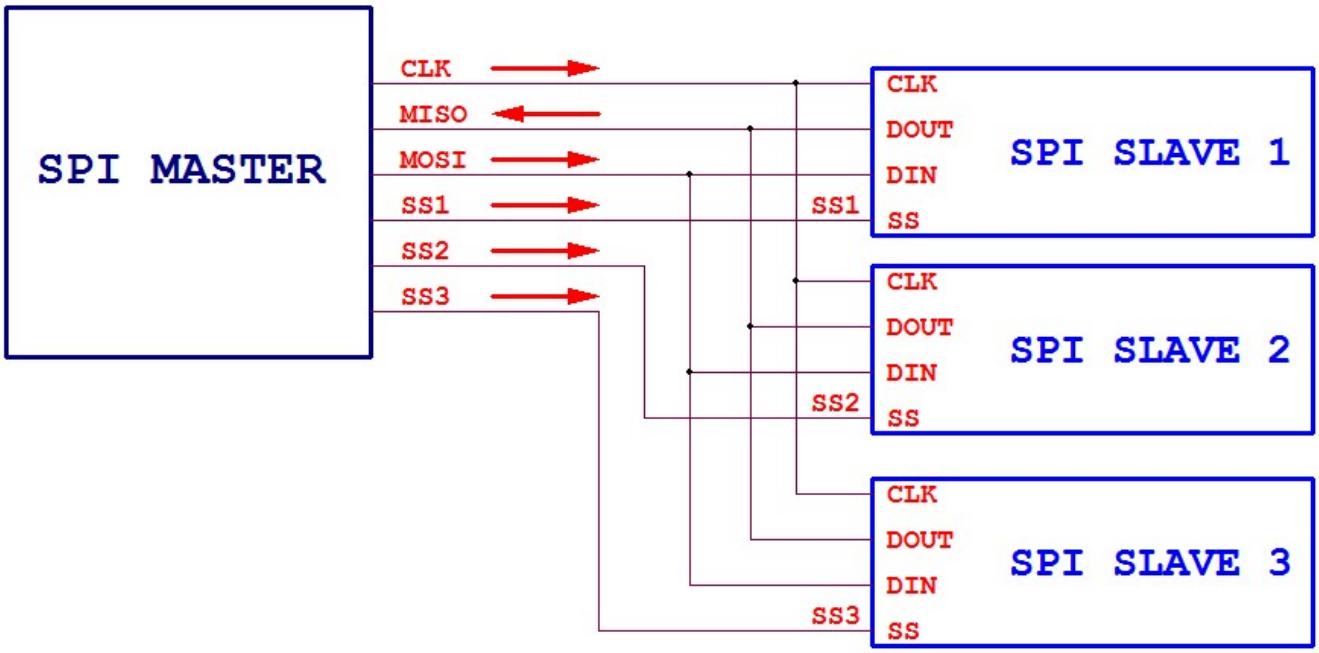
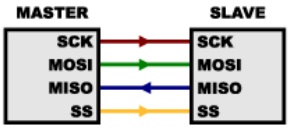
ADC (Analog Digital Converter) SU BUS SPI (Serial Peripherial Interface)

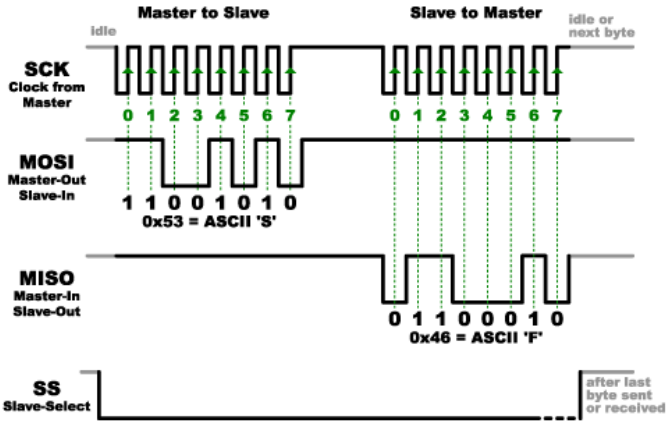
Il protocollo di comunicazione SPI, è uno standard di comunicazione seriale molto utilizzato nell'ambito dei microcontrollori, per interfacciarsi con altri dispositivi quali; memorie, convertitori ADC e DAC, sensori ed altri dispositivi.

