TASTIERA E LA PS2

INFORMAZIONI DI CARATTERE GENERALE

Il presente blocco implementa un modulo per la comunicazione via porta *PS2* tra una *tastiera* (*PS2* o *AT*/*XT*) e la scheda *Xess XSA-50*. Questo lavoro si appoggia in parte a quello svolto dall'università del *Queensland* (http://www.uq.edu.au/).

L'idea che sta alla base è quella di avere un modulo duttile che possa interfacciarsi ad un'eventuale logica presente sull'*FPGA* fornendole valori (numeri o stringhe) suddivisi per parametri.

Nella prossima sezione viene presentato in sintesi il protocollo PS2 (conoscenza imprescindibile per capire il funzionamento del blocco).

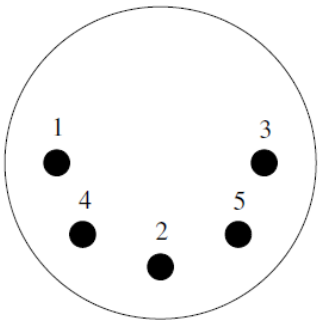
IL PROTOCOLLO PS2

INTRODUZIONE

Il *PS2* è un *protocollo seriale*, *sincrono* e *bidirezionale* sviluppato dall'*IBM* e ampiamente usato al giorno d'oggi per far comunicare *PC* con *tastiere* e *mouse*.

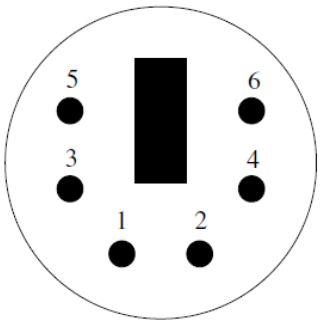
La scheda *XSA-50* ha in dotazione una porta *PS2* a 6-*pin Mini-DIN*. Per le porte *PS2* esiste anche un altro connettore, il 5-*pin DIN* (*AT*/*XT*): i due connettori sono (dal punto di vista elettrico) perfettamente compatibili (il *protocollo* è sempre il *PS2*, l'unica differenza sta nella disposizione dei *pin*), per cui nel caso si possedesse una tastiera con connettore a 5-pin è sufficiente l'uso di un adattatore.

5 pin DIN (AT/XT)



1. clock
2. data
3. non implementato
4. massa
5. +5V

6 pin mini–DIN (PS2)



1. data
2. non implementato
3. massa
4. +5V
5. clock
6. non implementato

LA COMUNICAZIONE

Poiché tramite il *protocollo PS2* è possibile gestire l'*I*/*O* da tastiera con molti dispositivi, per mantenere una certa generalità in questo contesto ci si riferirà alla scheda *XSA*-50 con il generico termine di *host*.

Come si può capire dalla precedente figura, tutta la comunicazione si basa su un *bus* formato da *due linee bidirezionali*: la linea di *clock* e la linea *dati*. Il *bus* è in uno stato di attesa quando le due linee di *clock* e di *dati* sono in uno stato logico alto: questo è l'unico stato nel quale è permesso alla tastiera trasmettere dati. L'*host* ha sempre il

controllo finale sul *bus* e può in ogni momento inibire la comunicazione mettendo a *massa* la linea di *clock*; alla tastiera invece (e solo a lei) spetta il compito di *generare* il *clock*.

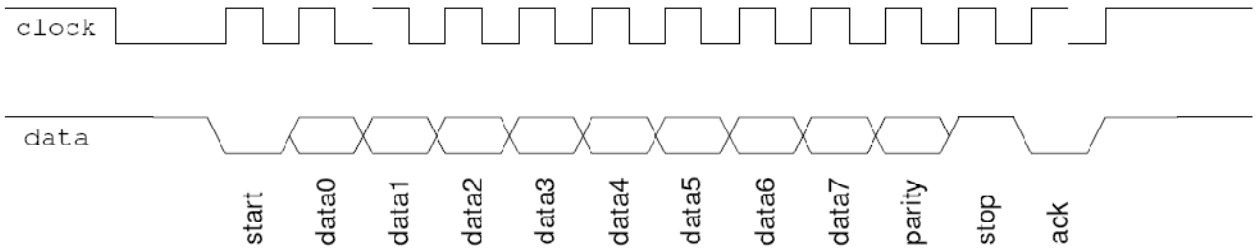
I dati spediti dalla *tastiera* all'*host* sono letti sul *fronte di discesa* del *clock*; viceversa i dati trasmessi dall'*host* alla *tastiera* vengono letti sul *fronte di salita*. La *frequenza* del clock deve stare all'interno del range 10,0 ÷ 16,7 kHz. Questo significa che il clock deve rimanere alto per 30 ÷ 50 µs. Tali parametri sono assolutamente cruciali e agire su di essi è estremamente delicato: questo è il motivo per cui si è scelto di adattare il progetto della *Queensland University* agendo su un divisore di clock a monte in modo da rendere le temporizzazioni meno strette.

