Flavio Ansovini (ian27177)

PIC 18F4682 -> VGA

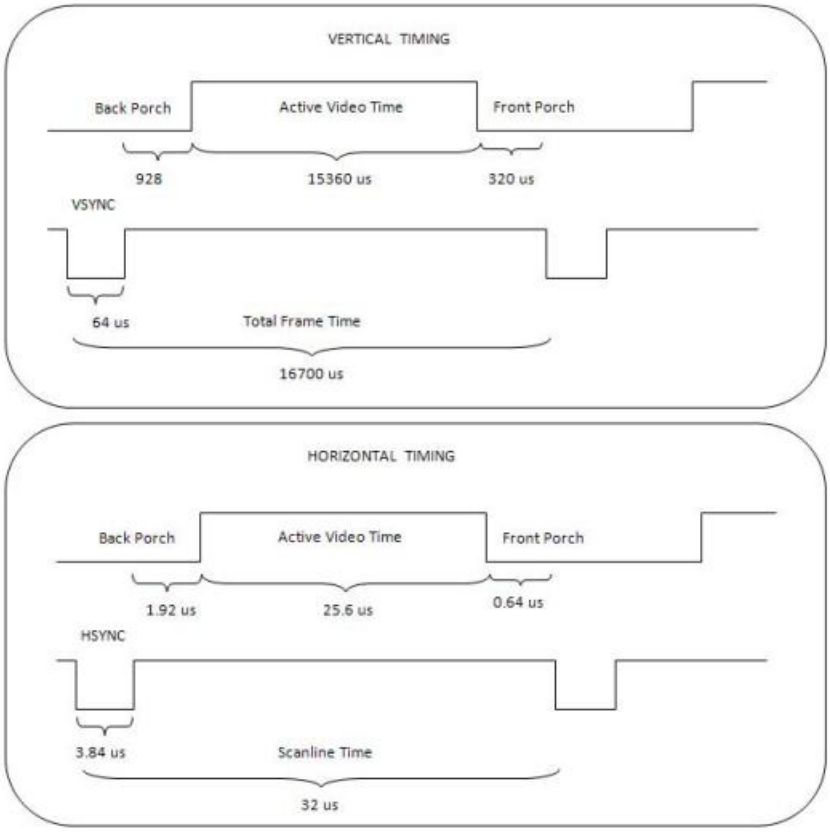
19 September 2008

Questo progetto nasce dalla curiosità di verificare se sia possibile visualizzare un' immagine a colori su un qualunque monitor VGA con il solo utilizzo di un microcontrollore Pic. In effetti il progetto è riuscito, anche se con i limiti propri dell'hardware utilizzato, e ovviamente ho avuto grande soddisfazione quando, dopo ore ed ore perse davanti al programmatore sono riuscito ad visualizzare sul monitor la scritta colorata "Ciao Mondo!!". Ovviamente le "capacità grafiche" sono molto limitate, soprattutto dalla memoria flash a disposizione (80Kbyte nel caso dell'MCU scelta) e dalla massima frequenza di clock supportata dal controllore. Prima di passare alla trattazione delle scelte hardware e software da me attuate vorrei descrivere con semplicità lo standard VGA. Vi risparmio il pistolotto sulla storia dello standard, che comunque potete trovare su wiki al seguente link: http://it.wikipedia.org/wiki/Video_Graphics_Array

Per iniziare bisogna scegliere la modalità grafica con cui lavorare, quella da me scelta è 640x480x16(colori) 60Hz , anche se in realtà il risultato ottenuto è molto diverso, si tratta di 60x85x8(colori) 60Hz ed in seguito ne spiegerò il motivo. Lo standard VGA prevede che vengano "colorati" i pixel dello schermo partendo dal primo in alto a sx fino all'ultimo in basso a dx, colorando riga per riga da sx a dx. I segnali elettrici interessanti per la realizzazione del progetto sono i seguenti: VSYNC pin n° 14 (sincronismo verticale), HSYNC pin n° 13 (sincronismo orizzontale), R pin n° 1 (rosso), G pin n° 2 (verde), B pin n° 3 (blu). La "comunicazione" monodirezionale tra la sorgente e il monitor avviene in questo modo:

- invio del sincronismo verticale VSYNC (viene inviato all'inizio di ogni schermata quindi ogni 16.66ms -> freq. 60Hz)
- tempo di latenza chiamato Back Porch (da qui in avanti vengono colorati i pixel orizzontali di ogni riga)
- invio del sincronismo orizzontale HSYNC (viene inviato all'inizio di ogni riga)
- tempo di latenza Back Porch
- Active Video Time, viene letta la combinazione dei colori RGB per il tempo dedicato ad ogni pixel della riga (tempo di 1pixel = AVT/640).
- tempo di latenza Front Porch ....si ripetono gli ultimi 4 passi per ognuna delle 480 righe.
- tempo di latenza Front PorchA questo punto la schermata è stata colorata interamente, quanto appena detto viene ripetuto con una frequenza pari a quella refresh (60 Hz).

