Steve Blackbird (TardoFreak)

SWITCH CONTROL UNIT

15 December 2010

La Switch Control Unit

A volte è utile avere un circuito semplice che sia in grado di operare come **interfaccia hardware** per un PC o cumunque essere comandato tramite la linea seriale **RS232**C.

Sin da quando ero ragazzo ho sempre voluto realizzare un circuito come questo perché ne vedevo l' utilità. I PC sono macchine potentissime ma sono macchine da ufficio. Non sono previsti ingressi e uscite digitali se non per pilotare stampanti o per interfacce di comunicazione. La vecchia interfaccia per stampate è stata così utilizzata in tutti i modi possibili immaginabili ma è poco, troppo poco a mio avviso.

L' apparecchio descritto in questo articolo è una **s**witch **c**ontrol **u**nit (**SCU**) costruita intorno ad un microcontrollore **PIC16F877** che serve per **comandare uscite digitali**, relè, circuiti integrati o **leggere gli stati degli ingressi**, il tutto comandato da un host connesso tramite l' interfaccia seriale.

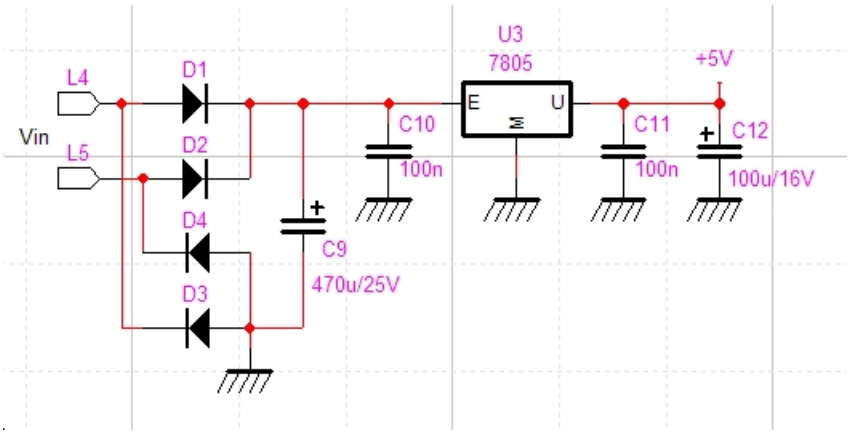
E' possibile, tramite questa unità di controllo, trasformare il PC, ad esempio, in un controllore programmabile oppure avere un comodo circuito per interfacciare integrati digitali, come ausilio per le prove, come componente per la realizzazione di attrezzature di collaudo, per comandare ed interfacciarsi con convertitori A/D e D/A e superare quindi le limitazioni d' interfacciamento che i PC hanno.

Il microcontrollore utilizzato è sicuramente un po' vecchiotto e limitato ma la sua grande diffusione, un **costo più che abbordabile** e il contenitore DIL a 40 pin lo rendono una scelta felice per l' applicazione hobbistica. Ho usato un vecchio PIC16F877/04P perché ne ho molti inutilizzati, ma il programma è così breve che si può implementare in un micro con meno di 1KByte di memoria flash, quindi praticamente su tutti.

