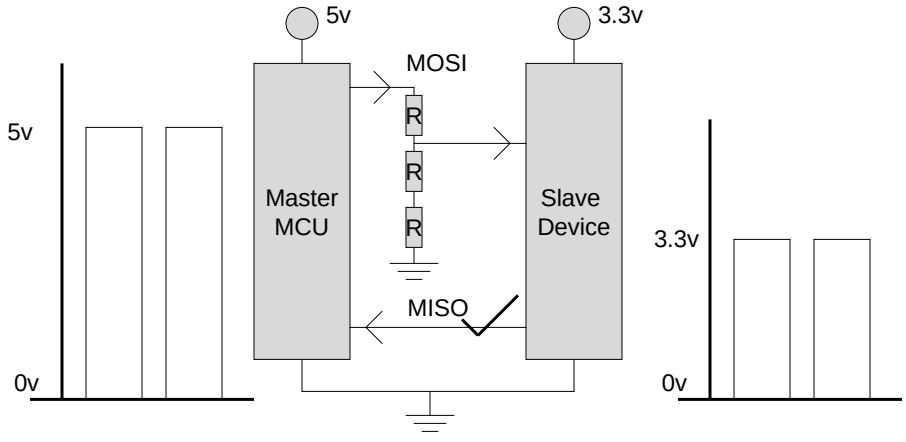
I/O

- Electrical Characteristics
  - Current Sink/Source
  - Max voltage output or Input



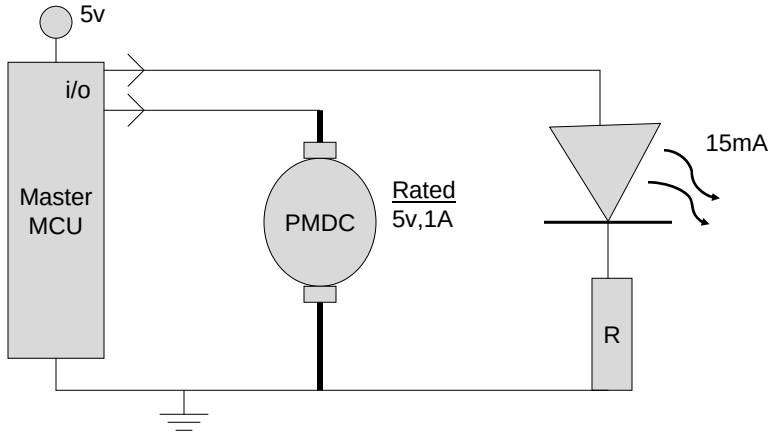
I/O

- Electrical Characteristics
  - Current Sink/Source
  - Max voltage output or Input



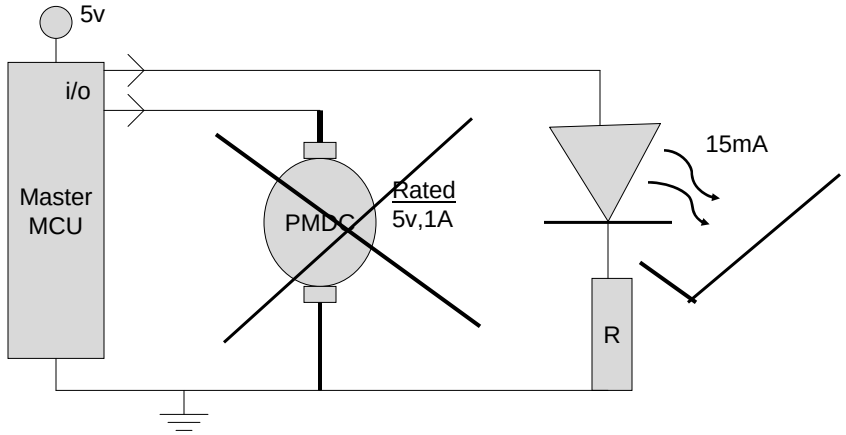
I/O

- Electrical Characteristics
  - Current Sink/Source
  - Max voltage output or Input