La comunicazione avviene in maniera seriale su due piedini MISO e MOSI (input dati ed output dati) sincronizzati da un segnale di clock. Essendo una comunicazione adatta per il collegamento di più dispositivi sullo stesso bus di segnali, ogni dispositivo dovrà essere dotato di un Chip Select (CS, anche detto SS Slave Select) per decidere con quale dispositivo comunicare.

La comunicazione è di tipo Master-Slave, il master avvia la comunicazione con lo slave selezionando il relativo CS.







A fianco un esempio di comunicazione SPI tra un Master ed uno Slave, si può notare il pin SS andare a livello basso prima dell'avvio della comunicazione, che avviene con l'invio dei singoli bit sul canale MOSI, e la successiva risposta sul canale MISO.

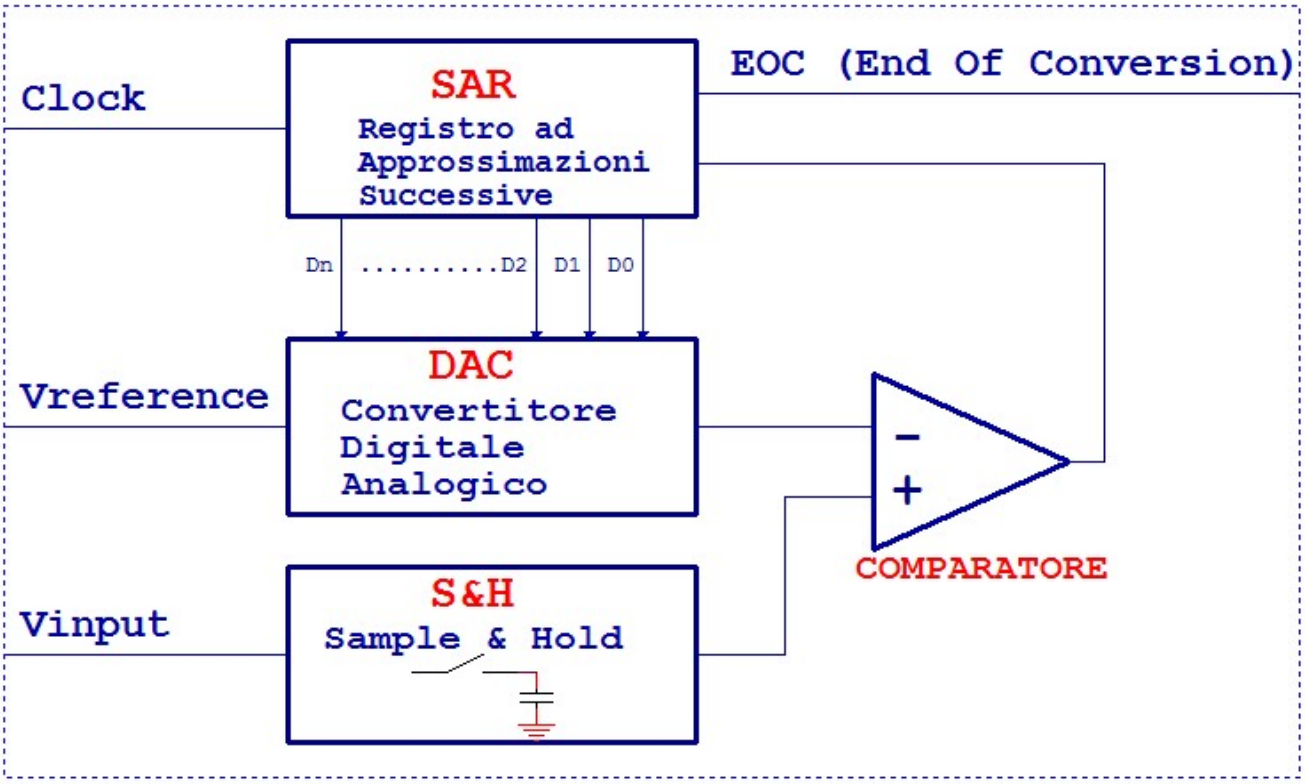
In entrambi i casi il sincronismo viene dato dal segnale di clock sulla linea SCK.

