Text, letter

Description automatically generated**UART (Universal Asynchronous Receiver Transmitter)**NON dipende dal clock. La velocità di comunicazione va accordata prima. FULL-DUPLEX.  
L’UART comunica solo P2P (1-1). Funziona con i pin TX e RX. Ha dei pin opzionali (RTS *request to send,* CTS *clear to send*).  
Per funzionare si connette TX(A) e RX(B) e viceversa, devono avere la stessa GND line.  
Proprio RTS a 1 -> richiedo di inviare. Proprio CTS a 1 -> accetto di ricevere.  
**Trasmissione UART**  
Durante l’idle la TX è a 1.  
Inizio trasmissione: si mette a 0 TX per il tempo di 1 bit (start bit). Questo indica al ricevitore che stanno per arrivare dati.  
Vengono inviati i bit uno ad uno (LSBit-first).  
Alla fine dei data bits può esserci un parity bit opzionale.  
Viene poi impostata a 1 la TX e trattenuta per 1 o 2 bits (stop bit).  
Lo start e lo stop bit non hanno significato nei dati. Trasmette un byte alla volta!  
Durata di trasmissione di 1 bit -> . Il tempo del BYTE è Il baud-rate può assumere valori diversi (300, 1200, 2400, 4800, **9600**, 14400)  
Esempio [ *(microsecondi), quindi TByte è circa 1 ms]***Registri Uart**UART utilizza diversi registri: UxMODE = configurazione, UxSTA = stato, UxTXREG = Trasmissione, UxRXREG = Ricezione, UxBRG = baud-rate generator. Ogni bit dei registri ha una funzionalità specifica.  
*UartBrg=(int)(((float)PbusClock / ( 16 \* baud ) − 1 ) + 0.5) ; // add 0.5 to round*

**Timer**I timer lavorano come contatori, incrementando il valore del registro TMR fino ad arrivare al valore di PR (*Period Register*).  
PR ha 16 bit quindi un valore massimo di 216-1.

dove PBLCK è la frequenza.

Registri Timer:

* TCKPS=Imposta il prescaler[1,8,64,256] nel 32 bit si ha anche 16 e 32 come valore
* T32=1 ->timer a 32 bit; 0-> timer a 16 bit
* PRx= valore Massimo calcolato con Treset
* TMRx=valore del timer

**Interrupt**

Nelle ISR non dovrebbero esserci chiamate ad altre funzioni. In fase di configurazione delle periferiche bisogna abilitare gli interrupts.

**Clock**

Struttura PLL

1. Prescaler (Divisione del segnale per portarlo tra 4 e 5 MHz)
2. PLL molt (Moltiplicazione del segnale di clk di un valore di scalatura PLLmolt )
3. Postscaler (Divisione ulteriore del segnale)

**PMP**

Consente la comunicazione parallela. Vengono usati per comunicare con dispositivi LCD, periferiche di comunicazione, dispositivi di memoria.

È un modulo che ha 8/16 bit di input e output, e 8/16 bit di bus indirizzo

Pin:

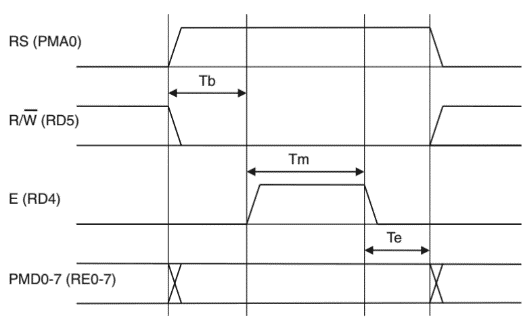
* Enable
* Lettura->PMDIN (32bit)
* Scrittura->PMDOUT (32bit)

Il bit Busy se 1: è in corso una trasmissione di un byte, 0 è libero

**PMP e LCD**

I segnali usati dal LCD sono:

* E: Enable
* R/W: read(1)/write (0)
* RS: register select(indirizzo)
  + 1 se trasferimento dati
  + 0 se trasferimento istruzioni

LCD è una periferica lenta ed è necessario temporizzarla.