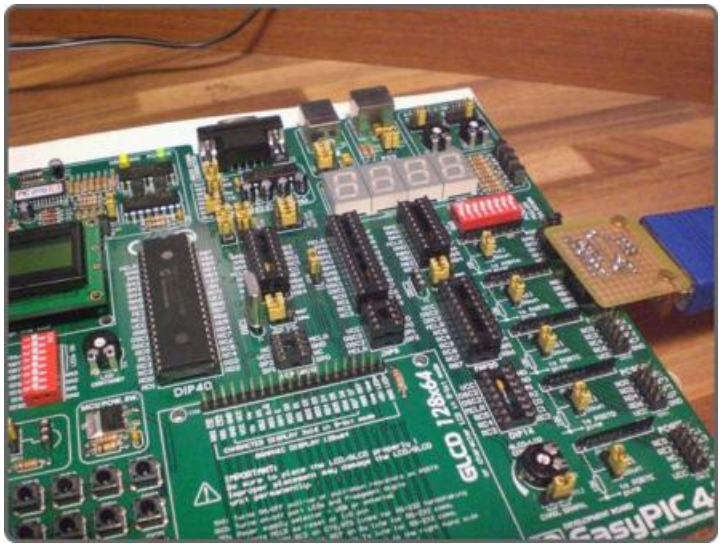
Affinché tutto funzioni è importante che i tempi dei segnali di sincronismo vengano rispettati. Ogni monitor è in grado di cambiare modalità grafica in automatico cercando di capire dai segnali di sincronismo in quale modalità viene trasmessa l'immagine. Va da sè che se i tempi non sono precisi il monitor continuerà a cambiare modalità grafica, se invece i sincronismi inviati sono addirittura lontani da ogni modalità accettata, allora il monitor andrà in stand-by con schermo nero oppure il classico messaggio "no signal...". I tempi previsti per la modalità da me adottata sono riportati nel disegno esplicativo seguente mostra come avviene la creazione dell'immagine sul monitor.



VGA Timing.JPG

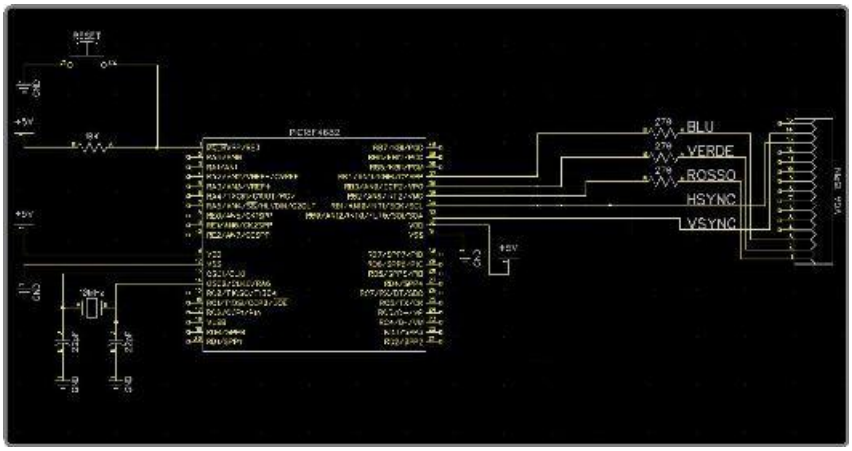
Tuttavia anche qui come in altri casi vale la regola "l'unico standard è l'assenza di standard!" , infatti in rete è possibile trovare un'infinità di documenti forniti da diversi produttori di monitor che prevedono tempi di latenza spesso diversi tra loro. Per quanto riguarda i colori, si può dire che il risultato ottenuto dipende da come si desidera miscelare i tre colori fondamentali rosso, verde e blu. Per questo progetto ho deciso di utilizzare solo 8 combinazioni, non è infatti prevista la variazione dell'intensità di ognuno dei singoli colori di base ma possono solo essere "ON" oppure "OFF". In realtà non sarebbe difficile aumentare il numero dei colori fino a 64 , ma costituirebbe un ulteriore carico per la MCU e non lo ho ritenuto necessario e

nemmeno troppo interessante. Per essere più chiaro , mi riferisco all'uso di 6 uscite digitali invece di tre (due per il rosso, due per il verde e due per il blu), usando due partitori di tensione per ogni uscita si potrebbero avere due diverse intensità per ogni colore di base e quindi arrivare ad un totale di 64 colori. Per provare il firmware ho utilizzato una demoboard in mio possesso, si tratta della EasyPic4 prodotta dalla ditta Mikroelecktronika http://www.mikroe.com/ .



