UNIVERSITÀ DEGLI STUDI DI PADOVA

SEDE DECENTRATA DI VICENZA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA TRIENNALE IN

INGEGNERIA MECCATRONICA

ELABORATO FINALE

INTERFACCIAMENTO DI UN TOUCH SCREEN AD UN MODULO GSM PER APPLICAZIONI DOMOTICHE

INTERFACING A TOUCH SCREEN AND A MODULE GSM FOR DOMOTIC APPLICATIONS

Relatore per l'Università: Prof. Ing. Roberto OBOE
Correlatore: Carlo FONGARO

Diplomando: Enrico MUNARON

ANNO ACCADEMICO 2010/2011

INTRODUZIONE

Nel presente elaborato viene descritto lo stage svolto presso l'azienda EAS Elettronica S.p.a., una delle principali ditte in Europa che opera nel settore delle schede elettroniche per uso industriale.

Negli ultimi tempi l'impresa, sollecitata da alcuni clienti importanti, ha rivolto attenzioni al mondo della domotica.
Nonostante questo ramo dell'automazione abbia già raggiunto livelli avanzati anche in Italia, si è deciso di dar vita ad un progetto molto avvincente in quanto nato e sviluppato totalmente all'interno dell'azienda.

L'obiettivo del tirocinio è utilizzare un TFT della Sharp a colori completo di Touch Screen come tastiera elettronica che, abbinato ad un modulo GSM della Telit, sia in grado di eseguire le seguenti funzioni:

1) Scrivere ed inviare SMS a qualsiasi telefono GSM.
2) Ricevere messaggi da un telefono GSM e visualizzarli sullo schermo TFT.
3) Elaborare eventuali messaggi ricevuti allo scopo di attivare e disattivare attuatori di vario tipo.
4) Trasmettere ad un telefono GSM dati precedentemente elaborati dal modulo TFT (temperatura, pressione atmosferica, ecc).

La difficoltà del progetto consiste principalmente nel dover fare interagire tra loro i vari protocolli di comunicazione presenti nel sistema.

Si è quindi provveduto alla creazione di un software in grado di gestire senza ambiguità tutti gli eventi possibili.

INDICE GENERALE

CAPITOLO 1: PRESENTAZIONE DEL PROGETTO

1.1 Descrizione generale del sistema ..... pag. 4

CAPITOLO 2: L'HARDWARE

2.1 Lo schermo a cristalli liquidi ..... pag. 6

2.2 Il Touch Screen ..... pag. 10

2.3 Gestione del modulo TFT ..... pag. 15

2.4 L'FPGA (Field Programmable Gate Array) ..... pag. 17

2.5 La memoria RAM ..... pag. 19

2.6 La memoria Flash ..... pag. 22

2.7 Il microcontrollore ..... pag. 26

2.8 Il modulo Telit ..... pag. 29

2.9 L'alimentazione del sistema ..... pag. 31

CAPITOLO 3: IL SOFTWARE

3.1 L'ambiente di sviluppo CodeWarrior ..... pag. 39

3.2 La programmazione dell'FPGA in linguaggio VHDL ..... pag. 41

3.3 Come visualizzare un'immagine su TFT ..... pag. 43

3.4 Riconoscimento del tasto digitato ..... pag. 47

3.5 Scrittura di un carattere su display ..... pag. 48

3.6 Il modulo Telit: sintassi dei comandi principali del Protocollo at ..... pag. 51

3.7 Inserimento ed utilizzo del Protocollo at nel codice sorgente C ..... pag. 57

CAPITOLO 4: CONCLUSIONI

4.1 Obiettivi raggiunti ..... pag. 59

4.2 Documentazione dei risultati ottenuti ..... pag. 59

4.3 Obiettivi futuri ..... pag. 61

RINGRAZIAMENTI

CAPITOLO 1

PRESENTAZIONE DEL PROGETTO

1.1 Descrizione generale del sistema

Le componenti a disposizione per questa applicazione sono le seguenti:

- modulo Telit GSM;
- carta SIM;
- terminale di interfacciamento con il Telit;
- microcontrollore;
- modulo TFT LCD Touch Screen;
- Personal Computer;
- Linguaggio di programmazione C;
- connessioni varie.

Come già sottolineato, la difficoltà più rilevante dell'intero progetto è predisporre la corretta interazione tra ciascuna componente ed i vari protocolli di comunicazione presenti. Essi sono 8, e verranno descritti in seguito:

1) Protocollo di interfacciamento del TFT;
2) I2C (tra Touch Screen e microcontrollore);
3) Protocollo di comunicazione tra microcontrollore e FPGA;
4) Protocollo di salvataggio dati della memoria SDRAM;
5) SPI (tra memoria Flash e FPGA del modulo TFT);
6) RS232 (per la comunicazione tra telefono e microcontrollore);
7) Protocollo JTAG (per programmare la FPGA);
8) Protocollo dei comandi at del modulo Telit.

Per iniziare a comprendere il funzionamento del sistema globale, può essere utile la visione del seguente schema, raffigurante i blocchi fondamentali.

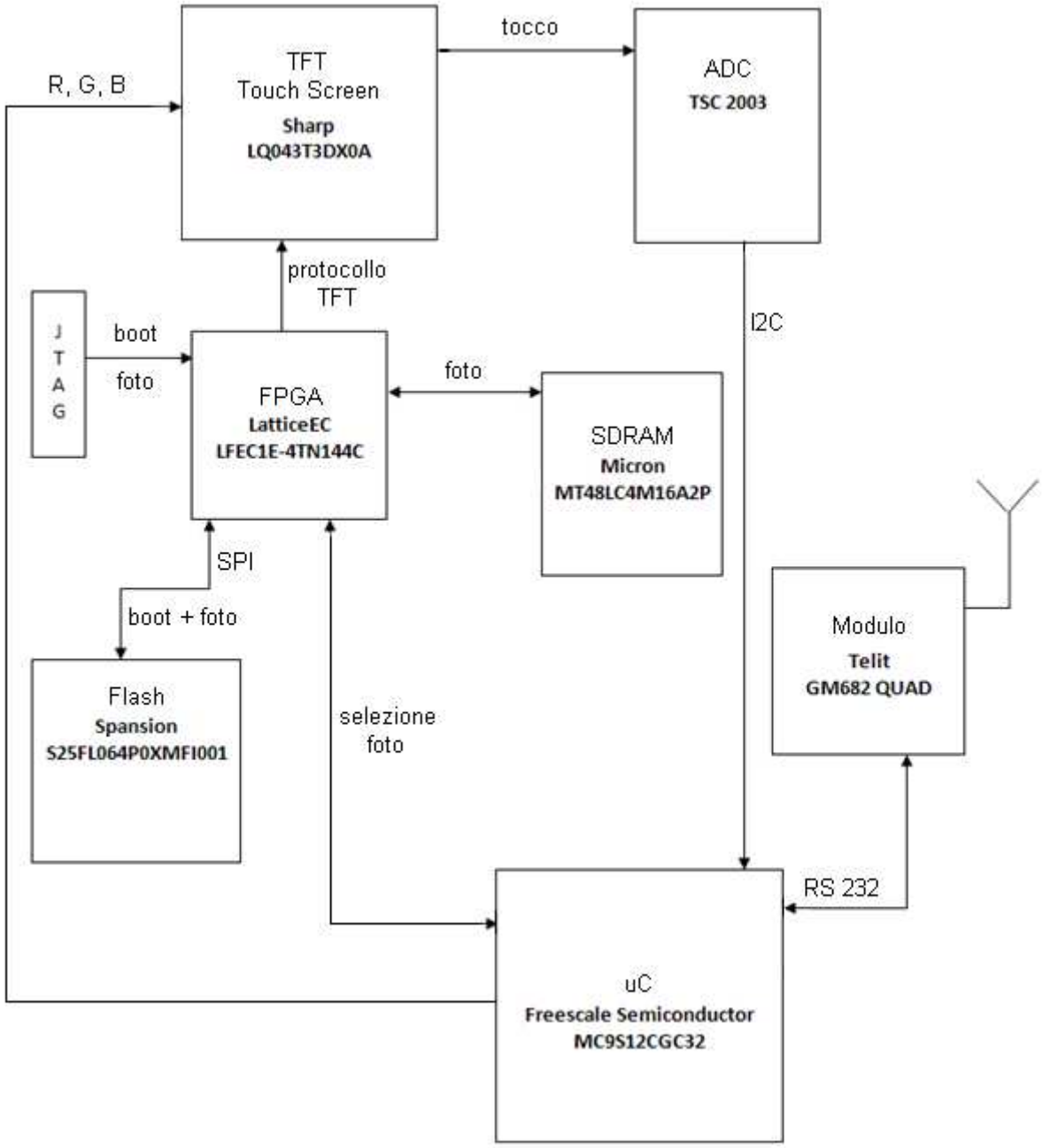


Fig. 1.1 – I blocchi fondamentali dell'intero sistema

Lo schermo a sfioramento rappresenta l'interfaccia tra l'utente ed il telefono cellulare.
Nel prossimo capitolo si analizzeranno i singoli dispositivi, iniziando dal display LCD.

CAPITOLO 2

L'HARDWARE

2.1 Lo schermo a cristalli liquidi

2.1.1 Introduzione

I primi display a cristalli liquidi vennero alla luce nel 1968 presso il Centro Ricerche "David Sarnoff" negli Stati Uniti.
Nel corso degli anni, la tecnologia si è evoluta notevolmente e gli LCD (Liquid Crystal Display) sono principalmente applicati nei seguenti settori:

- monitorìa;
- telefonìa cellulare;
- strumentazione elettronica;
- orologerìa.

Attualmente, sono disponibili sul mercato vari tipi di LCD. Tra essi, si ricordano i seguenti:

- **TN** (Twisted Nematic), i più semplici ed economici. Sono caratterizzati da consumi e tempi di risposta molto bassi. La qualità cromatica, però, è abbastanza limitata;
- **STN (**Super Twisted Nematic**),** un'evoluzione dei TN;
- **TFT** (Thin Film Transistor), distinti dai primi due per la loro qualità superiore. I costi e tempi di risposta sono più elevati rispetto alle altre categorie di LCD.

Lo schermo prescelto in questo progetto, per il suo buon rapporto qualità/prezzo, è il TFT Sharp **LQ043T3DX0A** da 4,3 pollici con risoluzione pari a 480*272 pixel.

2.1.2 Principio di funzionamento del display LCD

Il funzionamento di ogni LCD è basato sulle proprietà di particolari sostanze denominate cristalli liquidi. Tale liquido è intrappolato fra due superfici vetrose provviste di numerosi contatti elettrici con i quali poter applicare un campo elettrico al liquido contenuto. Ogni contatto elettrico comanda una piccola porzione del pannello identificabile come un pixel.
Sulle facce esterne dei pannelli vetrosi sono collocati due filtri polarizzatori disposti su assi perpendicolari tra loro (fig. 2.1). La particolarità naturale dei cristalli liquidi è torcere di 90° la polarizzazione della luce che arriva da uno dei due filtri, permettendole quindi di attraversare il secondo polarizzatore.

In assenza di campo elettrico, la luce può passare attraverso l'intera struttura e, trascurando la porzione di luce assorbita dai polarizzatori, l'apparecchio risulta trasparente. In presenza di un campo elettrico ortogonale ai piani delle due lastre di vetro, invece, le molecole del liquido si allineano parallelamente al campo, limitando la rotazione della luce entrante. Se i cristalli sono completamente allineati con il campo elettrico, la luce che vi passa attraverso è polarizzata perpendicolarmente al secondo polarizzatore e viene quindi bloccata del tutto facendo apparire il pixel non illuminato (fig. 2.1). Controllando la torsione dei cristalli liquidi in ogni pixel, tramite un terzo filtro, si è in grado di regolare quanta luce far passare. Si noti che in questo modo un pixel guasto apparirà sempre illuminato. Nella realtà alcune tipologie di pannelli funzionano all'opposto, ossia sono trasparenti se accesi ed opachi se spenti. In tal caso un pixel danneggiato risulta sempre opaco.

Un pixel di un LCD è costituito a sua volta da tre sotto-pixel dai colori rosso, verde e blu.
Un LCD non emette luce: ecco il motivo per cui questi schermi necessitano della retro-illuminazione (back light). La luce emessa da questo sistema attraversa il cristallo liquido e vi esce colorata dal sistema di filtraggio.
L'attivazione di ogni singolo punto del display viene comandata da un meccanismo di indicizzazione "rigaXcolonna", per cui ogni pixel si attiva quando passa corrente in entrambi gli elettrodi (anteriore e posteriore) che lo riguardano.
Nel display TFT i singoli pixel sono attivati da un apposito transistor. Quindi non è più necessario porre davanti al video una serie di elettrodi: è sufficiente la presenza di un'unica lastra trasparente da impiegare come GND.