CONVERTITORE ADC MCP3008

Il circuito integrato MCP3008 è un convertitore ADC ad approssimazioni successive a 12 bit prodotto dalla Microchip.

Come tutti i convertitori ad approssimazioni successive, è strutturato internamente con i seguenti 4 blocchi utilizzati per la conversione:

- Circuito sample and hold
- Comparatore
- Registro SAR
- Convertitore DAC

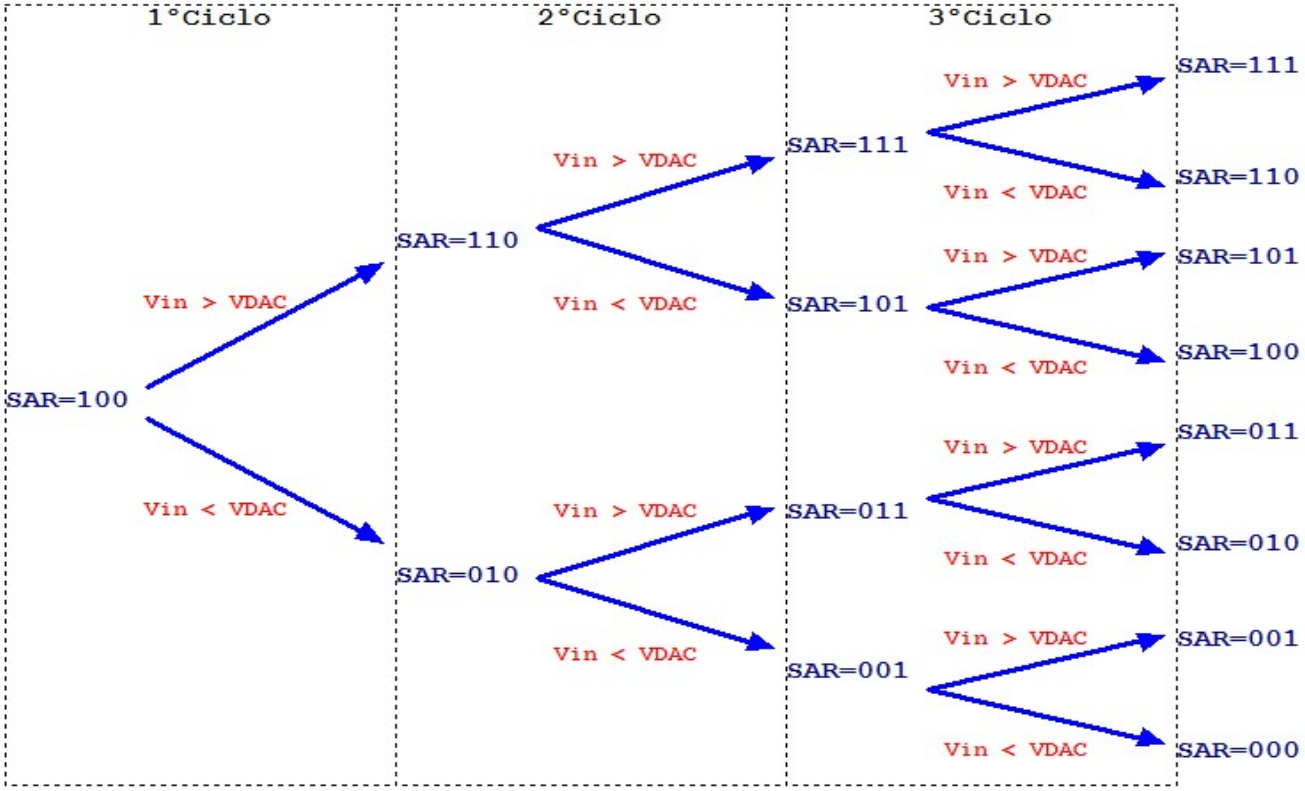


Apriamo una parentesi sul funzionamento di un convertitore ad approssimazioni successive.