CIMG1127_2.JPG

Questa scheda è molto comoda perché funziona come un'interfaccia tra i vari socket disponibili per i diversi controllori e varie periferiche quali LCD, Switch, LED, Display 7 segmenti etc. etc. Il che rende molto rapida la prototipazione. Nel caso si voglia realizzare il progetto descritto su un pcb dedicato, i collegamenti sono comunque molto semplici, non ho utilizzato altro che un micro PIC18F4682, un quarzo da 10MHz con i suoi due condensatori da 22pF al polyestere , due resistenze per i segnali di sincronismo e tre per i colori. Come uscite digitali ho utilizzato quelle della PORTB. Il tutto è cablato secondo il seguente schema elettrico:



Schematic.JPG

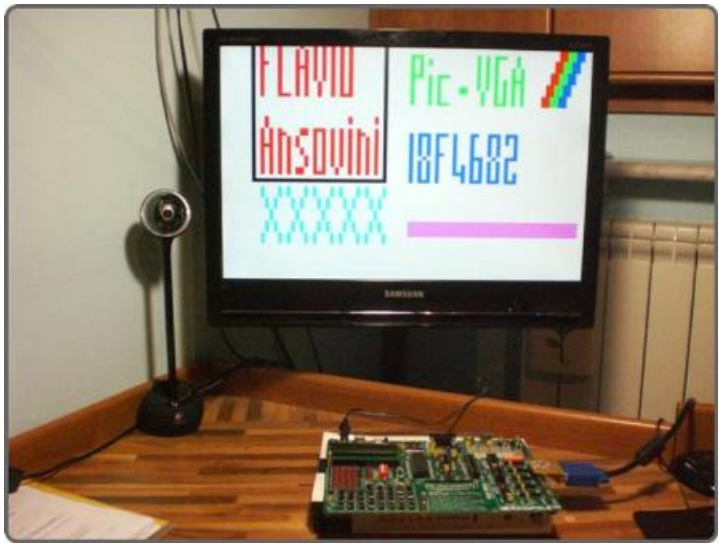
Come si può notare dallo schema è sempre bene inserire delle resistenze di Pull-Down da 10Kohm tra ogni pin di uscita del PIC e GND, questo per assicurarsi che al cessare di ogni impulso positivo il segnale torni a zero nel minor tempo possibile. Il motivo per cui ho scelto un quarzo da 10MHz è che il PIC18F4682 permette di impostare un moltiplicatore interno X4 ma solo con i quarzi da 10MHz, questo permette di raggiungere una frequenza di clock pari a 40MHz, se avessi scelto un quarzo da 20MHz non avrei potuto usare il moltiplicatore. In questa applicazione è molto importante considerare i tempi necessari all'esecuzione delle varie routine a causa della rapidità con cui le funzioni devono venire eseguite. Lavorando con una frequenza di clock pari a 40MHz, ogni singola istruzione elementare in assembler ha una durata di 0.1us (1/(FOSC/4)=0.1us), è quindi abbastanza chiaro che non è possibile introdurre ritardi precisi, nemmeno alla prima cifra decimale, ma grazie forse alla mancanza di uno standard "forte" i monitor dopo aver agganciato il sincronismo riescono ad adattarsi, entro certi limiti... Bisogna comunque considerare che anche un errore di 50ns durante l'horizontal timing, che viene quindi reiterato per il numero delle righe (480) , diventa un errore di 24us per ogni ciclo di refresh, il che non è da sottovalutare.