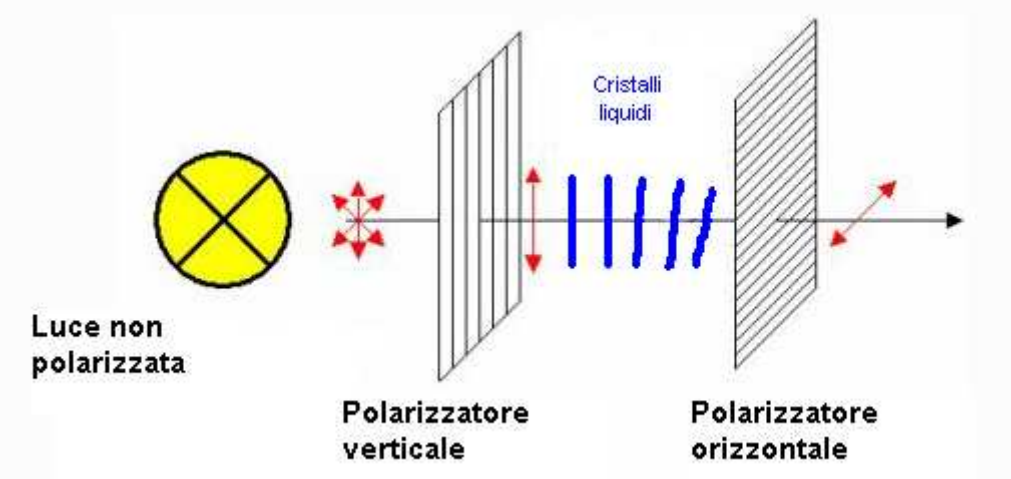


Fig. 2.1 – Principio di funzionamento del display LCD

2.1.3 Il protocollo di interfacciamento del TFT

Il funzionamento del protocollo di interfacciamento del TFT, ossia il processo tramite il quale un'immagine compare su display, è illustrato nella figura seguente e delucidato successivamente.

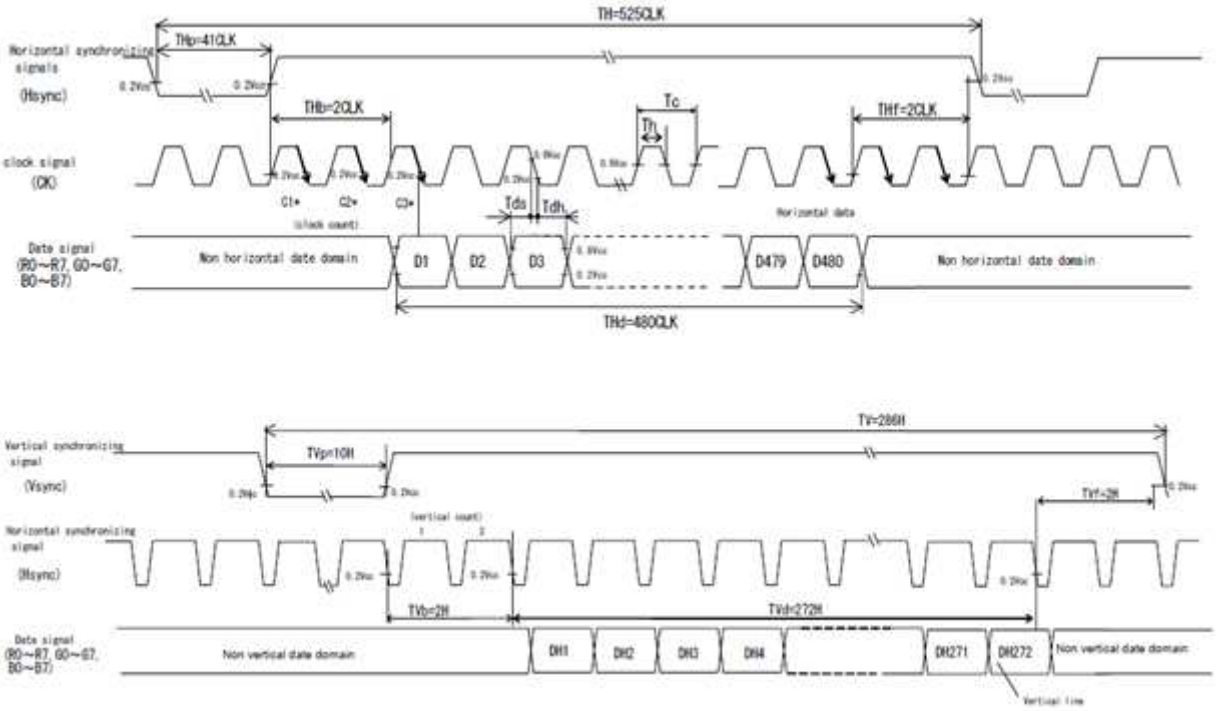


Fig. 2.2 – Protocollo di interfacciamento del TFT

Caratteristiche tecniche e tempistiche

Parameter		Symbol	Min.	Typ.	Max.	Unit
Clock	Frequency	1/Tc	7.83	9.00	9.26	MHz
	Duty ratio	Th/T	40	50	60	%
Data	Set up time	Tds	25	—	—	ns
	Hold time	Tdh	25	—	—	ns
Horizontal synchronizing	Period	TH	—	525	—	Clock
	Pulse width	THp	—	41	—	Clock
	Horizontal period	THd	—	480	—	Clock
	Back porch	THb	—	2	—	Clock
	Front porch	THf	—	2	—	Clock
Vertical synchronizing	Period	TV	—	286	—	Line
	Pulse width	TVp	—	10	—	Line
	Vertical period	TVd	—	272	—	Line
	Back porch	TVb	—	2	—	Line
	Front porch	TVf	—	2	—	Line

Funzionamento del protocollo

Parte orizzontale

1) Il segnale di sincronizzazione orizzontale passa da livello logico alto a basso in corrispondenza del fronte di salita del Clock e rimane tale per 41 colpi di clock (THp).
2) Quando Hsync si riporta a 1, si attendono altri 2 segnali di Clock (THb, finestra posteriore), prima di avviare la trasmissione dei dati.
3) Al termine del secondo colpo di Clock, inizia la trasmissione dei dati in serie, partendo da sinistra a destra, cioè dal pixel 1 al 480. Ogni pixel porta con sé l'informazione sulla combinazione dei colori rosso, verde, blu.

4) Quando il 480° pixel è stato inviato, si attendono altri due segnali di clock (THf, finestra anteriore). Alla fine di essi, il segnale di sincronizzazione orizzontale torna al livello basso e tutto si ripete.

Parte verticale

In questo caso, come si può vedere dall'immagine 2.2, il "vero" Clock di riferimento è il segnale Hsync: un periodo di questo segnale è pari a 525 colpi del Clock di partenza.

Quando è riempito l'elemento n di ogni colonna (n=1,2,…..480), si passa alla riga successiva. In sostanza una riga si completa ogni volta che si giunge alla 480° colonna, procedendo da sinistra a destra.

1) Il segnale di sincronizzazione verticale passa da livello logico alto a basso e rimane tale per un tempo pari a 10 cicli orizzontali (5'250 CLK).
2) Nel momento in cui esso torna a 1, un contatore verticale conta altri due cicli orizzontali (TVb=1050 CLK).
3) Al termine del secondo ciclo ha inizio la trasmissione: i dati inviati sono 272, pari al numero di righe e il passaggio di ogni singolo dato (contenente 480 valori "colonna") dura esattamente come un ciclo orizzontale.
4) Quando l'ultimo dato è stato inviato, ci sono altre due linee di attesa (TVf, 1'050 CLK) al termine delle quali ha inizio la trasmissione successiva.

Quindi, mentre un ciclo orizzontale si compie in 525 CLK, un ciclo verticale dura: 286 * 525 = 150'150 CLK.

Nota la frequenza di Clock, si può ricavare il periodo del segnale di temporizzazione:
Tc=1/Frequency,
risalendo ai seguenti 3 valori:

- Tc max (con F=7,83 MHz) = 127,7 ns;
- Tc tipico (con F=9,00 MHz) = 111 ns;
- Tc min (con F=9,26 MHz) = 108 ns.

Quindi:

- Un ciclo orizzontale si compie tipicamente in 111*525 = 58,275 us;
- Un ciclo verticale (quindi totale) ha durata tipica pari a 58,275*286 = 16,67 ms.

Ritengo utile definire i segnali di Back e Front porch che compaiono nella tabella.

Sincronismo orizzontale

Back porch: tempo che intercorre tra il ritorno del segnale di sincronizzazione orizzontale allo stato logico 1 e l'invio della sequenza dei dati D1,…D480 (tempo attivo orizzontale).

Front porch: intervallo temporale tra il termine del tempo attivo orizzontale ed il nuovo passaggio di Hsync da 1 a 0.

Sincronismo verticale

Back porch: tempo che intercorre tra il passaggio del segnale Vsync da 0 a 1 e l'inizio del successivo tempo attivo verticale.

Front porch: intervallo temporale tra il termine dell'invio della sequenza di dati DH1, DH2,….,DH272 ed il nuovo passaggio di Hsync da 1 a 0.

A questo punto, ci si potrebbe chiedere per quale motivo nel ciclo orizzontale siano previsti esattamente 41 colpi di clock di attesa, oltre ai 4 dei due segnali di porch. E perché un ciclo verticale attenda 10 cicli orizzontali e il back porch prima di avere inizio.

In primo luogo, vale la pena di affermare che i protocolli di interfacciamento di TFT di diverse marche sono pressoché simili tra loro.
Per motivi inerenti esclusivamente al sincronismo dell'immagine, si suppone che il display, anziché avere una risoluzione pari a 480*272 pixel, ricopra un'area di 525*286 pixel. Essa si ottiene aggiungendo 45 colonne e 14 righe a quelle già esistenti.
Si può immaginare un rettangolo di area 480*272 all'interno di un rettangolo più grande, di ampiezza 525*286. Fisicamente, l'area racchiusa tra le due figure non esiste: essa è però necessaria per stabilire l'esatta ubicazione del rettangolo interno, che è quello reale.
Il tempo di trasferimento di una riga di pixel (composta da 525 elementi) è noto e già riportato in precedenza. Quindi nei 41 colpi di clock di attesa si ha il passaggio di pixel non esistenti. Essi rappresentano un artificio dettato dalla necessità di non perdere sincronismo durante il trasferimento di un'immagine. Non appena si esce dall'area interdetta, inizia la vera e propria trasmissione dei dati, a partire dal primo punto di coordinate ben note (in rosso nella figura 2.3). Un ragionamento analogo vale per il caso verticale.

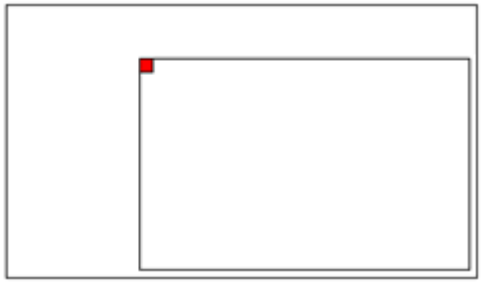


Fig. 2.3 – Il concetto di trasferimento delle immagini nel TFT

2.1.4 La back light

Si è visto come la retro-illuminazione sia necessaria per il funzionamento di un display a cristalli liquidi.
Il sistema back light consiste in 7 diodi LED, i quali vanno correttamente pilotati. Come si vede dalla tabella sottostante, per l'alimentazione del sistema back light si necessita di una tensione tipica pari a 22,8 V, oltre ad una corrente di 23 mA.

Parameter	Symbol	Min.	Typ.	Max.	Unit	Remark
Rated Voltage	V_{BL}	—	22.8	25	V	
Rated Current	I_L	—	23	—	mA	Ta=25℃

Poiché si dispone di una tensione di alimentazione pari a 5,5 V è necessario l'impiego di un circuito elevatore di tensione o "boost converter".
Per maggiori chiarimenti in merito, si rimanda alla sezione 2.9, "L'alimentazione del sistema".

2.2 Il Touch Screen

2.2.1 Generalità

Il TFT da noi utilizzato è un dispositivo tattile o Touch Screen.
Il Touch Screen è un sistema hardware che permette all'utente di interagire con un microprocessore (o un dispositivo simile) tramite lo sfioramento dello schermo.

Attualmente, i principali tipi di display tattili sono i seguenti:

- Resistivo;
- Capacitivo.

Nel primo si possono utilizzare sia le dita, sia un pennino, per le digitazioni.
Lo schermo capacitivo, invece, va adoperato esclusivamente con le dita poiché esse sono dotate di capacità. Infatti, dopo il tocco, il flusso di elettroni (necessario per il funzionamento) che attraversa la superficie del display non potrebbe subire variazioni con l'impiego di oggetti inanimati.
Il Touch Screen in dotazione è di tipo resistivo.

2.2.2 Il Touch Screen resistivo: funzionamento e caratteristiche principali

I Touch Screen di tipo resistivo sono i più diffusi e semplici da realizzare. Il principio di funzionamento di questi display si basa su tre strati sovrapposti, dove quelli esterni sono conduttivi (tipicamente ossido di indio) mentre il terzo, posizionato tra i due conduttori, è un pannello di vetro o di materiale acrilico che funge da isolante. Nel momento in cui viene esercitata una pressione sullo schermo i due strati conduttivi si avvicinano consentendo il passaggio di corrente tra di essi. La pressione del tocco causa un contatto elettrico che fornisce all'interfaccia elettronica tensioni analogiche proporzionali alle coordinate orizzontale e verticale. Le tensioni vengono poi convertite in segnali digitali dal controller del Touch.

La precisione di questo schermo è riconducibile alla risoluzione dei pixel del display (un risultato di notevole importanza per il riconoscimento della grafia, non raggiungibile da un Touch capacitivo).
Tuttavia, questo tipo di monitor è poco robusto e meno longevo rispetto al display capacitivo: occorre prestare attenzione a non portare a contatto con lo schermo oggetti troppo acuminati e, col passare del tempo, si nota una sensibile perdita di reattività al tocco. D'altro canto, il prezzo contenuto e la possibilità di operare in ampi intervalli di temperatura ed umidità rappresentano i suoi maggiori pregi.
Gli LCD Touch Screen di tipo resistivo sono attualmente impiegati nella maggior parte dei telefoni cellulari Touch, in particolare quelli di fascia medio-bassa.
Il display di cui si dispone utilizza una configurazione a 4 fili.