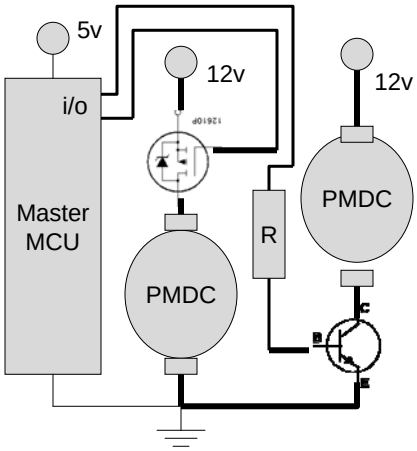
I/O

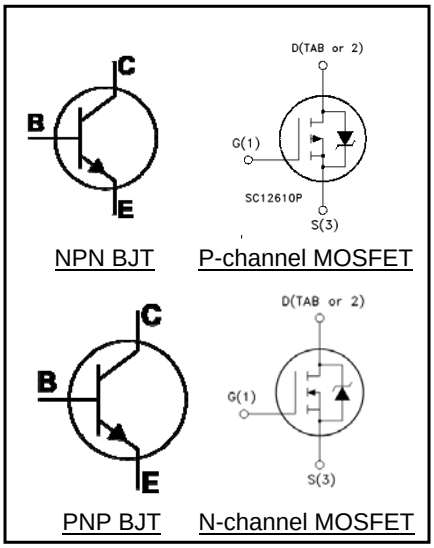
- Electrical Characteristics
  - Current Sink/Source
  - Max voltage output or Input



I/O

- Electrical Characteristics
  - Current Sink/Source
  - Max voltage output or Input

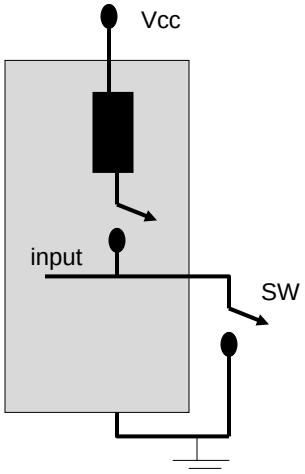




I/O

- OUTPUT PIN
  - DDRX – bit needs to be set (1)
  - PORTX – 1 for HIGH and 0 for LOW

- INPUT PIN
  - DDRX – bit needs to be cleared (0)
  - PORTX – 1 enables pull up Resistor (can source current)
    - 0 disables pull up Resistor (Hi-Z)
  - PINX – Read the logic level of voltage on the pin applied externally
    - 1 - pin has been turned HIGH externally
    - 0 – pin has been turned LOW externally



I/O

➢set one or more bits in a Register

```
#define LCD_DATA_DDR            DDRC
#define LCD_DATA_PORT           PORTC
#define LCD_DATA_PIN            PINC
#define LCD_Data_Pin7           7
#define LCD_Data_Pin6           6
#define LCD_Data_Pin5           5
#define LCD_Data_Pin4           4

Inside function use this →
LCD_DATA_DDR  |=
(_BV(LCD_Data_Pin7)|_BV(LCD_Data_Pin6)|_BV(LCD_Data_Pin5)|_BV(LCD_Data_Pin4));
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

DDRC = 0b11110000;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

DDRC = 0xF0;
```

I/O

➢clear one or more bits in a Register

```
#define ADC_DATA_DDR            DDRF
#define ADC_DATA_PORT           PORTF
#define ADC_Data_Pin0           0
#define ADC_Data_Pin1           1

Inside function use this →
ADC_DATA_DDR &= ~(_BV(ADC_Data_Pin0)|_BV(ADC_Data_Pin1));
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

DDRF = 0b00111111;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

DDRC = 0x3F;
```

I/O

- check a bit in a Register

```
#define LCD_DATA_DDR            DDRC
#define LCD_DATA_PORT           PORTC
#define LCD_DATA_PIN            PINC
#define LCD_Data_Pin7            7

Inside function use this →
while((LCD_DATA_PIN & _BV(LCD_Data_Pin7)) >> (LCD_Data_Pin7))
{

}

OR use

if ((LCD_DATA_PIN & _BV(LCD_Data_Pin7)) >> (LCD_Data_Pin7))
{

}
```

ADC

- 10 bit resolution
- Maximum value represents the voltage on the AREF pin
- Aref = External voltage or AVCC or 2.56v
- Ex – If 0 to 5v == 0 to 1023
  - Smallest measurable voltage = 4.88mV
- 8 Multiplexed Single Ended Input Channels
- 7 Differential Input Channels
- Free Running or Single Conversion Mode
- ADC Conversion Complete Indicator for Single Conversion Mode
- ADC Prescaler sets the Sampling Rate(50kHz - 200kHz)
- First AD conversion is generally erratic

ADC

ADCSRA:

ADEN	Enables ADC
ADSC	Starts conversion(s)
ADIF	Conversion complete (an interrupt flag)
ADPS2:0	Divides the clock

ADMUX: A big multiplexer. Tells you which pin you want to do a conversion on. Also lets you select voltage reference.

```
analogLow = ADCL; // read ACD low register
analogHigh = ADCH; // read ACD high register
analog8bit = ((analogHigh << 6) | (analogLow >> 2));
```