Il circuito

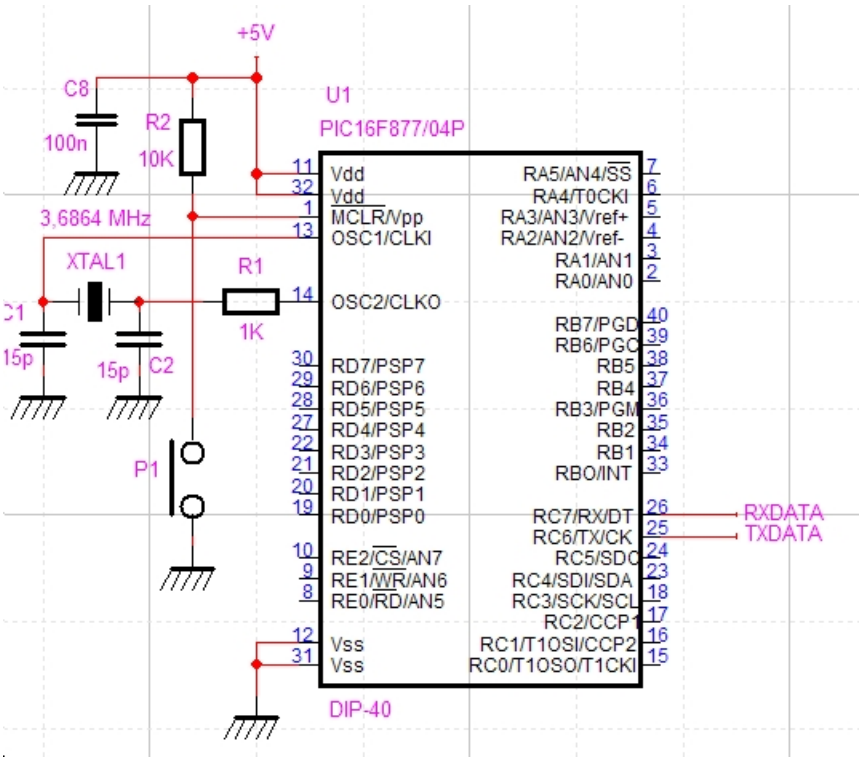
Lo schema circuitale dell' apparecchiatura è molto semplice; si tratta infatti del solo microcontrollore e di un ST202, un circuito integrato d' interfaccia per i livelli RS232. Nel caso si dovesse comandare questo circuito tramite un altro microcontrollore oppure collegato ad un adattatore USB/seriale, il circuito d' interfaccia realizzato

con l' ST202 può essere eliminato. Volendo si potrebbe utilizzare il circuito come periferica intelligente collegata ad un altro PIC ma l' applicazione per cui e' stato pensato è come unità slave per un PC. Tutti i pin delle porte sono programmabili per essere utilizzate come ingresso o uscita. La velocità di comunicazione è fissa a 19200 baud ma è facilmente modificabile all' interno del sorgente del firmware. E' sufficiente e necessario avere un programma di emulazione di terminale come l' hiper-terminal di Windows, o un qualsiasi programma di terminale su un computer qualsiasi ,o addiritura un terminale nudo e crudo come il VT50 per gestire il tutto. La semplicita' di utilizzo e di gestione è, almeno nelle intenzioni, l' aspetto principale.



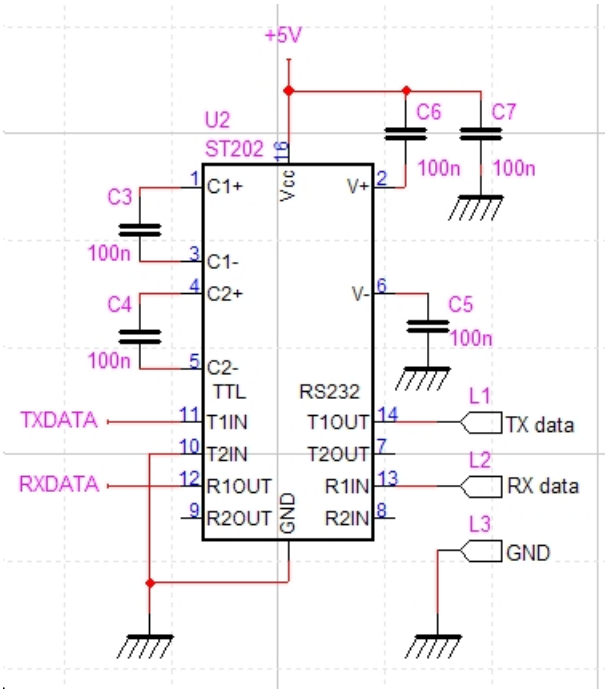
SCU_ps.jpg

Come dicevo prima il circuito e' molto semplice. In questa prima immagine vediamo l' alimentatore. Non è strettamente necessario se si dispone di un alimentatore a 5V ma puo' essere utile nel caso si volesse realizzare un'apparecchiatura stand-alone alimentata autonomamente. Il 7805 e' in grado di erogare una corrente di 500 mA, piu' che sufficiente ad alimentare il microcontrollore ed eventuali circuiti d'interfaccia come relè o ingressi e/o uscite optoisolati.