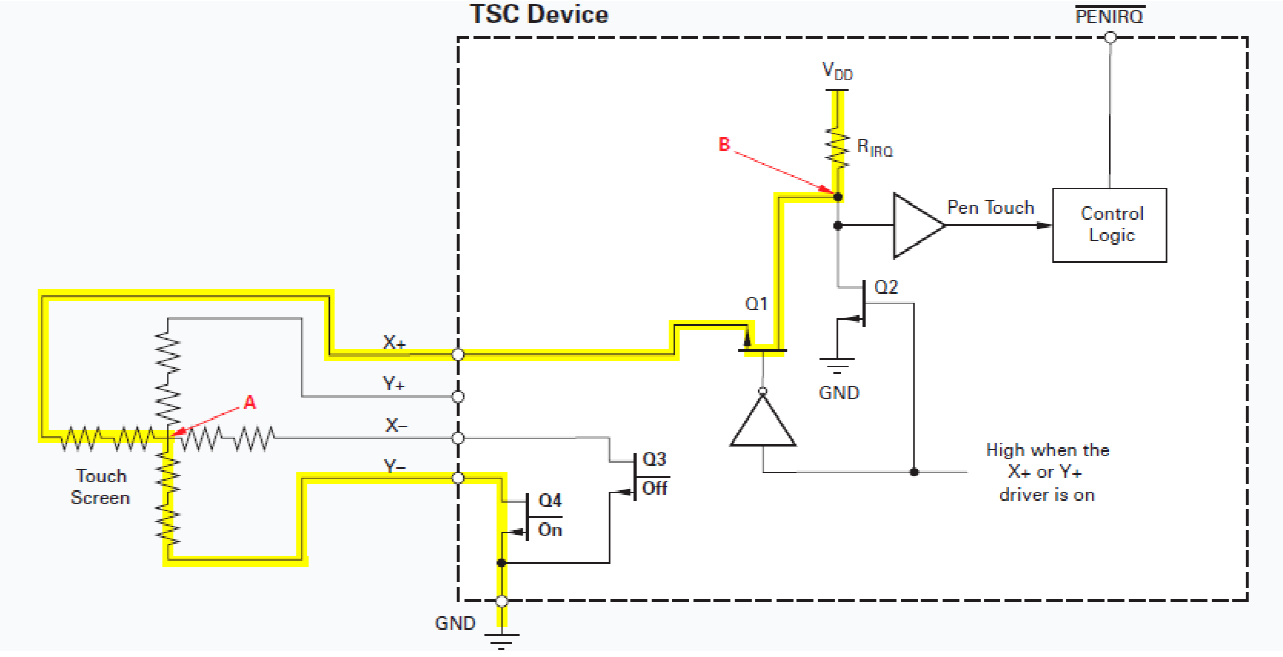


Fig. 2.4 – La configurazione a 4 fili

Entrando più in dettaglio, il circuito in figura 2.4 è in grado di determinare la locazione del pixel sollecitato utilizzando una coppia di coordinate (x,y).
In seguito ad una digitazione sul Touch Screen, è misurata la posizione corrispondente allo sfioramento, in coordinate cartesiane. La variazione di resistenza che ne consegue provoca l'applicazione di una ddp attraverso lo schermo in direzione Y. Poiché i due strati resistivi sono sovrapposti, un tocco li preme simultaneamente, c'è conduzione di corrente e la tensione può essere rilevata da uno degli elettrodi X.
Il contatto effettuato a seguito dello sfioramento crea un partitore di tensione in quel punto, quindi può essere determinata la coordinata Y. Il processo, essendo guidato, si ripete con la direzione X e la lettura della tensione viene effettuata da uno degli elettrodi Y.

Il controller del Touch Screen (**TSC2003**) è un convertitore analogico/digitale a 12 bit che provvede a leggere le variazioni di resistenza ai capi del Touch, la quale verrà convertita in segnale digitale.
Il nuovo valore di resistenza del circuito a 4 fili causato dal tocco sullo schermo ha come effetto una nuova tensione tra V_{DD} e GND. Questa tensione, all'ingresso dell'ADC, viene da esso convertita in digitale (2^{12} combinazioni possibili). I nuovi dati acquisiti saranno successivamente inviati al microcontrollore.

Il protocollo di comunicazione con cui il TSC dialoga con il microcontrollore è denominato "I2C".

2.2.3 Il protocollo di comunicazione I2C

L' I2C è uno standard ideato dalla Philips. È stato progettato nel 1980 per superare le difficoltà inerenti all'utilizzo di bus paralleli per le comunicazioni tra un'unità di controllo e le varie periferiche.
A differenza del protocollo RS232, che permette un collegamento punto-punto tra due sole periferiche, l'I2C permette di collegare sullo stesso bus un numero elevato di dispositivi, ognuno individuato da un suo indirizzo, tramite l'utilizzo di due sole linee.
Un grande vantaggio derivante dall' utilizzo di questo protocollo è che tutte le regole che bisogna rispettare per una corretta comunicazione vengono gestite a livello hardware.
Da quando è nato, l'I2C è stato aggiornato nel tempo per far fronte all'evoluzione dell'elettronica.
Tutte le modifiche apportate sono sempre state compatibili dall'alto verso il basso, ovvero gli integrati che soddisfano gli ultimi standard possono comunicare sempre con quelli della generazione precedente.
La prima versione del bus I2C permetteva di trasmettere fino a 100 Kbit/s. Nel 1998 la velocità è stata portata fino a 3,4 Mbit/s.
Un secondo pregio del protocollo è la possibilità di aggiungere o togliere delle periferiche dal bus senza influenzare il resto del circuito.

Specifiche elettriche del bus I2C

L'I2C è un bus seriale che necessita di due sole linee denominate SDA (Serial Data) e SCL (Serial Clock), entrambe bidirezionali, oltre alla linea di massa. La prima è utilizzata per il transito dei dati che sono in formato ad 8 bit, mentre la seconda è usata per trasmettere il segnale di clock necessario alla sincronizzazione della trasmissione.
Quando le linee SDA e SCL non vengono utilizzate, si trovano ad un livello logico alto.
Il bus I2C permette la connessione di più periferiche su uno stesso bus, consentendo però la comunicazione tra due soli dispositivi per volta.
Il dispositivo che trasmette le informazioni è chiamato trasmettitore mentre quello che le acquisisce è il ricevitore. L'essere trasmettitore o ricevitore non è una posizione fissa: un trasmettitore può anche divenire ricevitore in una fase diversa della trasmissione dei dati.
In ogni comunicazione è invece fissa la posizione del cosiddetto Master (Padrone) e dello Slave (Schiavo). Il Master è il dispositivo che inizia e termina la comunicazione, lo slave può solo ricevere o trasmettere

informazioni su richiesta del Master. Non tutti i dispositivi possono diventare dei Master del bus I2C. Ad esempio, una memoria per il mantenimento dei dati non potrà mai essere un Master, mentre è ragionevole supporre che un microcontrollore lo possa essere.
Su uno stesso bus è inoltre possibile la presenza di più Master, ma uno solo alla volta ricoprirà questo ruolo. Se per esempio due microcontrollori iniziano una comunicazione, anche se potenzialmente potrebbero essere ambedue dei Master, solo uno lo diventerà: in particolare sarà Master quello che ha iniziato per primo la comunicazione, mentre l'altro diverrà Slave. Ogni periferica inserita nel bus I2C possiede un indirizzo che la individua univocamente. Questo indirizzo può essere fissato dal produttore in sede di fabbricazione o parzialmente dettato dal progettista. Nel caso in cui l'indirizzo che ha l'integrato all'interno del bus I2C venga fissato dalla ditta che lo produce, su un bus non potranno essere presenti due dispositivi dello stesso tipo. Qualora si avesse tale necessità, questo limite potrebbe creare non pochi problemi. Per questo si ha la possibilità di impostare l'indirizzo dell'integrato intervenendo su alcuni bit.
Il **TSC2003** possiede due pin in uscita chiamati A0 e A1 (visibili in figura 2.5) per mezzo dei quali è possibile impostare i due bit meno significativi dell'indirizzo che definisce la periferica.

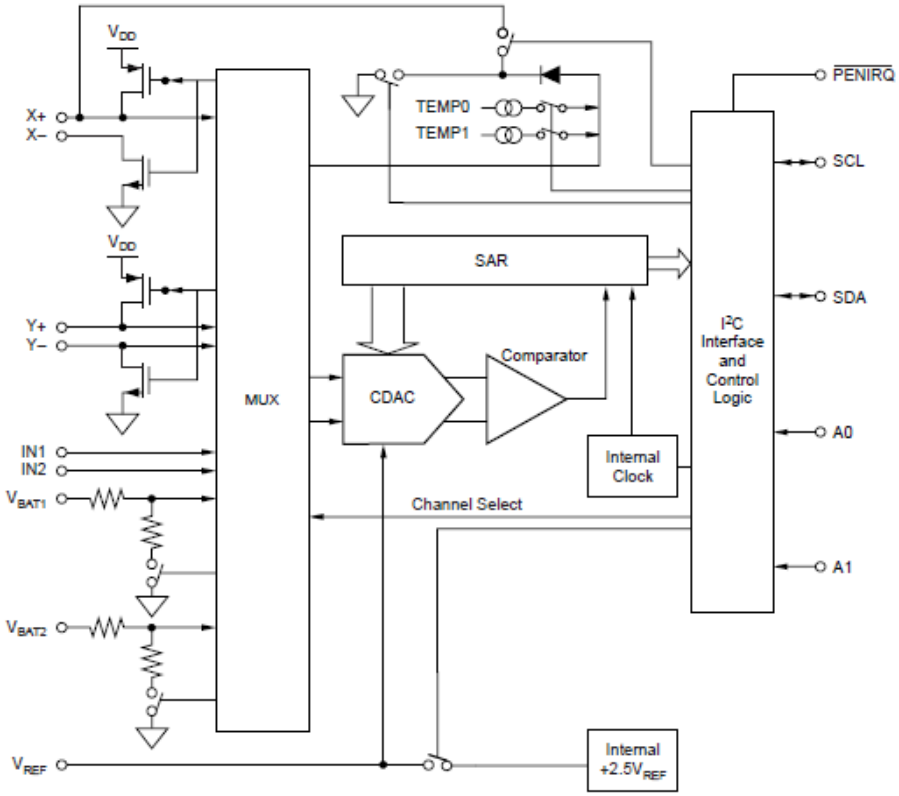


Fig. 2.5 – Implementazione delle linee SDA e SCL. Il piedino PENIRQ fornisce un segnale che, in seguito ad una pressione sullo schermo, passa da livello logico 1 a 0. Tale cambiamento è rilevato dal microcontrollore, il quale effettua determinati calcoli. Per gli approfondimenti si rimanda alla sezione 3.4 "Riconoscimento del tasto digitato".

Questo significa che è possibile inserire fino a quattro dispositivi del medesimo tipo sullo stesso bus. Quindi, un limite sul numero massimo di periferiche che è possibile connettere sul bus è impostato dall'indirizzo. Un vincolo più serrato è dato dalla capacità totale della linea che deve essere limitata a non più di 400 pF. In generale, poiché una linea introduce una capacità parassita di circa 80 pF/m, essa potrà essere lunga fino a 5 metri. Qualora si debbano raggiungere distanze maggiori, o il numero di dispositivi superi tale capacità, è possibile spezzare il bus in più parti, per mezzo di ripetitori. Grazie a questi integrati diventa possibile avere 400 pF di capacità parassite per ogni semiparte del bus. Sarà il Master, successivamente, a scandire il sincronismo e la velocità di trasmissione.

Comunicazioni sul bus I2C

Come detto in precedenza solo le periferiche che possono essere dei Master sono in grado di avviare una comunicazione.

I passi da compiere in questo tipo di trasmissione dati sono i seguenti:

- Il Master controlla che le linee SDA e SCL non siano attive (livello logico alto).
- Se il bus è libero il Master invia alle periferiche il messaggio di "bus busy" (il bus è ora occupato). Le periferiche si pongono in ascolto per comprendere con chi il Master ha intenzione di comunicare.
- Il Master provvede all'invio del segnale di sincronizzazione sulla linea SCL, rappresentato da una onda quadra non necessariamente periodica.
- Il Master invia l'indirizzo della periferica con la quale vuole comunicare.
- Il Master segnala poi se la comunicazione che vuole intraprendere sia di lettura o scrittura.
- Il Master attende la risposta da parte della periferica che nella chiamata ha riconosciuto il proprio indirizzo. Se nessuna periferica risponde, il Master libera il bus.
- Dopo l'avvenuto riconoscimento della periferica, il Master inizia lo scambio dei dati. La trasmissione avviene mandando pacchetti di 8 bit di dati. Ad ogni pacchetto si deve attendere il segnale che avvisa dell'avvenuta ricezione (Acknowledge).
- Quando la trasmissione è terminata il master libera il bus inviando un segnale di stop.

In seguito si riporta il processo dettagliato delle varie fasi sopra elencate.