In questo tipo di convertitore il valore binario corrispondente al segnale analogico di ingresso (Vinput) viene determinato confrontando il valore ottenuto dal codice binario generato dal SAR convertito dal DAC, ed il segnale Vinput.

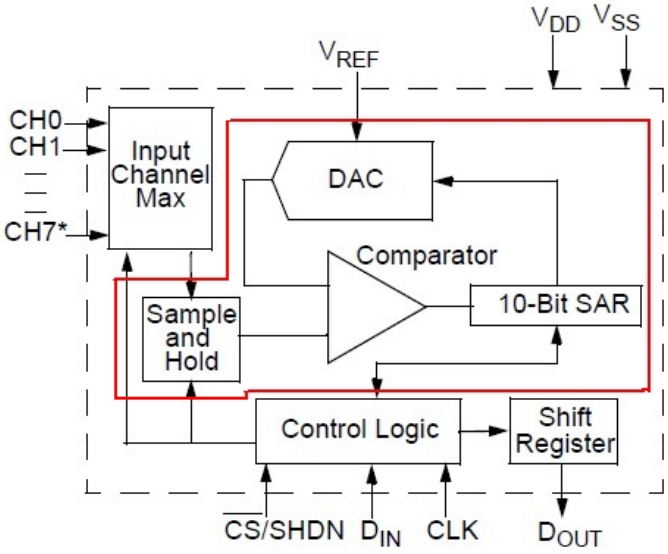
Per semplicità ipotizziamo che il SAR lavori con soli 3 bit D2,D1,D0, inizialmente viene posto il bit più significativo ad uno e nel primo ciclo viene comparato il valore del DAC (convertitore Digitale Analogico) con il segnale d'ingresso, in base al risultato ottenuto si decide di modificare successivamente il bit D1 e così via come nell'immagine di seguito.

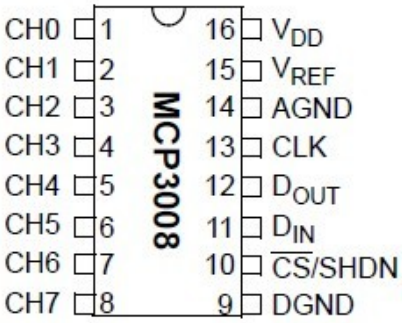
Al termine della conversione, nel registro SAR avremo il codice binario corrispondente al valore di ingresso. Ovviamente maggiore sarà il numero dei bit del SAR e maggiori saranno i cicli di confronto.



Nello schema a blocchi dell'integrato MCP3008, possiamo distinguere i blocchi sopra citati evidenziati in rosso, la parte in basso relativa alla comunicazione in SPI e la parte in alto sinistra dove sono rappresentati gli 8 canali di ingresso da CH0 a CH7.

Sotto il pinout del chip.



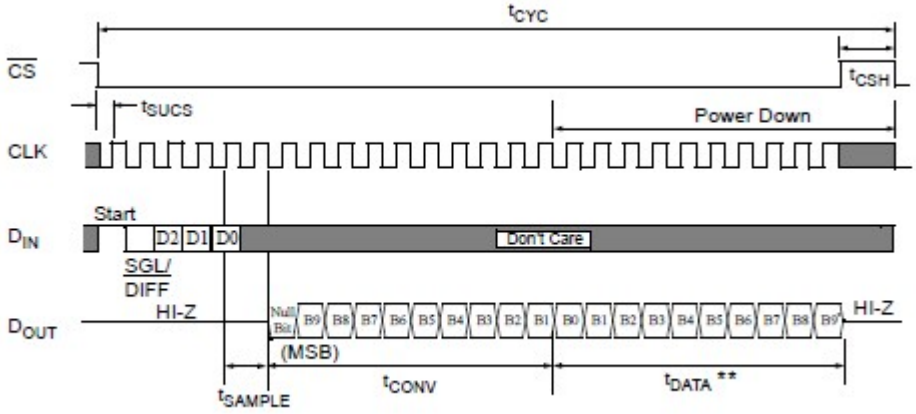


L'integrato va alimentato con una tensione di 5Vdc, che può essere anche utilizzata come Vref. Il pin 9 è la GND digitale e il pin14 quella analogica, in genere possono essere collegate entrambe allo stesso potenziale