* tempo di attesa prima della lettura o scrittura di un dato
* tempo di attea tra l’abilitazione di R/W
* tempo di attesa dopo enable

Per configurare il PMP va configurato come master, abilitato (richiede tempo)

* Tb: attesa data setup prima della scrittura
* Tm: attesa tra scrittura e enable,
* Te: attesa data setup dopo enable.

Data setup: si intende la scrittura o lettura sulla periferica

Dopo ogni operazione bisogna attendere un determinato tempo (la periferica è lenta).

Immagine che contiene tavolo

Descrizione generata automaticamenteRS=1 data transfer RS=0 instruction transfer

In fase di lettura bisogna attendere che si liberi prima l’LCD perché più lento e poi PMP, dopo di che si può scrivere.

Dopo ogni comando bisogna attendere un determinato tempo, dipende dal comando.

Spostare cursore si usa set DDRAM, in ACn ci va la posizione in cui vogliamo mettere il cursore.

Immagine che contiene testo

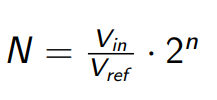
Descrizione generata automaticamentemuovere il cursore cmdLCD(0x8[indice della posizione 0…F]) esempio cmdLcd(0x82), pone il cursore in posizione 3

Per andare a capo con scrittura cmdLCD(0x80|0x40) 0x80 abilita la scrittura, se i mette solo 0x40 si muove solo il cursore

**ADC**

**Quantizzazione**

Assegnare il valore (ampiezza) rilevato al tempo di campionamento con un valore quantizzato (cioè una delle 2n parole binarie). Il numero di n bit dipende dallo sviluppatore.

N=valore digitale del segnale di ingresso, Vin= tensione input, Vref= tensione di riferimento, n numero di bit per codificare.

Vfs è solitamente 5-10V ed è l’ampiezza di input massima. Vlsb è la variazione di ampiezza che se raggiunta fa cambiare la parola  **.**

**Formula di conversione dati** =

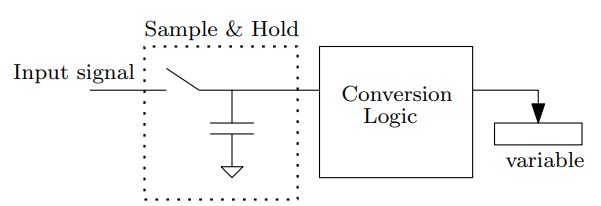
Errori ADC:

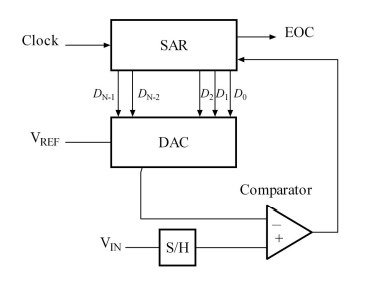
* Offset error: valore non 0 anche se l’input è 0
* Gain error: valore di R diverso da quello atteso
* Linearity error:
  + Differenza di non linearità: quantizzazione con step di ampiezza variabile
  + Integrale non lineare: quanto differisce l’output dalla curva ideale

L’ADC converter ha:

* Contatore: consente di contare da 0 a 2n
* Comparatore: riceve in input 2 segnali analogici (Vin e Vsoglia) e produce un output digitale Z se Vin>Vsoglia=>Z=1
* Integratore: calcola l’integrale in tempo del segnale
* Sample & Hold: è un circuito per campionare l’input e mantenere il valore fino alla conversione

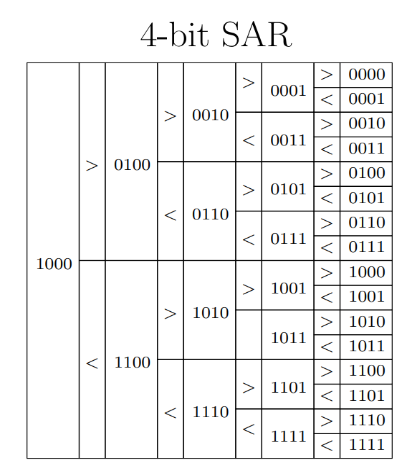
tensione condensatore

Nel SAR i bit inizialmente sono tutti a 1

Conversione:

* Prendi l’input e chiudi il circuito (Simple&hold) il segnale di input è nel condensatore
* Compara il segnale con il segnale DAC (converte da digitale ad analogico)
* Invia i dati al SAR: Se Vdac > Vin resetta il bit, altrimenti metti 1
* Effettua fin quando non si controllano tutti i bit
* Ricevi il nuovo segnale dal S&H Il campionamento può essere:
* Manuale: bisogna gestire le operazioni di “reset” manualmente
* Automatico: dopo ogni campionamento ne parte un altro

In pic32 c’è un bit done viene resettato in automatico dall’hardware quando parte il sample, il risultato viene posto in un registro a 32 bit. EOC è un flag che viene abilitato a fine conversione

**Output Compare**

Permette la generazione di impulsi di una durata specifica. Duty cycle: è il rapporto percentuale tra tempo on del segnale e tempo totale gli impulsi possono essere singoli o treni continui di impulsi.

Il valore analogico è proporzionale al duty cycle, VH è il voltaggio quanto si ha valore alto.

Periodo PWM il periodo non deve superare l’ampiezza di PRx (16 o 32 bit) se T è molto grande scegliere un prescaler più grande. La risoluzione massima del PWM indica il numero massimo di bit necessario a rappresentare un valore massimo

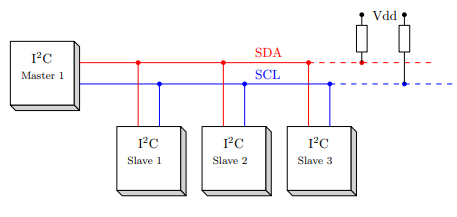
Output compare può funzionare in 3 modi:

* **Single compare:** 
  + Output drive high: quando il timer raggiunge OCR imposta il segnale OCx a 1
  + Output drive low: quando il timer raggiunge OCR imposta il segnale OCx a 0
  + Output drive Toggles: quando il timer raggiunge OCR effettua il toggle del segnale OCx
* **Dual compare:**
  + Single output pulse: genera un impulso quando si trova tra OCR0 e OCR1
  + Continuous output pulses: come il single, ma si ripete
* **PWM mode**:
  + Con Fault protection input
  + Senza Fault protection input

In PWM mode il segnale è 1 subito all’inizio di PR per il valore definito in OCxR, il duty cycle lo si ottiene , per modificare OCxR, da software modifico OCxRS e dopo un x tempo l’hardware lo aggiorna in OCxR

Tutte le modalità usano un timer da 16 o 32 bit. In OCRxR c’è il valore del OCR che consente la generazione del segnale in OCRxRS lo si usa se si usa il dual compare

**I2C**

È un protocollo di comunicazione seriale half duplex sincrono, il clock viene generato dal master e condiviso con gli slaves che rispondono alle richieste del master.

SDA: serial data, bidirezionale, viene inviato un bit alla volta

SCL: serial clock, unidirezionale, fornisce il clk per la comunicazione

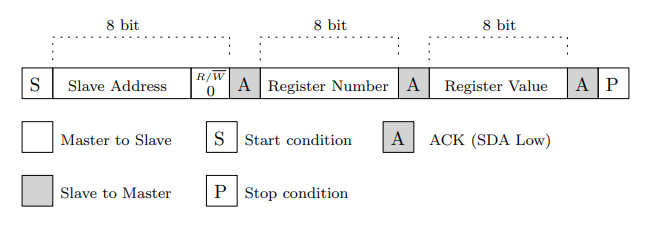
Le resistenze sono dette resistenze di pull-up e in idle le linee SCL e SDA sono a 1

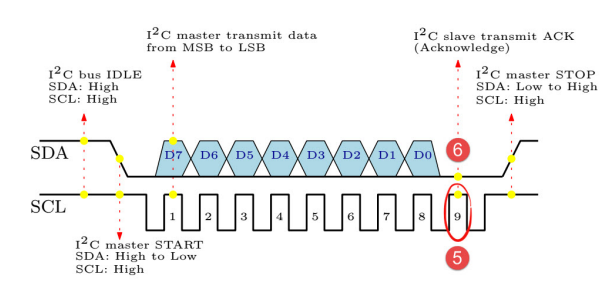
Ogni slave ha 7 o 10 bit di indirizzo e un register set di 8 bit.