Il firmware riproduce lo schema relativo ai tempi riportato in precedenza: porta a +5V il VSYNC (PORTB.0),fa trascorrere il tempo di latenza "Back Porch" e poi inizia il ciclo di ogni riga: porta a livello logico 1 (+5V) il segnale HSYNC (PORTB.1), attende il trascorrere del "Back Porch" orizzontale e quindi il monitor inizia a leggere lo stato dei segnali relativi ai tre colori di base. Questa lettura avviene durante "l'Active Video Time", tempo nel quale il monito acquisisce 640 combinazioni di colori, che corrispondono poi al colore di ognuno dei 640 pixel della prima riga. L'Active Video Time dura 25.6us, quindi il firmware dovrebbe leggere il colore che si vuole attribuire ad ogni bit e settare le tre uscite RGB (PORTB.2, PORTB.3, PORTB.4) in 0.04us (25.6/640=0.04us), che nel nostro caso non è possibile essendo necessaria l’esecuzione di più istruzioni per la lettura del byte "colore" del pixel e scrittura sulla PORTB del PIC. Questo è il motivo per cui sono stato costretto a ridurre drasticamente la risoluzione a 60 righe per 85 colonne, in pratica la larghezza di ogni pixel equivale a 7 pixel adiacenti e l'altezza a 10, il monitor legge 640 pixel per riga ma in realtà il colore cambia al meno ogni 7 pixel adiacenti.

Al termine dell'Active Video Time viene atteso il trascorrere del Front Porch orizzontale e poi viene inviato un nuovo sincronismo orizzontale e l’inizio di un’altro ciclo “orizzontale”. Tutto questo viene reiterato per 480 volte affinchè vengano colorate tutte le righe. Per evitare di dover memorizzare troppi dati e anche di avere un a risoluzione di 85X480 che avrebbe poco senso ho fatto in modo di ripetere ogni riga 8 volte, riducendo i dati relativi ai colori dei pixel e il numero di righe effettive. Al termine della routine , quando lo schermo è stato interamente colorato

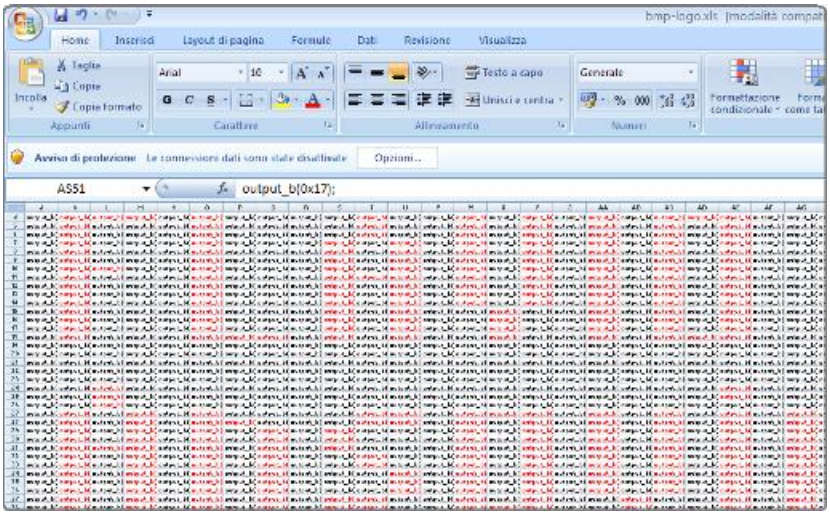
, viene atteso il Front Porch Verticale e poi il programma riprende con un'altra schermata. Affinchè la frequenza di refresh venga rispettata è necessario che il programma venga eseguito in un tempo di 16,66ms , che equivale ad una frequenza di 60Hz. Mantenere questo valore è molto importante e se si pensa all'errore di 0,5ns preso come esempio in precendenza, il valore già moltiplicato per 480 si andrebbe a moltiplicare ancora per 60, ritardo che influisce sulla frequenza di refresh abbassandola fino a circa 55Hz (non supportata dal monitor).

Il risultato finale del mio lavoro si può osservare nell'immagine sottostante.



CIMG1126_2.JPG

Per realizzare l'immagine mi sono servito di un file excel nel quale ho delimitato un'area di 60X85 caselle. In un'altra pagina ho invece creato una tabella con i codici esadecimali degli otto colori possibili e li ho colorati ognuno con il suo colore. Con copy &paste del colore interessato riesco quindi a scrivere o disegnare forme avendo già un'idea di quale sarà il risultato



excel.jpg

I file relativi al progetto sono disponibili in download al seguente link:

http://www.electroportal.net/users/files/PIC16f4682VGA-OUT.rar

Spero che l'articolo sia stato di interesse, chiunque volesse fare correzioni, domande o miglioramenti a questo articolo può contattarmi senza esitare.

Flavio Ansovini

Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Ian27177:articolo1"