SCU_pic.jpg

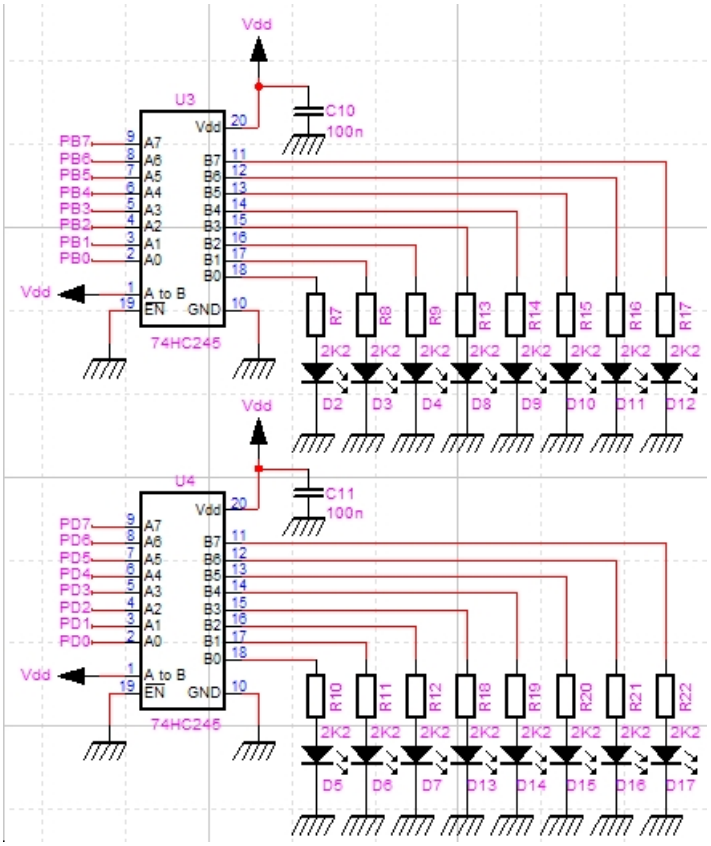
Il cuore dell'apparecchiatura è costituito dal solo PIC16F877. Come possiamo vedere il clock è ottenuto da un quarzo da 3,6864 MHz. La scelta di questa frequenza non è casuale, infatti un oscillatore a questa frequenza permette di utilizzare micro con limite di frequenza a 4 MHz ed è una frequenza studiata per ottenere le velocità di comunicazione standard con precisione assoluta. Il piedino MCLR e' collegato alla Vcc tramite la solita resistenza da 10k. Ho anche collegato un pulsante di RESET, molto utile in fase di progettazione del firmware. I due segnale RXDATA e TXDATA sono il canale seriale mediante il quale il micro comunica con il PC attraverso un circuito d' interfaccia RS232 (visibile nell' illustrazione seguente) oppure attraverso un integrato che funga da interfaccia USB/seriale come il MCP2200 della Microchip.