XSA-50 → TASTIERA

Se l'*host* vuole trasmettere dei dati deve innanzitutto inibire la comunicazione mettendo a *massa* il *clock* per almeno 100 µs, quindi mettere a *massa* anche la *linea dati* e *rilasciare* il *clock*: questa sequenza fa capire alla tastiera che sono in arrivo dei dati e quest'ultima inizia a *generare* il *clock*.

Tutti i dati vengono trasmessi in modo seriale un *byte* alla volta e ogni *byte* viene inserito in un *frame* di undici bit:

- 1 bit di start. Questo bit è sempre posto a 0;
- 8 bit di dati, ordinati dal meno significativo;
- 1 bit di parità;
- 1 bit di stop. Questo bit è sempre posto a 1.

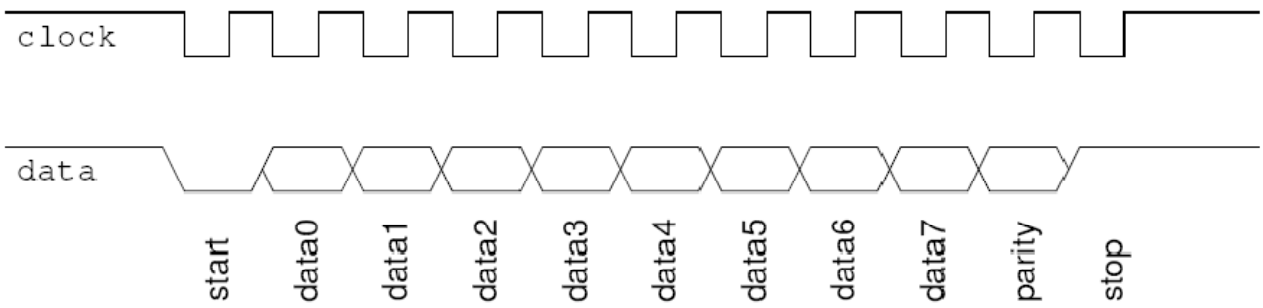


Alla fine di questo frame l'*host* si aspetta un *acknowledge bit* da parte della tastiera (nel caso il dato sia stato correttamente interpretato), quindi rilascia la *linea dati*. Se l'*host* non rilascia la *linea dati* dopo l'undicesimo ciclo di clock la tastiera capisce che c'è stato un errore nella comunicazione e continua a generare il *clock* finché la linea dati non viene rilasciata.

TASTIERA → XSA-50

Quando la tastiera vuole spedire un dato, come prima cosa controlla la *linea di clock* per assicurarsi che sia in uno stato logico alto. Se non lo è significa che l'*host* sta inibendo la comunicazione e la tastiera deve *bufferizzare* i dati da spedire finché la *linea di clock* non viene rilasciata per almeno 50 µs, quindi può spedire i suoi dati.

Per trasmettere i dati la tastiera usa lo stesso protocollo a undici bit viste nella sezione precedente.



I dati che vengono trasmessi dalla tastiera corrispondono al codice dei tasti che vengono premuti, viceversa i dati spediti dalla scheda corrispondono ad alcuni comandi di controllo per la tastiera.

L'INTERFACCIA

La tastiera consiste in una matrice di tasti ciascuno dei quali viene monitorizzato da un processore interno alla tastiera. Ogni tastiera possiede un processore specifico (diverso dagli altri in base alle caratteristiche della tastiera), ma tutti quanti i processori fanno sostanzialmente lo stesso lavoro: controllano quale tasto viene *premuto*/*rilasciato* e spediscono all'*host* il dato appropriato; all'occorrenza si occupano di *bufferizzare* i dati se la linea di comunicazione risulta occupata.