1) Bus libero e segnale di Start

L'hardware del microcontrollore controlla le linee SDA e SCL per un tempo superiore al massimo periodo di trasmissione consentito dall'hardware. Se il bus risulta libero, il Master invia la sequenza di Start, che consiste nel porre la linea SDA a livello basso quando SCL è a livello alto.

Dopo l'invio della sequenza di Start, il bus è considerato occupato. In seguito alla transizione della linea SDA da livello alto a basso, il Master invia il segnale di sincronizzazione per le altre periferiche. A differenza della sequenza di Start e di Stop la linea SDA assume un valore valido solo se la linea SCL è a livello logico basso. Ciò significa che non sono ammesse transizioni di livello logico della linea SDA durante il livello logico alto della linea SCL, se non da parte del Master per inviare un nuovo Start o uno Stop.

2) Indirizzamento

I sette bit dell'indirizzamento vengono inviati dal bit più significativo al meno significativo. In coda a questo indirizzo viene aggiunto un bit per segnalare se il Master vuole intraprendere, con la periferica individuata da tale indirizzo, una comunicazione di scrittura o di lettura.

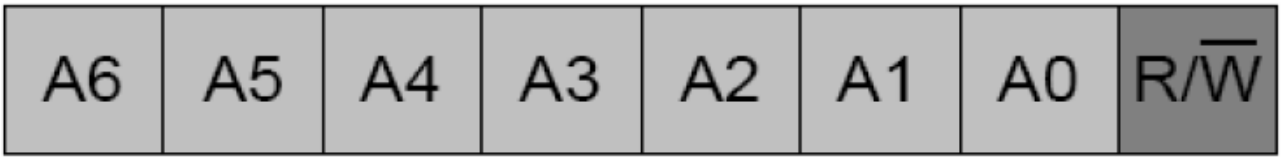


Fig. 2.6 – Formato del byte di indirizzamento

In particolare se tale bit è 0 vuol dire che il Master intende scrivere sulla periferica, se il bit è 1 esso vuole leggere dalla periferica.

3) Acknowledge

L'invio dell'indirizzo a 7 bit della modalità del colloquio (lettura/scrittura), avviene grazie a otto transizioni, da livello logico alto a basso, della linea SCL. Al nono impulso di clock il Master attende una risposta di un

bit da parte della periferica che ha interrogato. La risposta della periferica consiste nel mantenere a livello basso la linea SDA per la durata di un ciclo SCL.
In gergo si dice che il Master aspetta l'Acknowledge da parte della periferica chiamata, a cui una sola periferica risponde.

4) Scambio dati
Dopo l'avvenuto riconoscimento, si ha lo scambio dei dati verso la periferica, nel caso di scrittura, o dalla periferica al Master, in caso di lettura. Si prenda ad esempio una fase di lettura da memoria. In primo luogo bisogna scrivere alcuni byte per impostare l'indirizzo che si vuole leggere, dopodiché si può effettivamente recuperare il byte.
Ad ogni invio di un byte è necessario l'Acknowledge del byte inviato o ricevuto. In particolare, se il Master invia un byte allo Slave si aspetta, dopo l'ottavo bit, un bit basso sulla linea SDA. Se lo Slave sta inviando un byte al Master, attende che quest'ultimo invii un bit alto dopo aver ricevuto il byte.
La mancanza dell'Acknowledge determina un errore di comunicazione.

5) Sequenza di Stop
Quando la comunicazione è terminata il Master libera il bus inviando la sequenza di Stop che consiste nella transizione dal livello basso ad alto della linea SDA, quando la linea SCL si trova ad un livello logico alto. Se il Master deve effettuare una nuova comunicazione con uno Slave diverso può inviare una nuova sequenza di Start.

2.3 Gestione del modulo TFT

2.3.1 Introduzione

Per gestire in maniera corretta lo schermo TFT Sharp **LQ043T3DX0A**, la ditta costruttrice fornisce un modulo che, oltre al display, contiene:

- FPGA;
- memoria SDRAM;
- memoria Flash SPI;
- oscillatore a 25 MHz al quarzo interno (per fornire il segnale di temporizzazione).

Inoltre è stato necessario utilizzare un microcontrollore con lo scopo di svolgere varie operazioni, sia per quanto riguarda il sistema composto da TFT e Touch Screen, sia, come si vedrà in seguito, per il corretto funzionamento del telefono cellulare.

Lo schema a blocchi seguente introduce le relazioni tra le varie componenti.

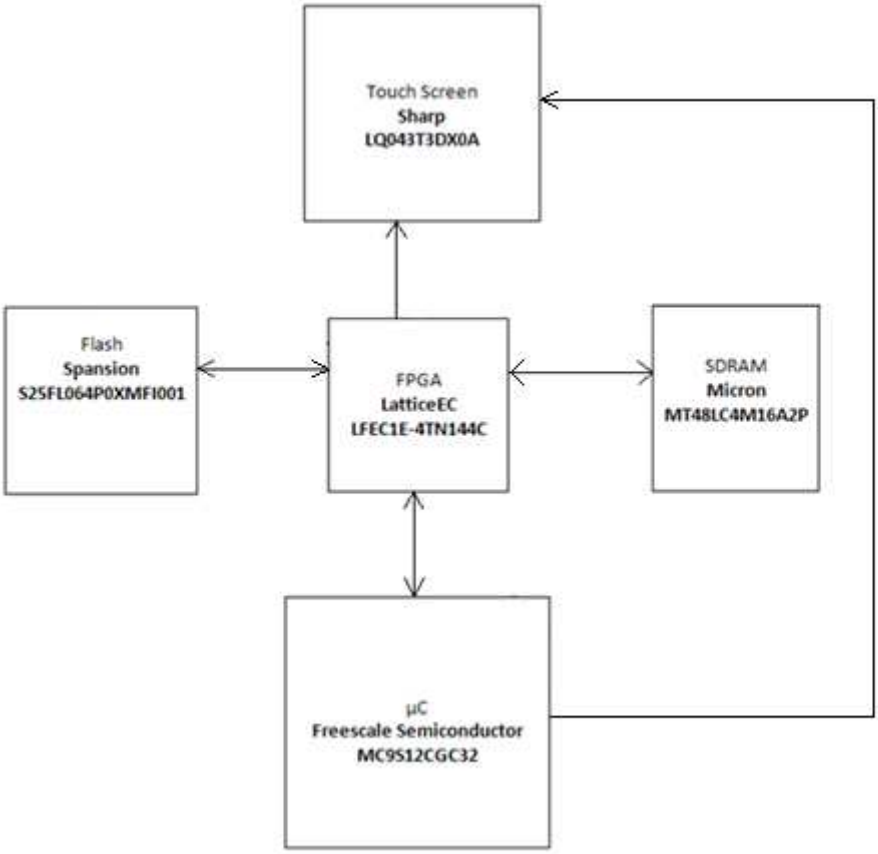


Fig. 2.7 – Gestione del modulo TFT

2.3.2 Motivazioni delle scelte tecniche

Un comune microcontrollore, per problemi dovuti alla velocità di esecuzione, non è in grado di coordinare in tempi ragionevoli il protocollo di interfacciamento del TFT. Per questo si è costretti ad utilizzare una FPGA.
I compiti del micro in questo caso sono ridotti a:

- far comparire nel TFT una sequenza di foto pre-caricate (al massimo 16);
- scrivere nel TFT singoli pixel RGB (si veda la sezione 3.5).

Per consentire al microcontrollore di svolgere queste due mansioni, l'FPGA prevede un set di 8 registri, ciascuno di essi ampio 8 bit.
Il micro può avere acceso ad ogni pixel del TFT, fornendone il corrispondente indirizzo ed i colori RGB che si vogliono visualizzare.

La memoria Flash è utilizzata per contenere il programma dell'FPGA e le foto che si desiderano visualizzare.
La RAM è impiegata dall'FPGA in due fasi successive:

1) Appena acceso il modulo, i dati presenti nella Flash (le 16 foto) vengono trasferiti nella RAM.
   È importante evidenziare che, non appena la prima immagine è caricata nella RAM, essa compare subito su TFT, come previsto dal programma dell'FPGA tra le impostazioni di avvio, nell'attesa che termini il trasferimento delle successive 15 immagini.
2) Se il micro impone all'FPGA di processare un'altra foto, l'FPGA, dopo aver acquisito il comando, trasferisce i dati della foto invocata dalla RAM al display TFT.

Ci si potrebbe chiedere per quale motivo sia stata utilizzata una RAM quando la Flash contiene già tutto il necessario. Essa è stata adoperata per risolvere problemi legati alla tempistica e alle esigenze umane: la Flash è molto più lenta della RAM nella fase di lettura dei dati, pur avendone in questo caso la medesima capacità.

Se le immagini da visualizzare su display provenissero direttamente dalla Flash e si volesse osservarle in sequenza, l'effetto visivo non sarebbe istantaneo, bensì un caricamento a tratti lento, poco elegante e decisamente poco innovativo. Infatti:

Tempo necessario per trasferire 60 Mb di dati (cioè 15 immagini) da Flash a RAM:

- T_{trasf} = 6,0 s (visualizzato su oscilloscopio).

Tempo necessario per trasferire 4 Mb di dati (cioè una sola immagine) da Flash a RAM :

- T_{trasf} = 6/15 s = 400 ms.

Ricordando il protocollo di interfacciamento del TFT, per comparire su display un'immagine impiega mediamente 16,67 ms, un tempo decisamente inferiore a quello necessario se essa provenisse dalla Flash.

È quindi obbligatorio l'utilizzo di una memoria che, in fase di lettura dei dati, sia molto più veloce rispetto alla Flash: la RAM.

In sintesi, l'FPGA svolge la funzione di un controller video: all'accensione del sistema, essa carica il contenuto della memoria Flash nella RAM tramite colloquio con il microcontrollore. I dati compaiono poi nel TFT seguendo il protocollo di interfacciamento già noto.

La figura 2.8 illustra il posizionamento dei vari elementi.

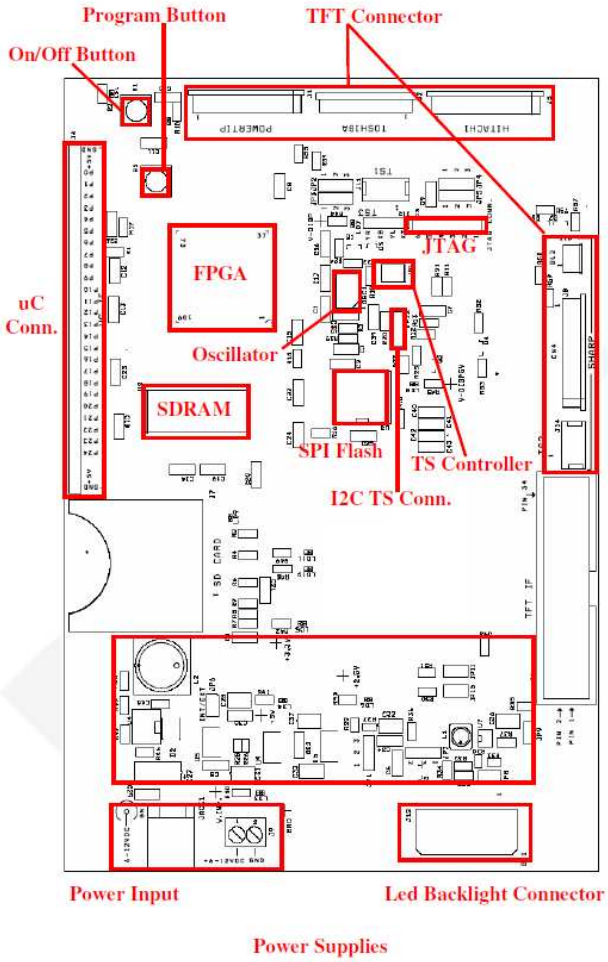


Fig. 2.8 – Organizzazione del modulo TFT

2.4 L'FPGA (Field Programmable Gate Array)

2.4.1 Introduzione

Le FPGA sono dispositivi digitali simili a un microprocessore la cui funzionalità hardware è implementabile via software.

Esse sono programmate direttamente dall'utente finale, consentendo la diminuzione dei tempi di progettazione e di prova sul campo dell'applicazione.
Il vantaggio derivante da loro impiego è che permettono di apportare eventuali modifiche o correggere errori semplicemente riprogrammando il dispositivo in qualsiasi momento.
L'FPGA montata nel TFT è la **LFEC1E-4TN144C** a 144 pins della LatticeEC. Essa è programmata in linguaggio VHDL.
VHDL è l'acronimo di "VHSIC Hardware Description Language". A sua volta VHSIC vuol dire "Very High Speed Integrated Circuits". Il VHDL nacque nel 1987.

2.4.2 L'organizzazione interna dell'FPGA

L' architettura LatticeEC (EConomy) prevede un array di blocchi logici circondato da celle programmabili di ingresso/uscita (PIC). Alle estremità di alcuni tipi di blocco (i PFU) sono intercalati segmenti di memoria RAM (EBR). I blocchi logici presenti sono di due tipi:

- Unità Funzionale Programmabile (PFU);
- Unità Funzionale Programmabile senza RAM (PFF).