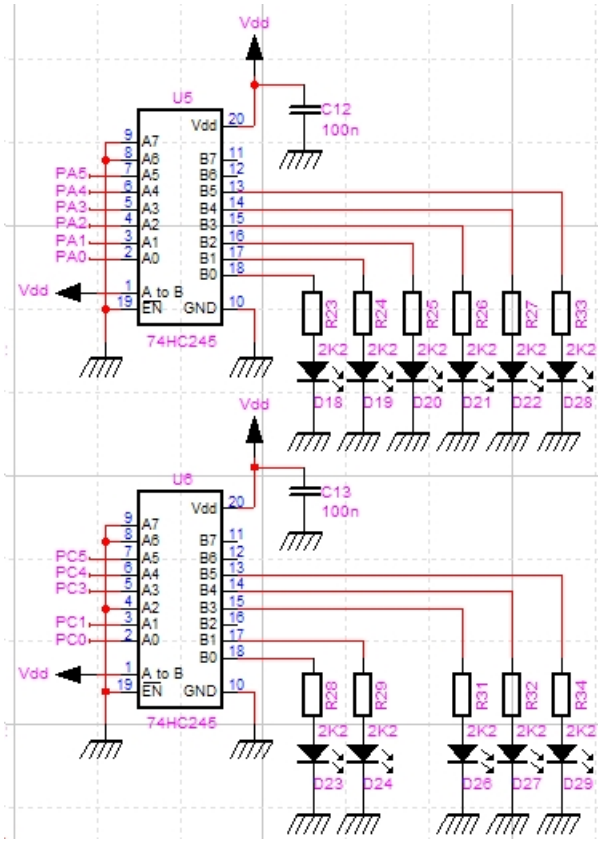
SCU_RS232.jpg

Esempio di realizzazione

Nel prototipo che ho realizzato ho montato buffers che pilotano i LEDs, uno per ogni linea del microcontrollore.

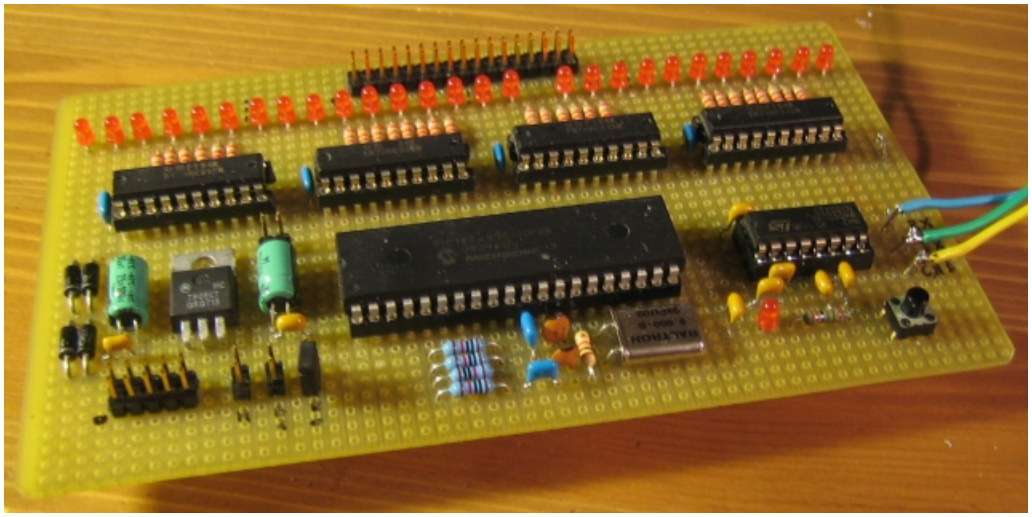


SCUleds1.jpg



SCUleds2.jpg

Come si può notare questo circuito non è niente di speciale. Quattro 74HC245 pilotano il LEDs in modo da poter facilmente visualizzare lo stato di ogni pin. L' ingresso di abilitazione (EN) e quello di direzione (A to B) sono forzati agli stati opportuni per poter funzionare con i LEDs collegati secondo lo schema. Questo circuito di visualizzazione e' particolarmente utile nel caso si voglia realizzare una SCU per essere strumento di laboratorio, un' apparecchiatura di ausilio per lo studio e la prototipazione di circuiti digitali. Avere sott' occhio lo stato dei segnali è quantomai utile in fase di sviluppo.



