## Periferiche

### PIO

Il RP2040 è dotato di una periferica detta PIO (Programmable Input Output) che consiste in 8 macchine a stati finiti, separate in 2 blocchi e programmabili via software. Ogni blocco è interfacciato con il bus di sistema e permette di impostare 4 macchine a stati finiti le cui uscite sono assegnabili a qualsiasi GPIO tramite la matrice di assegnazione. Ogni macchina a stati è programmabile e permette di implementare brevi funzioni e/o interfacce specifiche (es: VGA, DPI, I2C, SPI, ...) grazie all’elevata specializzazione dell’hardware che permette di ottenere temporizzazioni precise e un’efficiente integrazione con il resto del sistema microcontrollore.

Di seguito è illustrato lo schema di un blocco PIO:

Diagram

Description automatically generated

Ogni macchina a stati ha accesso ai:

* 2 shift register bidirezionali da 32 bit (shift programmabile)
* 2 registri general purpose ausiliari da 32 bit
* 8 code FIFO da 32 bit configurabili sia in ingresso che in uscita
* Divisore di clock frazionale (16 bit per la parte intera, 8 per quella frazionaria)
* Interfaccia DMA
* Interfaccia agli interrupt
* Accesso indipendente e simultaneo a 30 pin GPIO

#### Programmazione

La programmazione avviene caricando il codice programma in una memoria condivisa a cui le MSF di un blocco hanno accesso. Si noti il sistema ha accesso in sola scrittura a questa memoria e che la programmazione avviene a runtime! Una volta caricato il programma, configurate le MSF e impostato il mapping I/O, le macchine a stati sono autonome e interagiscono con il programma principale tramite DMA, interrupt e registri appositi, agendo di fatto come una sorta di processore secondario.

Le implementazionei di interfacce comuni, quali UART, SPI e I2C, sono già disponibili nella libreria PIO, se invece si vuole realizzare qualcosa di più specializzato o flessibile, lo sviluppatore ha a disposizione una breve lista di istruzioni simil-assembly:

* JMP
* WAIT
* IN
* OUT
* PUSH
* PULL
* MOV
* IRQ
* SET

Questa lista sembra breve all’apparenza, ma in realtà è un insieme di istruzioni piuttosto potenti che permettono di eseguire qualsiasi tipo di programma, specialmente perché esistono più versioni di ogni istruzione, con diversi argomenti od opzioni che la estendendono virtualmente.

Per tradurre un programma testuale scritto utilizzando queste istruzioni in un programma binario, il SDK mette a disposizione l’assemblatore pioasm.

#### Esecuzione

Ogni istruzione viene eseguita in un singolo ciclo di clock, e può specificare un ulteriore ritardo fino a 31 cicli. L’istruzione wait, che attende una certa condizione prima di proseguire nell’esecuzione del programma, è un esempio di eccezione a questa regola.

Di norma il program counter (PC) è incrementato di un’unità ad ogni ciclo di clock, tornando all’inizio del programma una volta raggiunta la fine della memoria. Tramite l’istruzione JUMP è possibile alterare questo comportamento e far puntare il PC ad un’istruzione qualsiasi, alterando di fatto il flusso di esecuzione predefinito.

Oltre alla memoria programma, sono disponibili altri mezzi per eseguire istruzioni:

* Scrivendo istruzioni sul registro di configurazione SMx INSTR, queste sono immediatamente eseguite interrompendo il normale flusso di esecuzione
* Tramite l’istruzione MOV EXEC è possibile eseguire un istruzione da un registro
* Tramite l’istruzione MOV EXEC è possibile eseguire un istruzione dallo shift register di uscita

#### Registri

Come già anticipato, ogni macchina a stati ha accesso ad alcuni registri sia per il trasferimento di dati, sia per valori temporanei. Questi sono:

* OSR (Output Shift Register)
* ISR (Input Shift Register)
* X
* Y
* TX FIFO
* RX FIFO

Diagram

Description automatically generated

L’OSR è un registro intermediario tra la TX FIFO e una destinazione che può essere i pin di uscita o altri registri. La scrittura avviene con l’istruzione pull che sposta 32 bit dalla TX FIFO all’OSR, mentre con l’istruzione out è possibile leggere (shiftare) da 1 a 32 bit verso una destinazione a scelta ed i bit restanti dopo lo shift sono posti a zero. La direzione di operazione è reversibile ed impostabile tramite registri appositi.

È inoltre disponibile la funzione autopull, per cui se vengono letti in totale un numero specificato di bit, l’OSR è scritto automaticamente con nuovi dati dalla FIFO. Questo permette di risparmiare un’istruzione di pull e quindi di aumentare il throughput fino ad 1B per ciclo di clock di sistema

Note x presentazinoe:

Una sorta di processore secondario

Programmabile runtime

Limite 32 istruzioni

interfaccia custom, esempio altre SPI o I2C o LED indirizzabili WS2812, driver brushless