La PFU racchiude le basi per la logica, l'aritmetica, la RAM, la ROM e le funzioni di registro. Il blocco PFF contiene elementi di base per la logica, l'aritmetica e le funzioni ROM.
Sia la PFU che la PFF sono molto flessibili e permettono di implementare operazioni complesse in modo rapido ed efficiente. I blocchi logici sono disposti secondo un array bidimensionale ed un solo tipo di blocco viene utilizzato per fila. Inoltre, ogni tre file di PFF è presente una riga di PFU.
Al termine di ogni riga contenente i blocchi di memoria RAM è presente un sistema PLL (Phase Locked Loop, anello ad aggancio di fase).
In estrema sintesi, questo circuito permette la moltiplicazione e la divisione di segnali di Clock da inviare ad altri dispositivi sfruttando la precisione di un oscillatore al quarzo interno.
L'FPGA, per poter essere programmata da Personal Computer, dispone di una porta JTAG.
I dispositivi LatticeEC prevedono una tensione di core pari a 1,2 V.
L'immagine seguente illustra con maggior chiarezza quanto appena descritto.

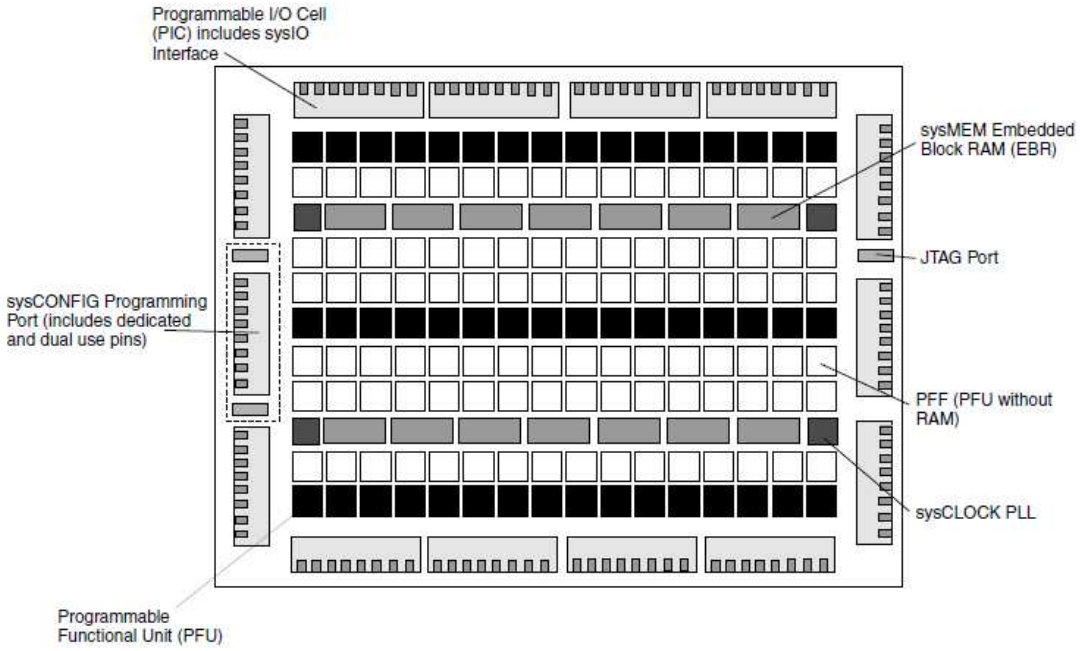


Fig. 2.9 – I blocchi interni dell'FPGA

2.5 La memoria RAM

2.5.1 Nozioni di base

La memoria RAM serve a contenere le immagini che devono comparire dopo l'accensione del display.
Le RAM (Random Access Memory) disponibili sul mercato si dividono in due grandi categorie:

- **DRAM**;
- **SRAM**.

La **DRAM**, acronimo di *Dynamic Random Access Memory*, è un tipo di RAM che immagazzina ogni bit in un diverso condensatore. Il numero di elettroni presenti nel condensatore determina se il bit è 1 o 0.
Se il condensatore perde la carica, anche l'informazione scompare: durante il funzionamento la ricarica avviene periodicamente. Per la caratteristica di perdere le informazioni in mancanza di energia, la DRAM viene definita anche *volatile.*
Attualmente le DRAM sono organizzate in un insieme di matrici quadrate di condensatori allo stato solido, la cui dimensione minima è 1024 x 1024 bit (cioè 1 Mbit).
Durante la fase di lettura si vuole acquisire il valore di un bit della matrice, identificato dal suo numero di riga e di colonna. Per ottenere la lettura di questa cella, in realtà vengono letti e riscritti tutti gli elementi della riga della matrice a cui la cella appartiene. L'operazione di riscrittura è necessaria perché durante la lettura la carica elettrica contenuta nelle celle si degrada; così i dati devono essere riscritti. Dei dati ottenuti viene poi selezionato per l'uscita un unico bit, quello relativo alla colonna della cella indirizzata. L'operazione di scrittura di una cella avviene in modo simile.
Ciascuna cella di una DRAM richiede un condensatore allo stato solido per memorizzare il bit di informazione e un transistore MOS per controllare l'accesso alla cella.
Indipendentemente dalle operazioni di lettura-scrittura, è necessario effettuare periodicamente (a frequenze dell'ordine dei KHz) un'operazione di lettura e riscrittura fittizia di ogni riga della matrice (*refreshing*). Da questa necessità deriva il termine *dinamica* che caratterizza questo tipo di memoria. Mentre nelle prime memorie DRAM questa operazione doveva essere comandata esternamente, al giorno d'oggi è automatizzata direttamente all'interno del circuito integrato della memoria utilizzando una unità logica dedicata; questo complica il circuito e ne diminuisce la capacità.

La **SRAM**, acronimo di *Static Random Access Memory*, è un tipo di RAM volatile che non necessita di refresh. I banchi di memorie SRAM consentono di mantenere le informazioni per un tempo teoricamente infinito, hanno bassi tempi di lettura e bassi consumi.
La necessità di usare molti componenti per cella le rende però più costose delle DRAM.

La memoria adoperata nella nostra applicazione, scelta per fattori inerenti al costo, è una **SDRAM.**
La **SDRAM** (Synchronous DRAM) è un tipo evoluto di DRAM. Mentre la DRAM ha un'interfaccia asincrona, cioè risponde immediatamente ai comandi, la SDRAM è sincrona, cioè attende il segnale di clock successivo per rendere effettivi i cambiamenti ordinati. Il clock le permette un modo di operare più complesso rispetto alla semplice DRAM: il pipelining.
Il termine *pipeline* in informatica indica un assemblato composto da più parti. Ogni elemento che lo compone provvede a ricevere un dato in ingresso, ad elaborarlo e poi a trasmetterlo all'elemento successivo. Quindi il flusso dei dati percorre tutti gli elementi fino all'ultimo, come quando una conduttura è attraversata da un fluido.
La parola pipeline, in inglese, indica appunto una tubatura.
Grazie alla pipeline, la memoria può accettare un comando in arrivo prima di aver terminato lo svolgimento del precedente. Così possono essere trasmessi in sequenza più comandi di scrittura senza aspettarne il completamento. In lettura il dato compare dopo un preciso numero di cicli di clock rispetto al comando, ma

non è necessario aspettare che appaia prima di mandare l'istruzione successiva. Questo intervallo è detto *latenza* ed è un fattore molto importante per le prestazioni della memoria.

2.5.2 Caratteristiche tecniche e specifiche

La memoria inserita nel modulo TFT è la **MT48LC4M16A2P** della Micron.
Essa è una SDRAM in grado di contenere 67'108'864 bit (= 64 Mbit). Internamente, è configurata come una DRAM a 4 banchi, suddivisi a loro volta in 4 settori, che opera ad una tensione di 3,3 V con un' interfaccia sincrona (tutti i segnali sono sintonizzati sul fronte di salita del segnale di clock, CLK).
Ogni banco contiene quindi 16'777'216 bit ed è suddiviso in 4'096 righe e 256 colonne. Ciascuna di queste colonne è composta a propria volta da 16 bit. Infatti:

- 4'096 righe * 256 colonne * 16 bit per colonna = 16'777'216 bit per banco;
- 16'777'216 bit per banco * 4 banchi = 67'108'864 bit totali.

Lo specchietto seguente permette di capire meglio la suddivisione della memoria.

sector 3	sector 3	sector 3	sector 3
sector 2	sector 2	sector 2	sector 2
sector 1	sector 1	sector 1	sector 1
sector 0	sector 0	sector 0	sector 0
bank 0	bank 1	bank 2	bank 3

Fig. 2.10 – 1° suddivisione della memoria RAM

Gli accessi per la lettura e la scrittura sulla RAM sono orientati: essi partono da una locazione selezionata e continuano per un numero programmato di locazioni in una sequenza prestabilita. L'accesso inizia con la registrazione del comando ACTIVE seguito da READ o WRITE.
Con il comando ACTIVE si selezionano il banco e la riga a cui si deve accedere. È immediato risalire al numero di pins necessari ad un indirizzamento senza ambiguità:

4 banchi → 2 piedini (BA0,BA1);
4'096 righe→ 12 piedini (A0-A11).

Il comando READ o WRITE è usato per selezionare la locazione della colonna di partenza per l'accesso in memoria.

256 colonne→ 8 piedini (A0-A7).

In sintesi:

	4 Meg x 16
Configuration	1 Meg x 16 x 4 banks
Refresh count	4K
Row addressing	4K (A0–A11)
Bank addressing	4 (BA0, BA1)
Column addressing	256 (A0–A7)

Questa SDRAM adopera un'architettura interna con pipeline per poter svolgere operazioni ad alta velocità.
La memoria prevede 2 possibili tipi di refresh: l'AUTO ed il SELF.
L'AUTO refresh è usato nelle normali operazioni della RAM. Questo comando non è permanente ma può essere dichiarato ogni volta sia richiesto un refresh. Tutti i banchi attivi devono essere pre-caricati prima di emettere tale comando.
Il SELF refresh può essere sfruttato per mantenere i dati in memoria nel caso in cui il resto del sistema venga spento. In questo caso la RAM mantiene i dati anche in assenza del clock esterno (fornito dall'FPGA). Dopo aver attivato il SELF refresh, la SDRAM fornisce solamente il proprio clock interno, per poter svolgere correttamente il ciclo di rinfresco dei dati.
In questa applicazione la modalità di rinfresco è AUTO. Esso dura in totale 16 ms, durante i quali avvengono 4096 cicli di refresh (il processo si compie di riga in riga). Si calcola la durata del rinfresco per ogni riga:

- (16 ms) /(4'096 righe) = 3,9 us/riga.

Ora che la struttura della SDRAM è stata definita, è importante comprendere in quale modo essa riesca ad immagazzinare i dati visualizzati sul display Touch.

2.5.3 La memoria RAM: interazione con il display LCD

Il TFT in dotazione prevede un display con risoluzione pari a 480x272 pixel. Ciò significa che il numero massimo di pixel da gestire è:

- 480*272 = 130'560.

Poiché per ogni Rosso (Red), Verde (Green) e Blu (Blue) facenti parte del singolo pixel sono previsti 8 bit (256 combinazioni) e i colori che si combinano sono per l'appunto 3, un'immagine occupa in memoria:

- 130'560*8*3 = 3,133440 Mbit (= 391,680 KB).

Fisicamente i Mbit allocati sono 4 (circa il 20% in più) poiché una parte della memoria è riservata per altre informazioni. La Micron da 64 Mbit ha un bus dati di 16 bit ed i dati sul colore sono organizzati come rappresentato in figura 2.11. Si evince che il massimo numero di immagini (frames) che possono essere memorizzate nella SDRAM è:

- 64 Mb / 4 Mb = 16.

Fig. 2.11 – Informazioni contenute in un singolo pixel

Ogni pixel del display TFT corrisponde ad una posizione in memoria, in cui il numero del pixel è l'indirizzo della locazione ed il dato sul colore è il contenuto della memoria stessa (le possibili variazioni di Rosso, Verde e Blu). La figura e la tabella successive illustrano come la memoria video del TFT è mappata nella SDRAM.