Sul bus SPI, viaggiano i dati inviati al convertitore e ricevuti da esso, secondo le modalità brevemente descritte inizialmente.

Dal datasheet dell'integrato MCP3008, possiamo vedere come avviene lo scambio di informazioni.

La comunicazione viene avviata dal Master che dopo aver messo a livello basso il Chip Select (chiamato Slave Selet SS, sul protocollo SPI) invia 5 soli bit contenenti uno start bit, un segnale SGL/DIFF ed i 3 segnali D2,D1 e D0, come nella figura di fianco.



Sempre dal datasheet, troviamo la tabella di fianco, dove vediamo il significato dei bit inviati.

SINGLE/DIFF permette di scegliere se il canale d'ingressoè riferito a massa (SINGLE) o differenziale e cioè riferito al potenziale di un altro canale (DIFF).

Nel primo caso avremo la possibiltà di collegare 8 dispositivi analogici, uno per ogni canale, che danno una tensione variabile rispetto a GND.

Nel secondo caso, possiamo collegare invece 4 dispositivi analogici, che danno una tensione variabile tra due piedini, ad esempio CH0 riferito a CH1.

Il tipo di impostazione dipende ovviamente dal tipo di dispositivo che collegheremo all'integrato, consideriamo sempre che un segnale differenziale è maggiormente immune ai disturbi di un segnale analogico riferito a GND.

I restanti bit D2,D1 e D0 permettono di selezionare il canale che vogliamo utilizzare, ad esempio volendo leggere il canale 1, nei 3 bit dovremo mettere 000, e volendo collegare sul canale 1 un potenziometro che genera una tensione riferita a GND (come nell'esempio che segue) il bit SGL/DIFF dovrà valere 1.