SCAN CODES

Come abbiamo già visto, il processore interno alla tastiera spende la gran parte del suo tempo monitorando la matrice di tasti; quando vede che un tasto è stato *premuto*, *rilasciato* o *tenuto premuto*, si occupa di trasmettere all'*host* un pacchetto di informazioni chiamato *scan code*.

Esistono due tipi di *scan codes*:

- make codes: vengono spediti quando un tasto è premuto o viene tenuto premuto;
- break codes: utilizzati quando un tasto è rilasciato.

Ogni tasto è associato ad un unica coppia < *make code* / *break code* >, in tal modo l'*host* può determinare esattamente cos'è successo e a che tasto semplicemente guardando il singolo *scan code*. Il set di *make* e *break code* per ogni tasto forma quello che viene chiamato *scan code set*. Esistono tre *scan code sets* standard chiamati *uno*, *due* e *tre*: tra di loro si differenziano sostanzialmente nella *codifica* usata per identificare i tasti della tastiera e i comandi inviati dall'*host*; quale di questi standard usare dipende da particolari d'implementazione. È importante avere sempre presente che i codici inviati identificano univocamente un tasto sulla tastiera e non un carattere: ovvero non è definita alcuna relazione tra *scan code* e codifica *ASCII* (tradurre *scan codes* in codici *ASCII* è compito dell'*host*).

La tastiera inoltre supporta un'altra funzionalità: il *typematic*; quando un tasto viene tenuto premuto, la tastiera invia un *make code* seguito da un treno di *make code* caratterizzati da un *delay* iniziale e da una velocità di ripetizione entrambi programmabili.

RESET

Al momento dell'accensione la tastiera esegue un auto-test diagnostico denominato *BAT* (*Basic Assurance Test*) e carica i seguenti valori di default:

- *Typematic*: *delay* iniziale = 500 ms;
- Typematic: velocità ripetizioni = 10,9 Hz;
- *Scan code set*: 2;
- Abilitati i *make code*, i *break code* e il *typematic*.

Al momento di iniziare il test la tastiera accende i suoi *LED* e li spegne una volta terminatolo. A questo punto la tastiera spedisce all'*host* un codice di successo (0*xAA*) o di errore (0*xFC*). Questo codice deve essere spedito 500 ÷ 750 ms dopo l'accensione.

COMMAND SET

Si chiama *command set* l'insieme dei comandi di controllo che *host* e tastiera possono scambiarsi. In questa sede verranno presentati solo i comandi utilizzati nel presente progetto (per una trattazione completa dell'argomento si rimanda a http://panda.cs.ndsu.nodak.edu/%7Eachapwes/PICmicro/keyboard/atkeyboard.html).

Sotto vengono riportati alcuni comandi che l'*host* può mandare alla tastiera e la reazione della tastiera a tali comandi:

- 0x*FF* (*Reset*) ⟶ La tastiera risponde con un *ack* (0x*FA*), quindi entra in modalità *reset*.
- 0x*FE* (*Resend*) ⟶ La tastiera risponde spedendo nuovamente l'ultimo byte.
- 0x*F0* (*Set Scan Code Set*) ⟶ La tastiera risponde con un *ack*, quindi legge il dato successivo che l'*host* le invia, che può essere:
  - 0x01 ⟶ *Scan Code Set 1*;
  - 0x02 ⟶ *Scan Code Set 2*;
  - 0x03 ⟶ *Scan Code Set 3*.

La tastiera quindi risponde con un secondo *ack* e carica il relativo *Scan Code Set*.

- 0x*F3* (*Set Typematic Rate*/*Delay*) ⟶ Serve per settare i due parametri del *Typematic*; la tastiera aspetta dall'host i dati successivi che definiscono delay e velocità delle ripetizioni;
- 0x*F6* (*Set Default*) ⟶ La tastiera carica i valori di default.
- I prossimi comandi possono venir spediti alla tastiera in ogni momento ma in uenzano il suo reale comportamento solo se la tastiera è in *Scan Code Set* 3:
  - 0x*F9* (*Set All Keys Make*) ⟶ La tastiera risponde con un *ack* e disabilita per tutti i tasti i break code e il typematic;
  - 0x*F8* (*Set All Keys Make*/*Break*) ⟶ Simile al precedente, con la differenza che disabilita solo il typematic;
  - 0x*F7* (*Set All Keys Typematic*) ⟶ Come il precedente, solo che disabilita il break code.

In particolare in questo progetto viene usato lo *Scan Code Set* 3, con l'opzione 0x*F9* (vengono disabilitati il *break code* e il *typematic*).