Analog Comparator

- Works like any Comparator
- Compares Ain0 (+ve) and Ain1 (-ve)
- Ain0 Voltage > Ain1 Voltage ➔ the Analog Comparator Output, ACO, is set.
- Can Trigger an Interrupt – Toggle OR RISING Edge or Falling Edge

Interrupt

- Breaks out from normal Flow of control to respond to an Interrupt Call
- sei();
- cli();
- #include <avr/interrupt.h>
- Example:- ISR(USART1_RX_vect)
- Types – External Interrupts, ADC complete, etc.
- External Interrupts – Toggle OR Rising Edge OR Falling Edge
- Very Powerful
- Better not to have Multiple Interrupts

Serial Communication

- USART
- SPI
- TWI or I^2C

- Asynchronous (No CLK) Serial Transmission
  -vs-
  Synchronous (NEEDS CLK SIGNAL) Serial Transmission

USART

- BAUD RATE
- FULL DUPLEX
- Async or Sync operation
- 5,6,7,8 or 9 data bits – LSB first
- 1 or 2 stop bits
- Even, Odd or No Parity – counts number of 1s
- TTL Logic Levels
- Interrupts or polling of status registers
- To Receive more than 1 byte == flush the data buffer (UDR) as soon as data comes in to avoid over write.

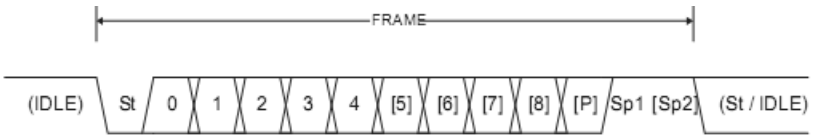
==========

- RS232 operates on +12v to -12V – GOOD for LONG Distance Communication
- MAX232 – RS232 Transceiver

==========

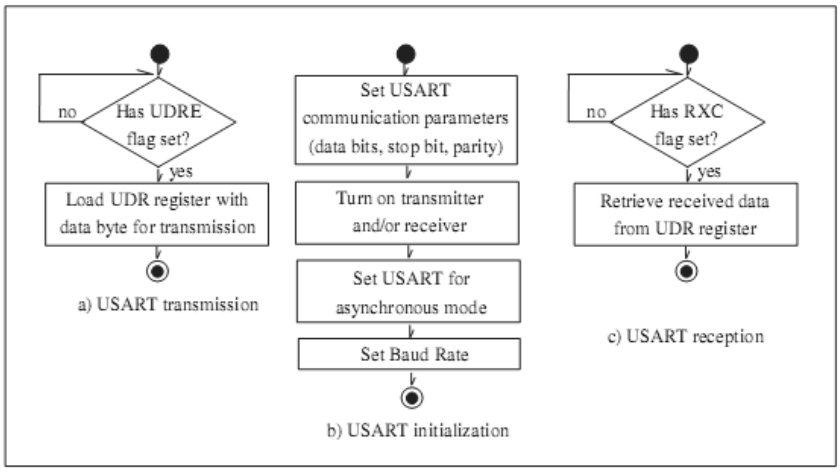
- FTDI – FT232RL- converts USB to TTL Serial