Ogni registro può essere letto o scritto dal master, velocità 5,3.4, … Mbits

Tpgd ha un valore fisso(datasheet) di 104 ns

I dati vengono inviati dal bit più significativo al meno significativo MSB -> LSB

Schema di trasmissione

* IDLE: SDA e SCL sono a 1 (High)
* Start condition: SDA la metto a 0(Low)
* SCL è low mentre si impostano i dati su SDA
* Stop condition: Dopo la trasmissione dei dati il master genera un ulteriore impulso di clock e rilascia SDA
* Lo slave lascia SDA a low se riceve il dato in modo corretto invia un ACK
* SDA e SCL tornano HIGH

in fase di lettura il master se ha terminato la lettura invia un NACK

Lettura:

* [Master] Start condition
* [Master]Invia slave address in scrittura (ultimo bit a 0), Slave] invia ACK
* [Master] Invia numero registro, [Slave] invia ACK
* Master restart (nuova start condition)
* [Master]Invia slave address in lettura (bit a 1), [Slave] invia ACK
* [Master] ricevi i dati (ultimo 8 bit), [Slave] invia i dati
* [Master] invia ACK
* [Master] continua a ricevere fin quando non invia NACK
* Stop condition

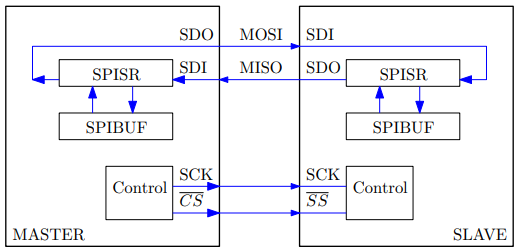
Scrittura

* Master Start condition
* [Master]Invia slave address+0 scrittura o 1 lettura, [Slave] invia ACK
* [Master] Invia numero registro, [Slave] invia ACK
* [Master] scrivi i dati, [Slave] riceve i dati
* [Slave] invia ACK
* Stop condition

L’ACK quando viene inviato SDA rimane a low, NACK SDA va high al 9 bit.

**SPI**

È un protocollo di comunicazione seriale sincrono full duplex.

* **SCK:** Serial Clock
* **SDO:** Serial Data Output
* **SDI:** Serial Data Input
* **SPISR:** Shift register: vengono messi i dati di input e di output
* **SPIBUF:** Serial Buffer: i dati vengono messi lì una volta terminato il trasferimento dati(input) o prima della trasmissione(output)
* **CS:** Chip select consente di selezionare lo slave con coi comunicare, ogni slave ha un collegamento con il master
* Immagine che contiene testo

  Descrizione generata automaticamente**MOSI:** Master Output Slave Input
* **MISO:** Master input Slave Output

**Caratteristiche**

* Ha bisogno di (3 + n) line (n: numero di slave)
* Aggiunge un dispositivo sul bus ha bisogno di una linea di selezione dedicata
* Protocollo più semplice
* Full duplex: riceve mentre si trasmette
* Push-Pull permette di avere linee più lunghe e velocità più elevate (decine di MHz è comune)

SPI è full duplex e scrive e legge sul fianco ascendente/discendente del clock, i dati vengono inviati dal bit meno significativo al più significativo LSB to MSB, dato che verranno shiftati dallo shift register.

Immagine che contiene testo, cielo, orologio, diverso

Descrizione generata automaticamentein pic32 richiede un rimappaggio dei pin, SPI è usato per dispositivi di memoria (flash, eeprom, sdcard …) ecc...

SPIxBUF è un registro a 32 bit che corrisponde al SPIBUF

In fase di trasmissione con un determinato slave il chip select è a 0