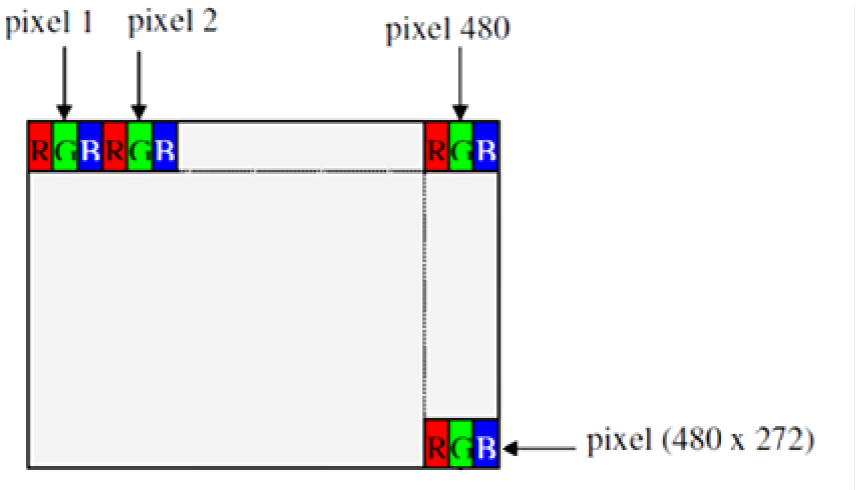


Fig. 2.12 – Mappatura dell'immagine nella SDRAM

TFT	**SDRAM**	
Pixel 1	Indirizzo 0	Dati RGB del pixel 1
Pixel 2	Indirizzo 1	Dati RGB del pixel 2
…..	…..	…..
Pixel 480	Indirizzo 479	Dati RGB del pixel 480
…..	…..	…..
Pixel (480*272)	Indirizzo (480*272 - 1)	Dati RGB del pixel (480*272)

Tenendo conto che la SDRAM è suddivisa in 4 banchi e sulla possibilità di memorizzare fino a 16 immagini, si intuisce che ogni banco deve immagazzinare 4 immagini. Ciascun banco, come si sa, è poi suddiviso in 4 settori (numerati da 0 a 3). A questo punto, lo schema concettuale è completo.

frame 4, sector 3	frame 8, sector 3	frame 12, sector 3	frame 16, sector 3
frame 3, sector 2	frame 7, sector 2	frame 11, sector 2	frame 15, sector 2
frame 2, sector 1	frame 6, sector 1	frame 10, sector 1	frame 14, sector 1
frame 1, sector 0	frame 5, sector 0	frame 9, sector 0	frame 13, sector 0
bank 0	bank 1	bank 2	bank 3

Fig. 2.13 – 2° ed ultima suddivisione della memoria RAM

2.6 La memoria Flash

2.6.1 Funzionamento

Il funzionamento delle memorie flash si basa sul principio dell'effetto tunnel.
Esso rappresenta una manifestazione molto affascinante della natura quantistica della materia. Lo si può interpretare con un esempio: si è di fronte un muro alto alcuni metri e lo si deve superare. L'unico modo, secondo la meccanica classica, è arrampicarsi fino in cima, ovvero portarsi ad un'energia potenziale

gravitazionale maggiore rispetto a quella del muro. Saltando giù dalla parte opposta, si riavrà indietro quanto speso sotto forma di energia cinetica. La meccanica quantistica, invece, sostiene che se si corre verso il muro, c'è una probabilità infinitesima ma non nulla di riuscire ad attraversarlo e di ritrovarsi dalla parte opposta, come se si avesse attraversato un tunnel. Il buonsenso suggerisce che ciò non avviene mai per oggetti di uso comune che si possono vedere e toccare; il fenomeno viene invece osservato nel campo dell'infinitamente piccolo.

Ora, si può pensare ad una particella in moto verso una barriera di potenziale.
Si può calcolare la probabilità che l'incontro con l'ostacolo produca un "rimbalzo" o che la barriera sia oltrepassata dal corpuscolo. La difficoltà di attraversamento è determinata dall'altezza della barriera, dalla sua larghezza e dalla massa della particella. Risulta ovvio che il passaggio è tanto più facile quanto più la barriera risulta bassa e stretta; inoltre, tale difficoltà si riduce rapidamente a zero quando la massa della particella oltrepassa i limiti del mondo subatomico (si ricorda che la massa di un elettrone è dell'ordine di 10^{-30} kg).

Il concetto di base che permette alla memoria Flash (o ad altri dispositivi come il cosiddetto "diodo ad effetto tunnel") di funzionare è quindi il seguente: un insieme di elettroni si ritrova a percorrere un punto in cui, statisticamente, ha scarsissima probabilità di passare. Qui vi rimane "intrappolato" senza più uscirne, andando a costituire un bit che si "insedia" in memoria.

2.6.2 Caratteristiche di base

La **memoria Flash** è una tipologia di EEPROM inventata in Giappone a metà degli anni '80. Si trattava di un sistema di archiviazione innovativo in quanto includeva alcuni vantaggi delle memorie EEPROM (impiegate nei BIOS dei PC e modificabili solo in via eccezionale) di cui sono un'evoluzione, uniti alla versatilità delle memorie RAM (di rapido accesso in lettura e scrittura ma incapaci di conservare i dati dopo lo spegnimento dell'alimentazione elettrica).
La tecnologia della memoria Flash consente un accesso in lettura più lento rispetto a quello di una memoria RAM ed impone meno flessibilità in riscrittura e cancellazione. La Flash, a differenza della SRAM, è però in grado di mantenere i dati memorizzati anche quando è disconnessa dal circuito, quindi si presta bene a immagazzinare informazioni di massa che non devono essere continuamente riscritte nel dettaglio, mentre non è adatta a memorizzare dati che evolvono in tempi molto brevi.
Si distinguono due tipi di memoria Flash: la **NOR** e la **NAND**.

La **NOR,** così chiamata a causa del tipo di funzione logica che utilizza nel processo di funzionamento, ha un accesso lento a scrittura e cancellazione, ma permette una localizzazione delle informazioni memorizzate di tipo casuale, ossia qualsiasi dato può essere recuperato in ogni momento indipendentemente da tutti gli altri. Questo tipo di memoria Flash è indicata per l'utilizzo nei casi in cui le informazioni memorizzate non devono essere aggiornate di continuo.
La **NAND** è più rapida in fase di cancellazione e scrittura, ha una capacità di memorizzazione maggiore ed un costo per byte minore rispetto alla NOR. Lo svantaggio è che può accedere ai dati solo in maniera sequenziale, ossia in modo da recuperarli in un ordine prestabilito. Questo la rende adatta ad un impiego sui dispositivi portatili, perché in fase di recupero delle informazioni registrate non serve una velocità elevata.

In generale le memorie Flash non perdono i dati memorizzati quando vengono disconnesse dall'alimentazione e possono essere trasferite con facilità da un sistema all'altro. Inoltre sono estremamente compatte e, non contenendo parti meccaniche (a differenza degli hard disk) sono estremamente robuste e

resistenti agli urti, il ché le rende ideali per essere associate a dispositivi portatili come le macchine fotografiche digitali. Significativo è il loro bassissimo assorbimento energetico.

In questo progetto, la Flash serve a contenere il programma che viene lanciato all'avvio del sistema e tutte le foto che si devono caricare in RAM (al massimo 16). Quindi, nella trasmissione dati, la FPGA è il Master e la Flash è l'unico Slave. Il protocollo con cui i due dispositivi comunicano si chiama SPI.

2.6.3 Il protocollo SPI

Il Serial Peripheral Interface o SPI è un sistema di comunicazione tra un microcontrollore (o un dispositivo analogo) ed altri circuiti integrati.
È un bus standard ideato dalla Motorola.
La trasmissione avviene tra un Master ed uno o più Slave. Il Master controlla il bus, emette il segnale di clock, decide quando iniziare e terminare la comunicazione.
Il bus SPI si definisce:

- di tipo seriale;
- sincrono, per la presenza di un clock che coordina la trasmissione e ricezione dei singoli bit e determina la velocità di trasmissione;
- full-duplex, in quanto il colloquio può avvenire contemporaneamente in trasmissione e ricezione.

Per quanto riguarda la velocità di scambio dei dati, non vi è un limite minimo ma un massimo che va determinato dai datasheets dei singoli componenti connessi e dal loro numero, in quanto ogni dispositivo collegato sul bus introduce sulle linee una capacità parassita.
Il sistema di comunicazione è progettato per lo scambio di dati tra dispositivi montati sulla stessa scheda elettronica (o tra schede vicine tra loro) in quanto non prevede soluzioni hardware per trasferire dati tra dispositivi lontani connessi con cavi soggetti a disturbi.
Il sistema è comunemente definito a quattro fili. Con questo si intende che le linee di connessione che portano i segnali sono quattro. Va però tenuto conto che deve esserci una connessione di massa (GND) e che fisicamente i fili sono cinque.

Per lo scambio dei dati l'SPI utilizza i seguenti segnali:

- SS - "Slave Select". Quando passa a livello logico 0, lo Slave si mette in attesa per ascoltare il segnale di clock ed i dati. In questa applicazione non viene utilizzato in quanto lo Slave è uno solo.
- SCK - clock seriale generato dal Master che controlla l'invio e la ricezione dei dati.
- SDO - linea seriale percorsa dai dati di uscita da un dispositivo all'altro.
- SDI - linea seriale dei dati in ingresso ad un dispositivo SPI .

Nessun componente è solo un trasmettitore o solo un ricevitore nel protocollo SPI. Ogni dispositivo ha due linee di dati, una di ingresso SDI e una di uscita SDO.
Lo scambio dati è scandito dal clock presente sulla linea denominata SCK. Il clock è generato dal Master che, tramite esso, sincronizza i vari eventi. Soltanto il Master può controllare la linea di clock SCK. Questo è un vantaggio perché il clock può variare senza corrompere i dati. La frequenza di passaggio dei dati cambia in base al clock.

2.6.4 Caratteristiche tecniche e specifiche

La memoria installata nel modulo TFT è la **S25FL064P0XMFI001** della SPANSION, in logica **NOR**.

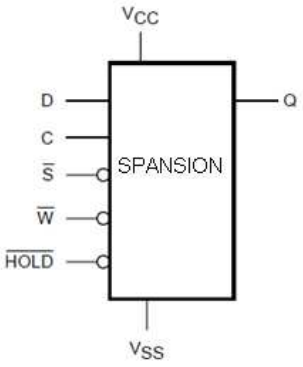
È una Flash seriale funzionante a bassa tensione con capacità pari a 64 Mbit, suddivisa in 8 banchi da 8 Mbit l'uno. Le sue principali caratteristiche sono:

- Possibilità di cancellazione per settori (512 Kbit);
- Possibilità di cancellazione totale (64 Mbit);
- Programmazione di una pagina in 1.4 ms (tempo tipico);
- Singola alimentazione, con tensioni comprese tra 2.7 e 3.6 V;
- Interfaccia seriale compatibile con Bus SPI ad alta velocità;
- Frequenza di Clock massima 50 MHz;
- Oltre 100'000 cicli di programmazione/cancellazione per settore;
- Ritenzione dei dati per oltre 20 anni.

La memoria può essere programmata da 1 a 256 byte per volta, ed è organizzata in 128 settori da 512 Kbit l'uno. Ogni settore contiene a sua volta 256 pagine, ciascuna di esse ampia 256 byte.
Quindi:

- 256 pagine * 128 settori = 32'768 pagine totali;
- 32'768 pagine totali * 256 byte per pagina = 8'388'608 byte totali (= 64 Mbit, esattamente come la SDRAM).

La figura 2.14 illustra la piedinatura della memoria Flash.



C	Serial Clock
D	Serial Data Input
Q	Serial Data Output
$\overline{S}$	Chip Select
$\overline{W}$	Write Protect
$\overline{HOLD}$	Hold
V_{CC}	Supply Voltage
V_{SS}	Ground

Fig. 2.14 – Disposizione dei piedini della memoria Flash

2.6.5 Descrizione dei segnali

Serial Data Output (Q). Questo segnale di uscita è usato per trasferire dati in modo seriale verso altri dispositivi. L'uscita del dato avviene in corrispondenza al fronte di discesa del Serial Clock (C).
Serial Data Input (D). Questo segnale di ingresso è adoperato per trasferire dati seriali nella memoria. Esso riceve istruzioni, indirizzi e le informazioni per la programmazione del dispositivo stesso. I valori vengono **"**agganciati" durante il fronte di salita del Serial Clock.
Serial Clock (C). Questo ingresso fornisce la temporizzazione per l'interfaccia seriale.
Chip Select (S). Quando **S** è alto, il dispositivo è disabilitato e Serial Data Output si trova in alta impedenza. Mentre è in corso un ciclo di cancellazione o scrittura, il dispositivo è in standby. Pilotando Chip Select a 0, si passa in modalità attiva. Dopo l'accensione è richiesto un fronte di discesa in **S** per l'avvio di qualsiasi istruzione.
Hold (HOLD). Il segnale di Hold è usato per porre in pausa una comunicazione seriale con un dispositivo (nel nostro caso la FPGA) senza deselezionarlo. Per attivare questa condizione occorre selezionare il dispositivo interessato (con Chip Select pilotato allo stato logico basso).
Write Protect (W). Lo scopo principale di questo segnale di ingresso è "congelare" la porzione di memoria da proteggere verso programmi o istruzioni di cancellazione.

2.7 Il microcontrollore

2.7.1 Generalità

Il microcontrollore (MCU) è un dispositivo elettronico integrato nato come alternativa al microprocessore, rispetto al quale possiede prestazioni ridotte.
È ampiamente utilizzato in sistemi embedded ovvero per applicazioni specifiche (special purpose) di controllo digitale.
È progettato per interagire direttamente con periferiche "intelligenti" tramite un programma memorizzato internamente e mediante l'uso di pins specifici o impostabili dal programmatore. Sono disponibili in 3 fasce di ampiezza del bus dati: 8, 16 e 32 bit.
La vasta gamma di funzioni di comando disponibili, sia analogiche che digitali, permette l'impiego dei MCU in alternativa alle schede elettroniche tradizionali molto più costose e complicate.

2.7.2 Specifiche tecniche

L'MCU prescelto fa parte della famiglia **MC9S12C** della Freescale Semiconductor.
Questi dispositivi presentano:

- CPU a 16 bit;
- Circuito PLL;
- Port Integration Module PIM 9C32;
- Memoria Flash fino a 128 KB;
- Memoria RAM fino a 4 KB;
- Interfaccia di comunicazione seriale (SCI);
- Modulo Timer a 16 bit e 8 canali;
- Modulatore di ampiezza (PWM);
- Convertitore analogico/digitale a 10 bit.