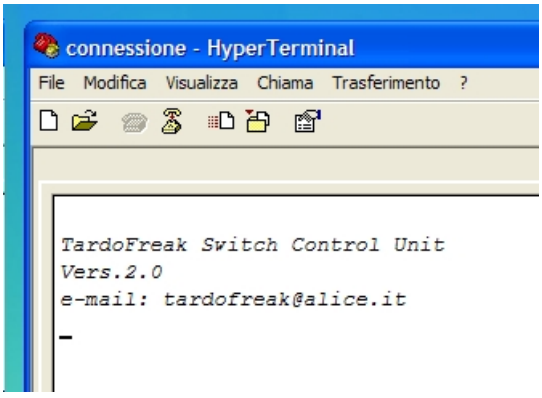
SCU2.jpg

Come si può notare in basso a destra ho anche collegato un pulsantino di RESET fra l' ingresso MCLR del PIC e la massa, indispensabile in fase di sviluppo del software e che può essere comunque utile. Nella parte superiore della millefori ho riportato le linee di I/O insieme alla massa ed al +5V in modo da avere tutto a disposizione per applicazioni e/o sviluppi futuri.

Come iniziare

Supponiamo di aver realizzato il circuito minimo, di aver programmato il PIC e di volere verificarne il buon funzionamento. Colleghiamo quindi il tutto con una porta seriale del PC, avviamo il programma di emulazione di terminale (HyperTerminal o altro) e settiamo i parametri di comunicazione: 19200 baud 8 bit di dati, 1 bit di Stop, nessuna parita', nessun controllo di flusso hardware, nessun controllo Xon/Xoff ... il minimo indispensabile insomma!

Diamo tensione al circuito. Sul terminale deve comparire il messaggio di inizio del fimware interno.



SCU_term.jpg

Se non compare verifichiamo che dal piedino RC6/TX (pin 25), appena data alimentazione al circuito, ci sia un segnale. Se non c'è segnale verifichiamo l' oscillatore visualizzando il segnale al pin 14. Verifichiamo anche che i segnali di TX e RX siano collegati ai giusti pin della porta seriale.

Il circuito DEVE funzionare al primo colpo vista la semplicità quindi se ci sono problemi possono solo derivare dal clock (quarzo e condensatori) o dall' interfaccia. Con un oscilloscopio tutto si risolve in pochi minuti.

Il messaggio iniziale si può tranquillamente modificare. Io ci ho messo il mio nickname e la mia e-mail perche' ... ih ih ih ... sono un megalomane e mi farebbe piacere ricevere una mail da qualcuno che l' ha usato ed è rimasto soddisfatto, ma potete eliminarlo se vi dà fastidio.

Dicamo che potete fare quello che volete perché **il firmware e' fornito "as is" (nello stato in cui è) e non mi assumo nessuna responsabilità per quello che può derivare dal suo utilizzo.**

Per verificare che riceva i dati dal terminale premete <INVIO> dalla tastiera e la SCU vi risponderà con un rassicurante "OK".

I comandi

Per ogni linea di I/O bisogna avere la possibilità di definirne la funzione (ingresso o uscita), di leggere il suo stato e, nel caso di una linea di uscita, di scrivere un 0 o un 1 logico. Questo è possibile con soli 5 comandi.

Configurazione come linea di input

Sintassi:
SI(p)(n)
Definisce il bit (n) della porta (p) come linea di input. Per esempio il comando:
SIB3
definisce la RB3 del micro come input.
Restituisce come risposta un "OK" seguito da i caratteri <cr>e <lf>

Configurazione come linea di output

Sintassi:
SO(p)(n)
Definisce il bit (n) della porta (p) come linea di input. Per esempio il comando:
SOE0

definisce la RE0 del micro come output.
Restituisce come risposta un "OK" seguito da i caratteri <cr>e <lf>

Srittura nella linea di output

Per scrivere un "1" od uno "0" in una linea di output si usano rispettivamente questi due comandi: Sintassi:

W1(p)(n)

Mette il bit (n) della porta (p) a livello logico "1". Per esempio il comando:
W1E0
Farà andare a livello "1" (Vdd) la linea RE0.