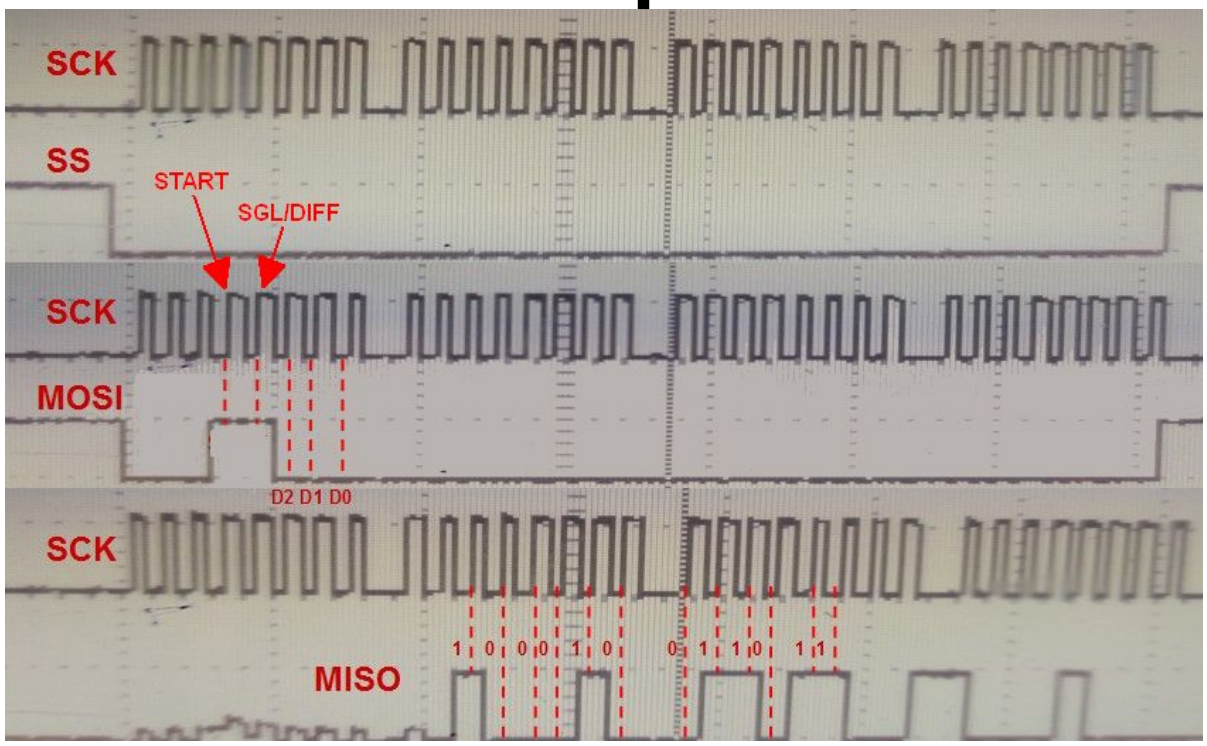
Control Bit Selections				Input Configuration	Channel Selection
Single /Diff	D2	D1	D0		
1	0	0	0	single-ended	CH0
1	0	0	1	single-ended	CH1
1	0	1	0	single-ended	CH2
1	0	1	1	single-ended	CH3
1	1	0	0	single-ended	CH4
1	1	0	1	single-ended	CH5
1	1	1	0	single-ended	CH6
1	1	1	1	single-ended	CH7
0	0	0	0	differential	CH0 = IN+ CH1 = IN-
0	0	0	1	differential	CH0 = IN- CH1 = IN+
0	0	1	0	differential	CH2 = IN+ CH3 = IN-
0	0	1	1	differential	CH2 = IN- CH3 = IN+
0	1	0	0	differential	CH4 = IN+ CH5 = IN-
0	1	0	1	differential	CH4 = IN- CH5 = IN+
0	1	1	0	differential	CH6 = IN+ CH7 = IN-
0	1	1	1	differential	CH6 = IN- CH7 = IN+

Dopo aver inviato i 5 bit, il Master riceverà la risposta dallo Slave, nel caso sotto il primo byte inviato sul canale MOSI è composto da 00011000 (18 Hex), selezionando così il canale 0, in quanto D2,D1 e D0 valgono 0.
Successivamente lo slave invia la risposta su due byte, la risposta è composta da 12 bit, che rappresenta la risoluzione del convertitore.
I bit sul canale MISO vengono inviati dopo due clock, pertanto nei due byte letti troveremo quanto segue:

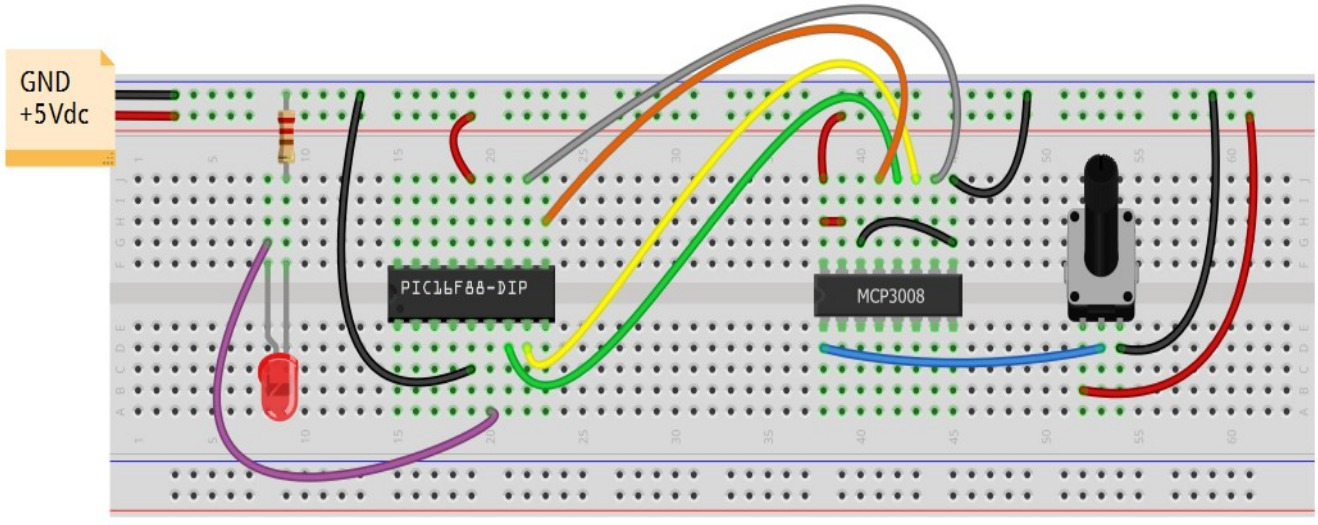
1° byte								2° byte							
		1	0	0	0	1	0	0	1	1	0	1	1		