USART



Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

USART



USART

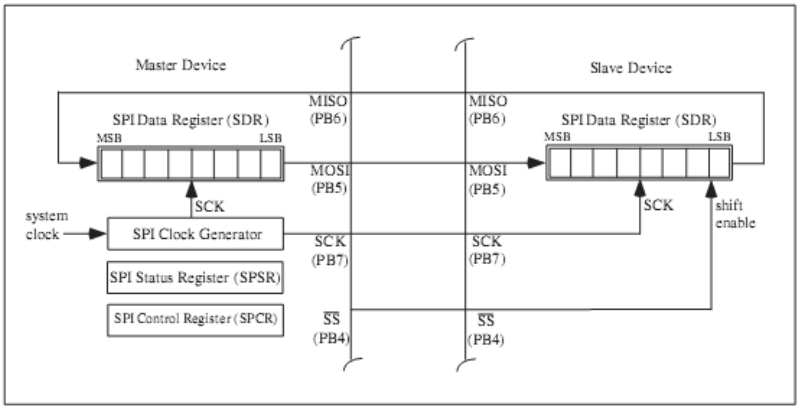
```
void UART1_Init( unsigned int UBRR1 )
{
        // UBRR = (F_CPU / (16 * BAUDRATE)) - 1

        /* Set the baud rate */
        UBRR1H = (unsigned char) (UBRR1>>8);
        UBRR1L = (unsigned char) UBRR1;

        UCSR0A = 0b00000000 ;
        // RXC0 = 0;  TXC0 = 0; UDRE0 = 0 ; FE0 = 0 ; DOR0 = 0;
        // UPE0= 0; U2X0 = 0 (dont need double speed UART) ; MPCM0 = 0
        UCSR0B = 0b00011000 ;
        // bit 7 : RXCIE0 = 0 (RX complete interrupt enable)
        // bit 6 : TXCIE0 = 0 (TX complete interrupt enable)
        // bit 5 : UDRIE0 = 0 UART data register empty interrupt enable
        // bit 4 : RXEN0 = 1 RX enable
        // bit 3 : TXEN0 = 1 TX enable
        // bit 2 : UCSZ02 = 0 (8 data bits - see intitlization values of UCSZ00 and UCSZ01)
        // bit 1 : Dont Care
        // bit 0 : Dont Care
          UCSR0C = 0b00000110;
        // bit 7 : Dont care
        // bit 6 : UMSEL0 = 0 (Asynchronous Operation)
        // bit 5 and 4 : UPM01 and UPM00 = 00 (no Parity bit)
        // bit 3 : USBS0 = 0 (1 stop bit)
        // bit 2 and 1 : UCSZ01 and UCSZ00 = 11 (8 data bits - UCSZ02  was intialized to 0)
        // bit 0 : dont care
}
```

SPI

- Synchronous – needs a CLK signal
- Master generates the CLK signal



SPI

- 8 bit transmit or Receive at a time.

SPI Control Register - SPCR

SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
7							0

SPI Status Register - SPSR

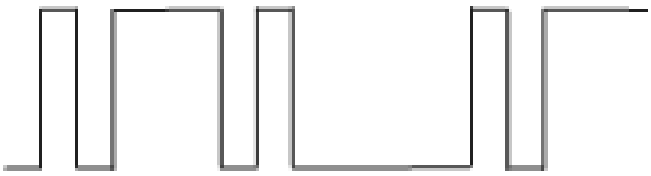
SPIF	WCOL	---	---	---	---	---	SPI2X
7							0

SPI Data Register - SPDR

MSB							LSB
7							0

TIMER

- Input Capture---Measuring External Timing Event
- Counting Events
- Output Compare -- Generating Timing Signals to Interface External Devices – e.g., PWM for motor drivers, SERVOs.



TIMER

Timer 0	Timer 1	Timer 2
- 8-bit timer/counter - 10-bit clock prescaler - Functions: – Pulse width modulation – Frequency generation – Event counter – Output compare - Modes of operation: – Normal – Clear timer on compare match (CTC) – Fast PWM – Phase correct PWM	- 16-bit timer/counter - 10-bit clock prescaler - Functions: – Pulse width modulation – Frequency generation – Event counter – Output compare – 2 ch – Input capture - Modes of operation: – Normal – Clear timer on compare match (CTC) – Fast PWM – Phase correct PWM	- 8-bit timer/counter - 10-bit clock prescaler - Functions: – Pulse width modulation – Frequency generation – Event counter – Output compare - Modes of operation: – Normal – Clear timer on compare match (CTC) – Fast PWM – Phase correct PWM

AVRLIB

- AVR-LIBC: Detailed descriptions of library files. Really useful!
  http://www.gnu.org/savannah-checkouts/non-gnu/avr-libc/user-manual/modules.html

ICSP

Symbol	Pins	I/O	Description
MOSI	PB5	I	Serial Data in
MISO	PB6	O	Serial Data out
SCK	PB7	I	Serial Clock

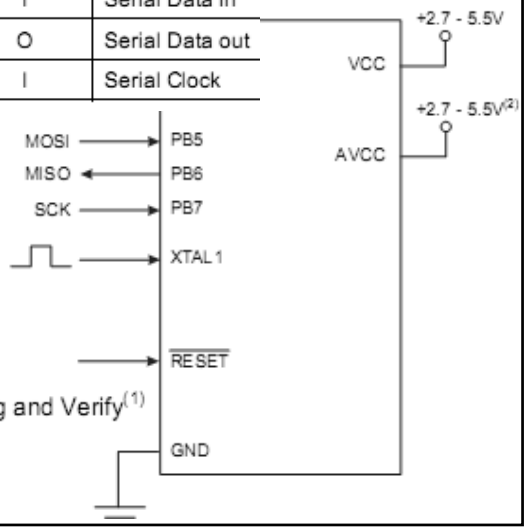


Figure 136. SPI Serial Programming and Verify(1)