In particolare il modello utilizzato in questa applicazione è il **GC32**: è un dispositivo provvisto di 80 pins, presenta una Flash EEPROM da 128 KB ed una memoria RAM da 4 KB. Può raggiungere una frequenza di bus massima pari a 25 MHz.
La presenza della memoria Flash appena citata è fondamentale in quanto essa dovrà contenere l'intero programma (in linguaggio C) nel momento in cui il sistema funzionerà in modo autonomo.
Questo rappresenta un pregio notevole poiché in tale fase non è più necessaria la presenza del PC.

L'immagine 2.15 raffigura lo schema a blocchi del microcontrollore.

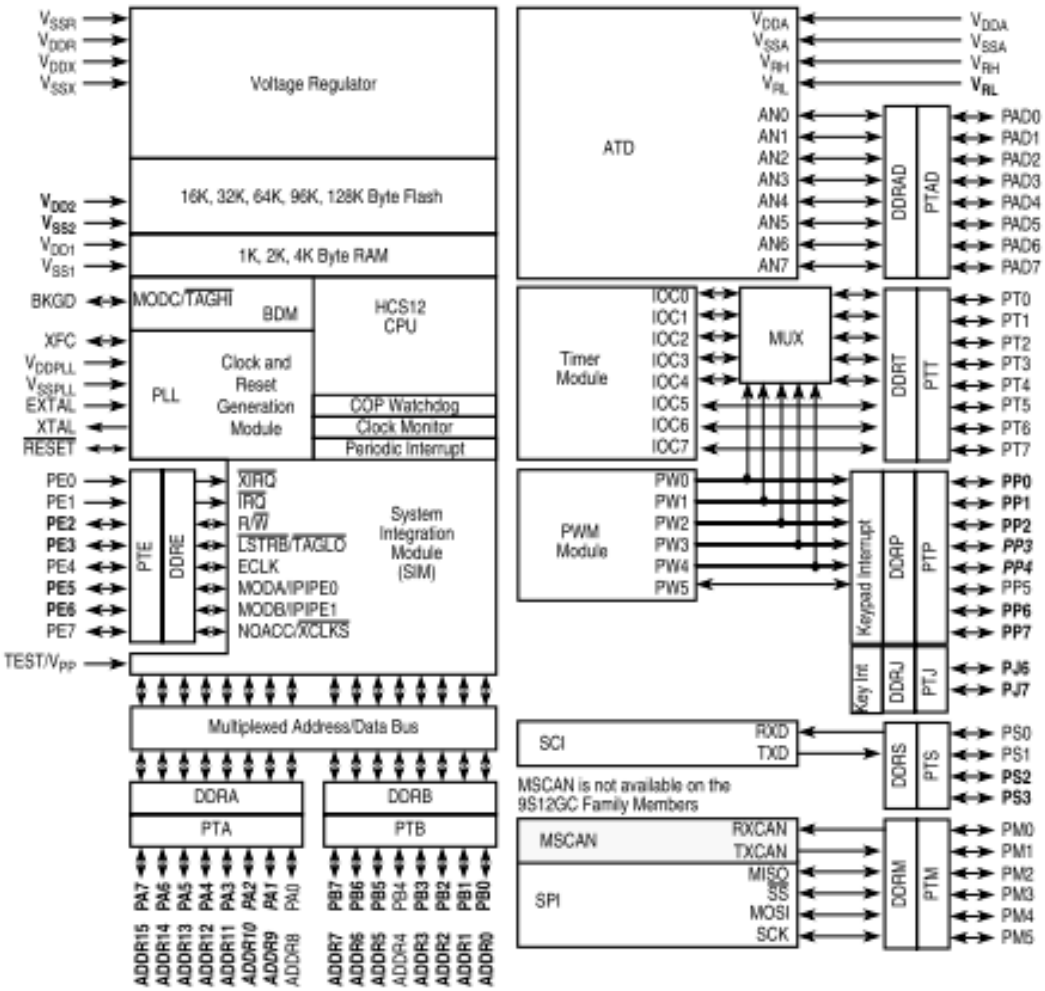


Fig. 2.15 – Schema a blocchi del microcontrollore

2.7.3 Il modulo PIM 9C32

Degli 80 piedini di cui è dotato il micro, 52 sono impiegabili come pins di Ingresso/Uscita e sono suddivisi in 8 gruppi chiamati porte. Le porte vengono denominate con una lettera (ad es. PORT A, PORT B, PORT J) ed i rispettivi pin da 0 a 7 vengono indicati con il nome di Pxy, dove x indica la lettera di riferimento della porta ed y il numero del pin a cui si fa riferimento.
Il PIM 9C32 stabilisce l'interfaccia tra i moduli presenti all'interno del microcontrollore ed i piedini di I/O di tutte le porte.
Nella figura successiva si osserva la disposizione delle porte nel microcontrollore.

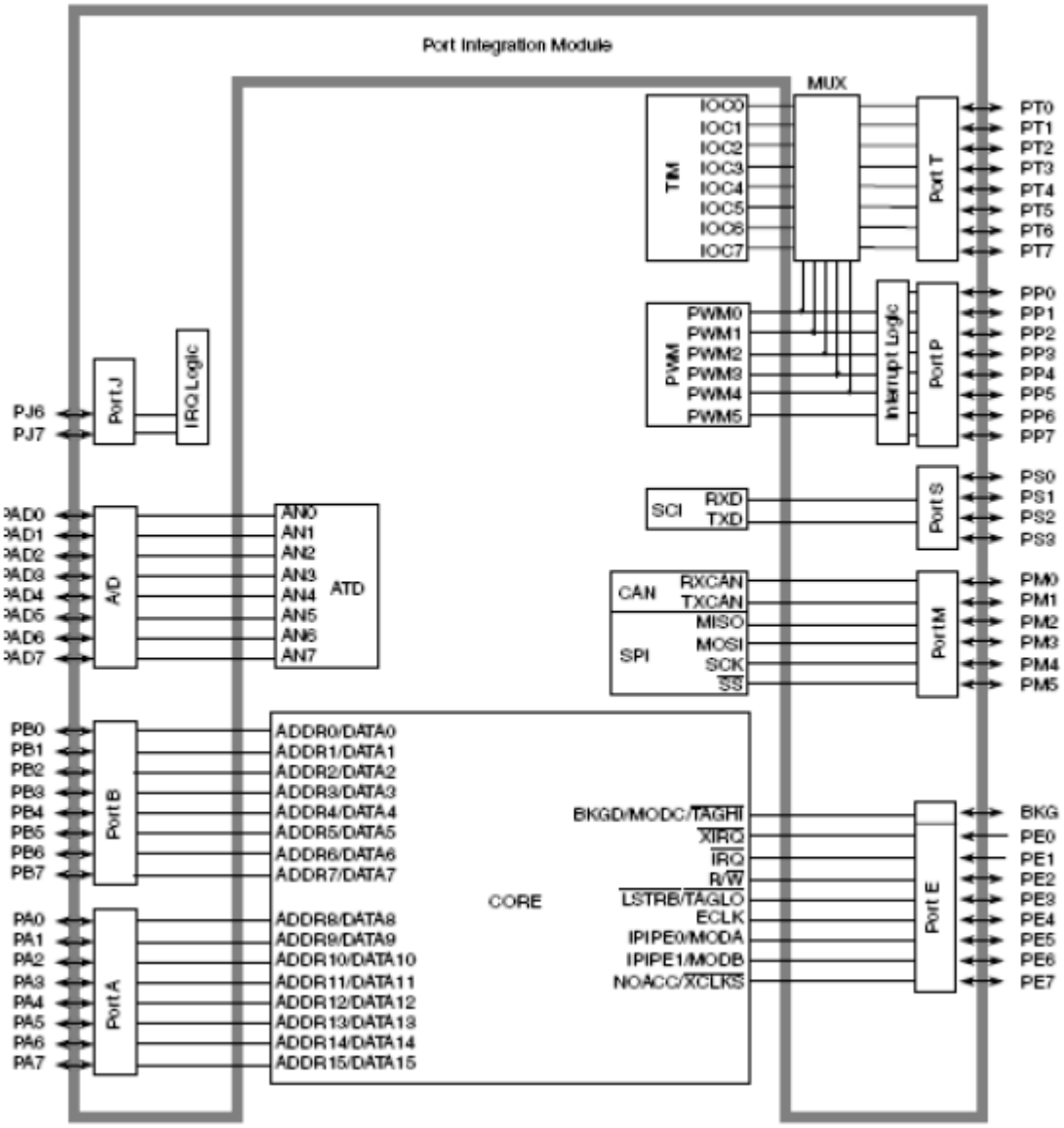


Fig. 2.16 – Disposizione delle porte del microcontrollore

Le porte A, B ed E si riferiscono all'interfaccia del multiplexer; la PORT T è connessa al modulo TIM (Timer); la PORT S è collegata all'interfaccia di comunicazione seriale (SCI); la PORT M è associata al modulo SPI; la PORT P è connessa al modulo PWM ed ai vari interrupt, la PORT J può essere utilizzata sia come interrupt esterno che come porta di I/O ed infine la PORT AD è associata al convertitore analogico/digitale.

Ogni porta possiede due registri che la definiscono:

- il registro PTx;
- il registro DDRx,

dove x indica la lettera della porta a cui il registro fa riferimento.

Nel registro PTx si scrivono in uscita i valori binari della porta mentre il registro DDRx definisce se i pin della porta vengono utilizzati come input o come output: un bit con valore logico 1 implica che la porta sia adoperata come output.

2.8 Il modulo Telit

Utilizzando le componenti descritte finora, è possibile visualizzare sullo schermo TFT alcune foto pre-caricate oppure disegnare caratteri ASCII sullo schermo. Se vogliamo realizzare l'obiettivo proposto, cioè mandare degli SMS, è necessario interfacciare il TFT con un modulo GSM, nel nostro caso il Telit **GM682 QUAD.**
Quindi il microcontrollore, oltre a gestire la FPGA ed il Touch Screen, deve coordinare la comunicazione con il modulo Telit. Essa viene effettuata tramite lo standard seriale RS232.

2.8.1 Il protocollo RS232 (cenni)

Lo standard EIA RS232 nacque negli anni '60 per opera della "Electronic Industries Association". Era predisposto alla comunicazione tra i computer ed i terminali (**D**ata **T**erminal **E**quipment) attraverso la linea telefonica, utilizzando un modem (**D**ata **C**ommunication **E**quipment).
Oggi la porta seriale EIA RS232 è presente in quasi tutti i PC, anche se sta cedendo il posto all'interfaccia USB in quasi tutti gli utilizzi.
L'interfaccia EIA RS232 ridotta (ovvero asincrona) utilizza un protocollo di trasmissione seriale di tipo asincrono.
"Seriale" implica che i bit che compongono l'informazione vengono trasmessi uno alla volta in un unico filo.
"Asincrono" significa che i dati sono trasmessi senza l'aggiunta di un segnale di clock. In questo caso viaggiano in pacchetti di 1 byte in modo anche non consecutivo. Ovviamente sia il trasmettitore che il ricevitore devono essere dotati di un proprio clock per poter scandire i dati. La sincronizzazione tra i due segnali di clock è necessaria ed avviene in corrispondenza alla prima transizione dei dati sulla linea.

La velocità della trasmissione si determina in base a quanti bit per secondo (bps) vengono trasferiti.
In questa applicazione il numero di bps è fissato a 9'600.

2.8.2 Introduzione al modulo Telit

Fig. 2.17 – Il Telit GM682 QUAD

Il modulo **Telit GM682 QUAD** funziona come un telefono cellulare: previo inserimento di una carta SIM, esso è in grado di inviare e ricevere SMS ed effettuare o rispondere a chiamate.
La peculiarità che lo distingue da un comune telefonino è il fatto che tutte le operazioni che esso può svolgere si comandano via software, seguendo la sintassi del **Protocollo at**.
La flessibilità del dispositivo è quindi molto elevata e predisposta al mondo della domotica.
Ritengo interessante introdurre il modulo Telit iniziando dalle sue caratteristiche principali.

Product Features

- Quad-band EGSM 850 / 900 / 1800 /1900 MHz
- Output Power
  Class 4 (2W) @ 850 / 900 MHz
  Class 1 (1W) @ 1800 / 1900 MHz
- Control via AT commands according to GSM 07.05, 07.07 and proprietary Telit
- Supply Voltage Range: 3,4–4,2 V DC (3,8 V DC recommended)
- Power Consumption (typical values)
  power off: 26 uA
  Idle (registered, power saving): ‹ 4 mA
  Dedicated mode: 170 mA
  GPRS cl.10 (max): 500 mA
- Serial Port Multiplexer GSM 7.10
- SIM Access Profile
- Sensitivity:
  -107 dBm (typ.) @ 850 / 900 MHz
  -106 dBm (typ.) @ 1800 / 1900 MHz
- Dimensions: 43,9 x 43,9 x 6.9 mm
- Weight: 19 gr.
- Extended Temp. range: -30° / +80° C
- RoHS compliant
- TCP/IP stack access via AT commands

Fig. 2.18 –Caratteristiche principali del Telit GM682 QUAD

Il Telit può comunicare con un computer tramite una porta seriale RS232. Il software che abbiamo utilizzato per instaurare le trasmissioni dati tra il modulo ed il PC è l'**Hyper Terminal**.
L'Hyper Terminal è un programma che funziona come un normalissimo editor di testo il quale si interfaccia con una porta di comunicazione esterna del PC (nel nostro caso la RS232), quindi esso è in grado di visualizzare anche le risposte del modulo in seguito all'invio (da tastiera) delle rispettive domande.
Nella pratica esso può esser visto come una "chat line" tra utente e telefono cellulare, ed è ideato per accettare solamente istruzioni dalla sintassi predefinita e consultabile negli appositi manuali. Quindi, per esempio, dal Terminal si può interrogare il Telit in merito a potenza e qualità della ricezione del segnale, o al nome dell'Operatore della SIM. È inoltre prevista la possibilità di accesso alla rubrica della carta SIM.
Oppure si possono sfruttare istruzioni avanzate quali l'invio o la ricezione di SMS.