Per avere il valore gustificato a destra bisognerà successivamente shiftare a destra di due posizioni.

1° byte								2° byte							
				1	0	0	0	1	0	0	1	1	0	1	1



CIRCUITO E PROGRAMMA DI ESEMPIO CON MICROCONTROLLORE PIC



Il programma di lato, è stato realizzato con il software mikroC della mikroelektronika.
Il pic è il 16F88, utilizzato con clock interno ad 8Mhz.

In questo caso il microcontrollre, legge il valore sul bus SPI dallo slave MCP3008, ed utilizza il valore per variare il ritardo di lampeggio di un LED collegato al pin RB0.

Il pin SS è collegato su RB5, il clock sul RB4 ed i segnali MISO e MOSI rispettivamente su RB1 e RB2

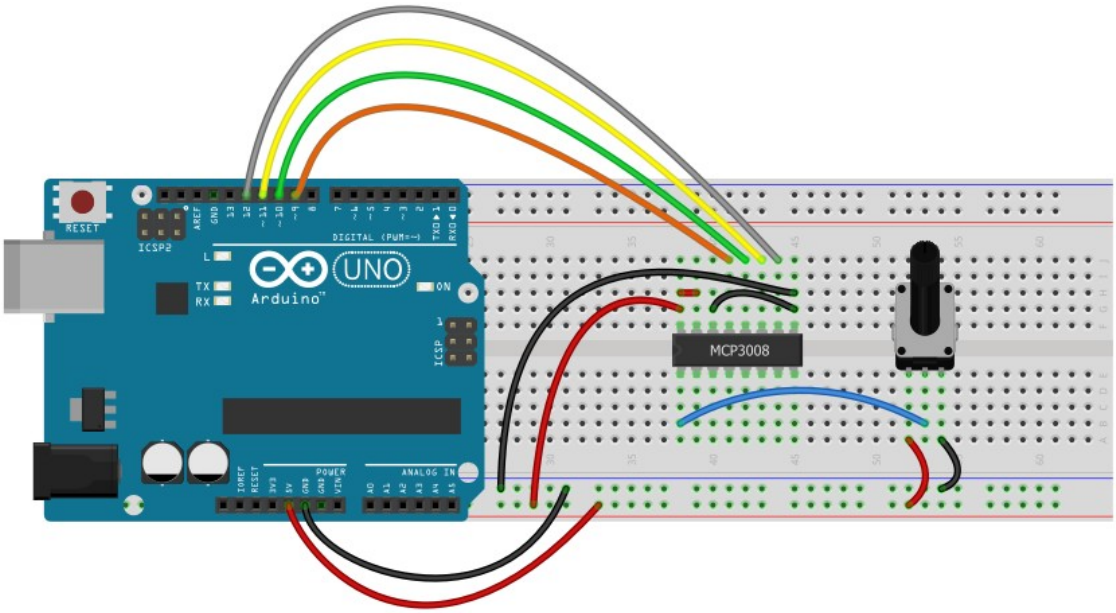
Nella routine ReadADC, viene inviato il valore esadecimale 18, che corrisponde a quanto visto precedentemente negli screenshot dell'oscilloscopio.
Successivamente vengono letti 3 byte, i primi due contengono i 12 bit del convertitore analogico.
Dopo aver shiftato il valore ottenuto di due posizioni come indicato in precedenza, il valore viene restituito al main.

