13 Ottobre 2023

**1 Lezione Embedded**

Microcontrollore -> scatola che effettua dati dall’ esterno (input) e fornisce informazioni (output)

Diversi fili -> tutti programmabili input o output

Input -> leggo V

Output -> Effettuo V

Informazioni I o O che sia acceso o spento (1 o 0) per ogni filo

Filo -> programmabile in base a come mi seve -> programma sequenziale

Creazione Hardware dedicati -> solo funzione che mi serve -> monto nella sua piattaforma -> non si può aggiornare così e basta

Sistemi programmabili -> PC

FPGA -> chip logica programmabile (AND, OR, NOT) ->puoi programmare -> programmi app come vuoi -> sistem on chip (costo migliaia di €)

Semi controllori -> costo non troppo alto

Microprocessore -> 128 fili nel PC dotato di bus -> numero di fili -> collegano diversi file -> 0 o 1 -> bus -> che si collegano con la memoria (bus address) -> dove capita dato -> collegi dati all’indirizzo interessato

Bus collegato RAM -> trasferisci dao in memoria e così via -> vai sistemi -> e conta in binario -> logica Input e Output

Processore -> parte di elettronica, molte potenze di calcolo -> Gh2 movimento

Micro controllore -> diversi fili, RAM dentro chip del controllore, e altre periferiche -> già sistema di elaborazione -> Mh2 movimento -> architettura più semplice

Architetture micro Controllore -> Harvard

Architetture Microprocessore -> von-Neumann

Microcontrollore -> sistemi embedded, programmatici da noi -> bug

Watchdog -> cronometro interno, avvia accensione macchina -> incrementato ms e arriva in un secondo (1000 macchina resetta) , resetta watchdog non arriva mai a 1000 ms -> sistemi gestisca questa -> evitare bug

Microcontrollori -> 8,16,32 bit -> dimensione bus sistema -> dato che devo manipolare -> spacchetto dati in moduli da 8 -> lavorare char e byte -> non int -> riduce manipolazione dati -> se no rallenti codice -> devo possibile ridurre

Memoria -> < 8kB

Programma -> oscillatore

Ogni operazione -> 4 colpi di clock -> 4 impulsi -> clock 4 MH2 -> istruzione 1 ms

Micro Controllori -> basso costo -> saper tempo di impiego per saper svolgere il tempo di esecuzione funzione / metodo

Circuito di clock -> circuito si usa molto ma non è buono

Soluzione più precise -> componente quarzo insieme alla resistenza -> compensatore -> costante carica e scarica condensatore

Condensatore a quarzo -> sincronizza subito sbaglia dopo anni -> risincronizzare

Circuito condensatore RC e con il quarzo -> quarzo più preciso però costo maggiore

Hardware -> costare il meno possibile

EEPROM -> memoria a cui tiene memoria i dati -> posso programmarla elettricamente -> accettare dati e memorizzo e mi dice i dati senza perderli e cancellabili

Micro Controllore che si usa: PIC16F628A

Massimo 20 MHZ ,5 kB memoria -> non più di 3500 righe, 224 byte memoria, conosci 35 istruzioni -> massimo 35 istruzione, interupt -> blocca codice per avviarne un altro e riprende da dove aveva ripresa prima, stack dove salva dati con 8 livelli

Chip -> 16 fili, 13 programmabili I/O, 2 Alimentazione, 3 Quarzo, 2 clock, 1 poter programmare

Sia A che B hanno informazioni -> vedere se è alto oppure se è basso

2 parametri -> ogni settore -> livello logico alto o basso -> fornito macchina o esterno

Diversi gruppi -> con parametri divisi in byte con i bit -> divisi in piedini -> posso impostare ma con delle difficoltà -> imposto I/O -> altro H/L -> acceso o spento -> memoria azione diretta sul registro

I/O -> registro tris

H/L -> registro port

Ogni bit -> definisce input o output -> registro tris

Ogni bit -> definisce alto o basso -> registro port

Ci sono 4 registri

* TRISA
* PORTA
* TRISB
* PORTB

Scrivere il numero che viene fuori in binario tradotto in decimale

**Registro TRIS**

1 -> input

0 -> output

**Registro PORT**

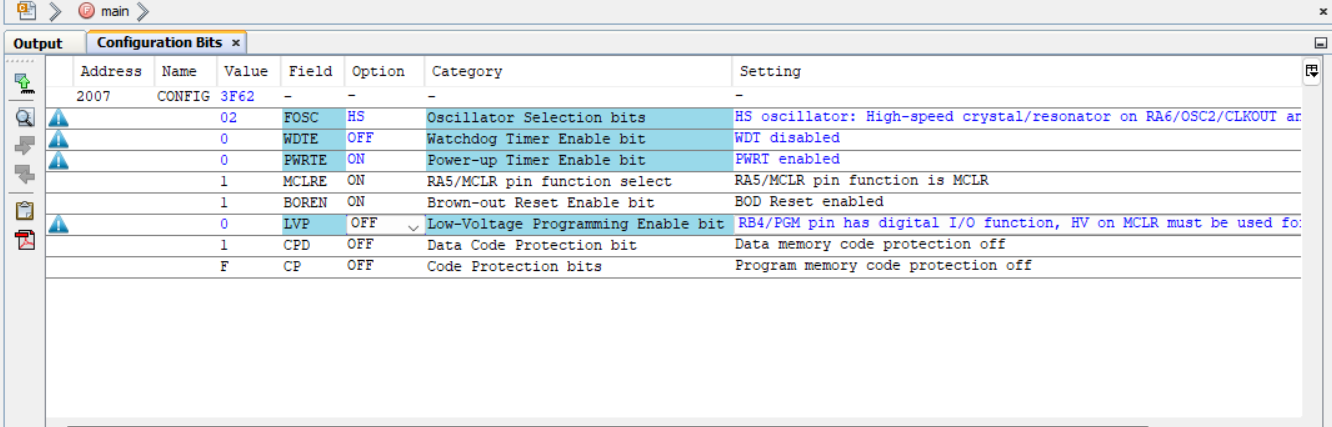
1 -> high

0 -> low

Se scrivo 0b davanti dopo metto il numero italiano

Se scrivo ox davanti dopo metto il numero esadecimale

Pagina configurazione



Righe configurazione

// CONFIG

#pragma config FOSC = HS // Oscillator Selection bits (HS oscillator: High-speed crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = ON // Power-up Timer Enable bit (PWRT enabled)

#pragma config MCLRE = ON // RA5/MCLR pin function select (RA5/MCLR pin function is MCLR)

#pragma config BOREN = ON // Brown-out Reset Enable bit (BOD Reset enabled)

#pragma config LVP = OFF // Low-Voltage Programming Enable bit (RB4/PGM pin has digital I/O function, HV on MCLR must be used for programming)

#pragma config CPD = OFF // Data Code Protection bit (Data memory code protection off)

#pragma config CP = OFF // Code Protection bits (Program memory code protection off)

X && Y -> and

X != Y -> not

X || Y -> or

X & Y -> and per singola entità, converti i dati in binario

X | Y -> and per singola entità, converti i dati in binari