Sintassi:

W0(p)(n)

Mette il bit (n) della porta (p) a livello logico "0". Per esempio il comando:
W0E0
Farà andare a livello "0" (0V) la linea RE0.

Questi due comandi restituiscono come risposta un "OK" seguito da i caratteri <cr>e <lf>

Lettura linea

Sintassi:

RD(p)(n)

Legge lo stato del bit (n) della porta (p). Restituisce "1" o "0" ASCII. Per esempio il comando:
RDC5
Restituisce come risposta il ivello logico della linea ("0" o "1") seguito dai caratteri <cr><lf>.
Se questa è stata configurata come linea di output darà comunque il valore del livello logico della linea.

Messaggi di errore

Abbiamo visto che se i comandi impartiti sono andati a buon fine la SCU restituisce una risposta "OK". Se qualcosa è andato storto può restituire uno di questi messaggi di errore:

E1

Indica un **errore di sintassi** del comando. Il comando non è stato riconosciuto come comando valido.

E2

Indica un **errore sul parametro**. Ad esempio se si cerca di settare come uscita il bit

9 (che non esiste) di una porta, oppure i bit 6 o 7 della porta C (utilizzati dall' UART) la SCU ritornerà questo codice di errore. Il firmware si preoccupa solo di assicurare l' impossibilità di scrivere in queste due ultime linee (RA6 e RA7) e che l' indicatore del bit di linea sia nel range 0-7.

E3

Indica un **errore di porta**. Se si sbaglia la lettera della porta (ad esempio specifichiamo la porta H che non esiste) verrà generato questo errore

Note

Solitamente, nelle applicazioni che fanno uso di linee seriali, è consuetudine inserire un campo di checksum per la verifica dell' integrità dei dati o dei comandi. In questa applicazione non l' ho inserito. Ho preferito fare in modo che la SCU fornisse l' echo dei caratteri inviati. Basta verificare che l' echo ricevuto dopo aver inviato il carattere sia corretto per essere sicuri che questi sia stato ricevuto correttamente. C'è anche un altro motivo per cui non ho inserito il checksum: probabilmente questo circuito verrà abbinato ad un convertitore USB/RS232 montato vicino al PIC ed a lui collegato direttamente con due piste e, data l' elevata precisione del baud rate (ottenuto tramite un quarzo apposta da 3,6864MHz) non penso sia il caso di verificare l' integrità perché l' USB comprende già il controllo d' integrità e qualche centimetro di filo per il collegamento fra l' interfaccia ed il PIC non può distrubare il segnale in modo preoccupante (a meno che non viviate sotto un ripetitore televisivo). Un' altra cosa da tenere presente è che il programma non è case-sensitive. Questo vuol dire che le lettere possono essere inviate sia maiuscole che minuscole ma l' **echo ritornerà sempre lettere maiuscole**. L' echo è facilmente eliminabile dal firmware ed il checksum si può implementare. A voi la scelta.

Il firmware

Il firmware è scritto in C con il MikroC. E' talmente corto che si può benissimo compilare e modificare con la versione di valutazione. E' necessario impostare nei dati di progetto la frequenza di clock a 3,6864 Mhz altrimenti la velocita' di comunicazione risulterà errata. potete anche usare un PIC diverso, più veloce e con un altro quarzo. Il povero PIC che ho usato ha i suoi anni, è un avanzo di un prodotto di tanti anni fa e la scelta del quarzo era obbligata se volevo ottenere precisione e velocità di comunicazione. Diciamo che fa poco ma aggiungo che **quel poco che fa lo fa molto bene.**

Download

Il file lo si può scaricare da **qui**

Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Tardofreak:scu"