Nel main, dopo le impostazioni generali viene effettuato il lampeggio del LED utilizzando il valore ricevuto dal convertitore.

```
sbit Chip_Select at RB5_bit;
sbit Chip_Select_Direction at TRISB5_bit;
sbit LED at RB0_bit;
sbit LED_DIR at TRISB0_bit;

unsigned short value1,value2,value3,buffer=0;
int tempo;

int ReadADC()
{
  int valore;

  Chip_Select = 0;                        // Select ADC chip
  SPI1_Write(0x18);     // channel 0
  value1=  SPI1_Read(buffer);  // read high byte
  value2 = SPI1_Read(buffer); // read low byte
  value3 = SPI1_Read(buffer);   // none
  Chip_Select = 1;                        // Deselect DAC chip
  valore=value1;
  valore=valore<<8;
  valore=valore|value2;
  valore=valore>>2;
  return(valore);
}
void main() {

ANSEL=0x00;      //disabilito i convertitori
CMCON=0x07;      //disabilito i comparatori
OSCCON=0xFC;     //imposto il clock interno a 8MHz
Chip_Select_Direction = 0;  //imposto il bit in uscita
LED_DIR=0;                  //imposto il bit in uscita
Chip_Select = 1;            //1=ADC non attivo
LED=0;                      //spengo il LED
SPI1_Init();
while (1)
 {
    tempo=ReadADC();    //lettura del valore analogico
    LED=1;              //accendo il LED
    while(tempo)        //ritardo variabile in base al valore letto
     {
       delay_ms(1);
       tempo--;
     }//while
    tempo=ReadADC();    //lettura del valore analogico
    LED=0;              //spengo il LED
    while(tempo)        //ritardo variabile in base al valore letto
     {
       delay_ms(1);
       tempo--;
     }//while
  }//while
}//main
```

CIRCUITO E PROGRAMMA DI ESEMPIO CON ARDUINO

Un analogo esempio si può realizzare con ARDUINO, di seguito schema e programma di esempio, per vedere il valore letto su monitor seriale.




```
#define CS_PIN 12
#define CLOCK_PIN 9
#define MOSI_PIN 11
#define MISO_PIN 10

void setup() {
    pinMode(CS_PIN, OUTPUT);
    pinMode(CLOCK_PIN, OUTPUT);
    pinMode(MOSI_PIN, OUTPUT);
    pinMode(MISO_PIN, INPUT);
    Serial.begin(9600);
}//setup


int readADC(int adcnum) {
  if((adcnum > 7) || (adcnum < 0)) return -1;

  digitalWrite(CS_PIN, HIGH);
  digitalWrite(CLOCK_PIN, LOW);
  digitalWrite(CS_PIN, LOW);

  int commandout = adcnum;
  commandout |= 0x18;
 for (int i=0; i<5; i++) {
   Serial.println(commandout);
   if(commandout & 0x80)
    {
     digitalWrite(MOSI_PIN, HIGH);
     Serial.println("*");
    }//if
   else  digitalWrite(MOSI_PIN, LOW);

   commandout <<= 1;
   digitalWrite(CLOCK_PIN, HIGH);
   digitalWrite(CLOCK_PIN, LOW);
 }//for

 int adcout = 0;
 for (int i=0; i<12; i++) {
   digitalWrite(CLOCK_PIN, HIGH);
   digitalWrite(CLOCK_PIN, LOW);
   adcout <<= 1;
   if (digitalRead(MISO_PIN))
     adcout |= 0x1;
 }//for
 digitalWrite(CS_PIN, HIGH);
  adcout >>= 1;
  return adcout;
}//read

void loop() {
  int val = readADC(0);
  Serial.println(val);
}//loop
```