2.8.3 Accensione del modulo

Per accendere il Telit si deve tenere premuto il pulsante ON# per almeno 1 secondo e poi rilasciarlo.
L'elettronica che si occupa di tale compito è rappresentata schematicamente in figura 2.19.

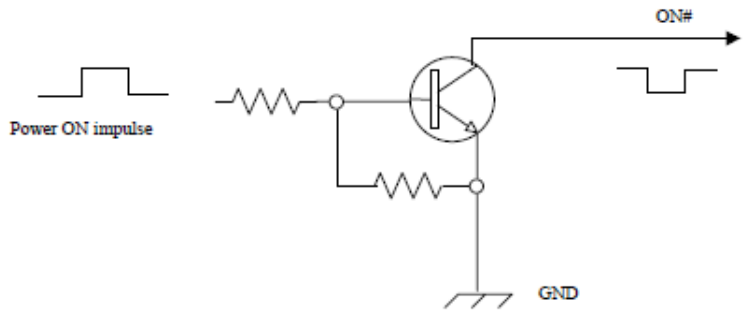


Fig. 2.19 – Circuito elettrico per l'accensione/spegnimento del modulo Telit

2.8.4 Spegnimento del modulo

Il dispositivo può essere spento in due modi:

- da comando software (Terminal);
- tramite pulsante.

Quando il sistema viene spento, esso invia in rete la richiesta di disconnessione, informandola che il telefono non è più raggiungibile.

Spegnimento via software

- Digitare il comando at#shdn<cr>
- Attendere l'OK di risposta.

Il dispositivo si arresta seguendo questi passi:

- distaccamento dalla rete;
- spegnimento del modulo.

Nel caso di assenza di campo, la disattivazione della rete richiede tipicamente 6 secondi.

Spegnimento via hardware

È sufficiente tenere premuto il pulsante ON# per almeno 2 secondi e poi rilasciarlo.
Si utilizza lo stesso circuito visto per l'accensione.

2.9 L'alimentazione del sistema

2.9.1 Generalità

Il modulo TFT, da specifica, deve funzionare da una tensione minima di 6 V ad una tensione massima di 20 V. Le tensioni che devono alimentare i dispositivi presenti all'interno del modulo TFT sono le seguenti:

- 1,2 V: FPGA;
- 3,3 V: FPGA, Touch (quindi la parte "sensibile" dello schermo), microcontrollore, SDRAM, Flash, modulo Telit;
- 2,5 e 5 V: display LCD (rispettivamente VCC e AVDD, fig. 2.25);
- 20 V: back light.

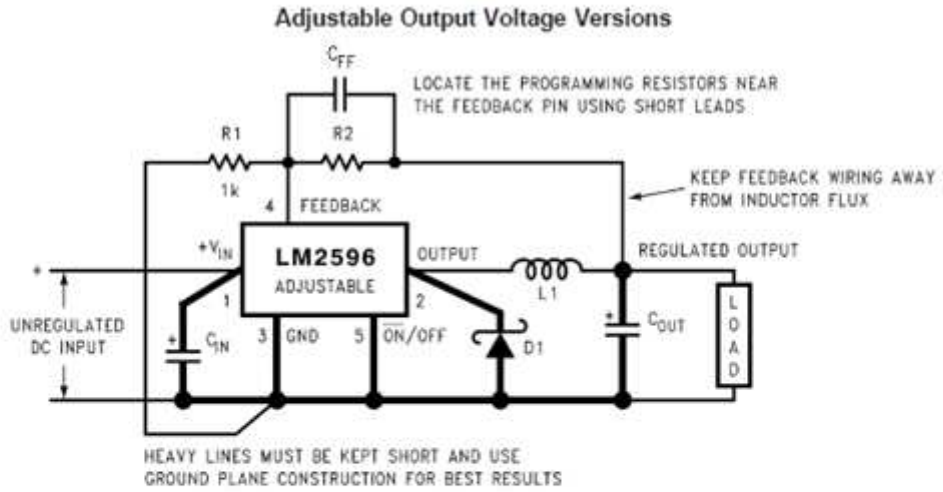
Si è quindi progettato un sistema di alimentazione con le caratteristiche sopra citate.
Si è provveduto a convertire la tensione di ingresso (che può variare da 6 a 20 V) in una tensione continua stabilizzata a 5,5 V con una corrente massima di 1 A.
Il dispositivo utilizzato è l'integrato **LM2596S** della National Semiconductor, uno step-down converter.
Per ottenere 1,2 V, 3,3 V e 5 V si è adoperato un dispositivo regolatore di tensione lineare: il **TPS 73701** della Texas Instruments**.**
La tensione di 20 V è stata ottenuta con l'impiego di un **LM27313**, della National Semiconductor, un boost converter.

2.9.2 Il convertitore step-down o buck converter

Le caratteristiche tecniche della serie **LM2596** sono riportate nell'elenco seguente.

- Tensioni di uscita fisse a 3,3 V, 5 V e 12 V e versioni con uscita regolabile.
- Tensioni di uscita comprese tra 1,2 V e 37 V nel caso regolabile.
- Capacità di pilotare una corrente di carico fino a 3 A.
- Tensione in ingresso ammessa fino a 40 V.
- Richiesta di sole 4 componenti esterne per il dimensionamento.
- Eccellente regolazione del carico.
- Frequenza dell'oscillatore interno fissata a 150 KHz.
- Consumo di corrente in modalità standby pari a 80 uA.
- Rendimento elevato.
- Utilizzo di induttori standard facilmente reperibili.
- Limitatore di corrente.

In particolare, la versione **S** permette la regolazione della tensione di uscita.



$$V_{OUT} = V_{REF}\left(1 + \frac{R_2}{R_1}\right) \qquad \text{where } V_{REF} = 1.23V \qquad R_2 = R_1\left(\frac{V_{OUT}}{V_{REF}} - 1\right)$$

Select R_1 to be approximately 1 kΩ, use a 1% resistor for best stability.
C_{IN} —470 µF, 50V, Aluminum Electrolytic Nichicon "PL Series"
C_{OUT} —220 µF, 35V Aluminum Electrolytic, Nichicon "PL Series"
D1 —5A, 40V Schottky Rectifier, 1N5825
L1 —68 µH, L38
R1 —1 kΩ, 1%

Fig. 2.20 – Il convertitore step-down LM2596S, con alcune componenti pre-calcolate

<u>Dimensionamento delle componenti passive rimanenti</u>

In base alla formula di V_{OUT} in figura, R_2 va dimensionata in modo da ottenere 5,5 V in uscita. È stata adoperata una resistenza di precisione di valore 3'470Ω. Infatti:

- $V_{OUT} = 1{,}23 * (1 + 3'470/1'000) = 5{,}4981 \approx 5{,}5$ V.

L'induttanza L1 si calcola a partire dalla legge di Lenz:

$$E \bullet T = (V_{IN} - V_{OUT} - V_{SAT}) \bullet \frac{V_{OUT} + V_D}{V_{IN} - V_{SAT} + V_D} \bullet \frac{1000}{150\ kHz}\ (V \bullet \mu s)$$

Dove:

- V_{SAT} = tensione di saturazione degli switch interni = 1,16 V;
- V_D = caduta di tensione sul diodo = 0,5 V.

Assumendo V_{IN} = 12 V e V_{OUT} = 5,5 V, si ottiene:

- E*T = 18,83 V*us.

Tramite il grafico seguente si effettua la scelta dell'induttanza. Poiché si vuole limitare la corrente massima nel circuito ad 1 A, si cerca l'intersezione dei 18,83 V*us in ordinata con la rispettiva ascissa.
L'induttanza indicata vale quindi 68 uH.

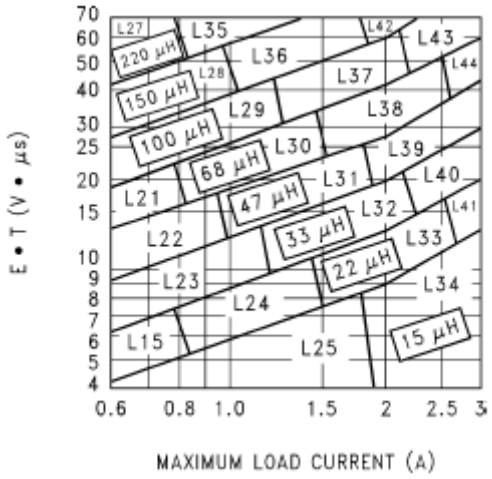


Fig. 2.21 – Calcolo dell'induttanza L1 opportuna

La capacità di compensazione C_{FF} (feedforward) è legata al valore di R_2:

$$C_{FF} = \frac{1}{31 \times 10^3 \times R_2}$$

Poiché il calcolo da' come risultato 9,3 nF, il valore di C_{FF} adoperato è pari a 10 nF.

Piedinatura dell'integrato

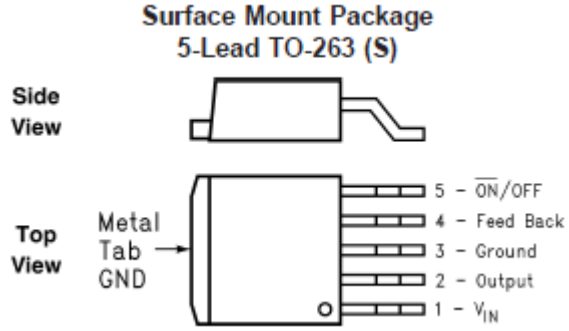


Fig. 2.22 – Piedinatura dell'LM2596S

Il piedino "Feed Back" preleva la tensione regolata in uscita per completare la retroazione.

2.9.3 Il convertitore boost

Per alimentare la back light del TFT, composta da 7 diodi LED in serie tra di loro, si necessita di una tensione di 20 V e di una corrente pari a 23 mA. È stato quindi utilizzato un generatore di corrente costante.
La retroazione in corrente avviene tramite la resistenza Rs da 56 Ω (fig. 2.23) che è direttamente collegata al piedino di Feedback. Quindi, un qualsiasi cambiamento della corrente dovuto a variazioni della tensione di alimentazione, al riscaldamento o all'invecchiamento dei diodi LED viene compensato dalla retroazione.
Per provvedere a questa alimentazione si è utilizzato un convertitore boost in grado di elevare la tensione da 5,5 V a 30 V massimi, ed imporre una corrente costante di 23 mA.
Il convertitore DC/DC scelto per questa applicazione è l'**LM27313**.

Esso opera ad una frequenza fissa di funzionamento pari a 1,6 MHz ed accetta tensioni di ingresso comprese tra 2,7 V e 14 V. Il sistema switch è particolarmente indicato per incrementare tensioni da 5 V fino a 28 V.
Il dimensionamento del circuito è raffigurato nell'immagine seguente.

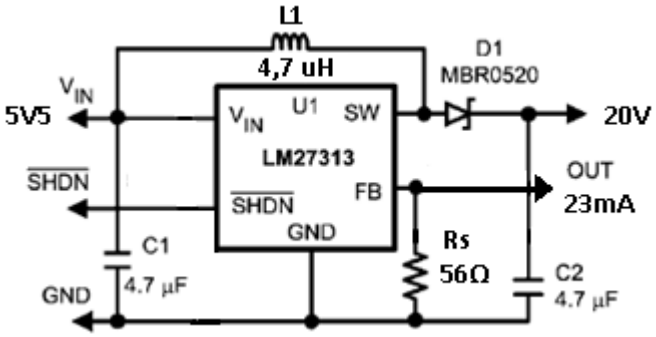


Fig. 2.23 – Il convertitore boost LM27313

Piedinatura e descrizione dell'integrato

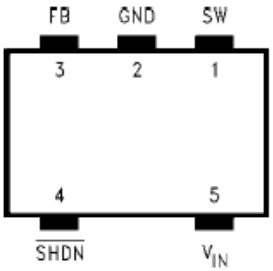


Fig. 2.24 – Piedinatura dell'LM27313

Piedino	Nome	Funzione
1	SW	Drain del FET interno
2	GND	Massa
3	FB	Punto di retroazione che connette a partitori di tensione esterni per settare Vout
4	$\overline{SHDN}$	Chiusura del controllo in ingresso. Connettere a Vin se non viene utilizzato
5	V_{IN}	Alimentazione

Il piedino "Shutdown" (4), se posto a stato logico basso, provoca l'interdizione del dispositivo.
Il pin "Feedback" (3) fornisce una tensione di riferimento tipica pari a 1,23 V.

Caratteristiche di funzionamento

Temperatura di utilizzo	-65 ÷ +150° C
Dissipazione di potenza	Limitata internamente
Tensione sul pin FB	-0,4 ÷ +6 V
Tensione sul pin SW	-0,4 ÷ +30 V
V_{IN}	2,7 ÷ 14 V
$V_{SW(MAX)}$	30 V
V_{SHDN}	0 ÷ V_{IN}
Temperatura alla giunzione	40 ÷ 125 °C

Duty cycle
Il massimo duty cycle (D.C.) dell'onda quadra a frequenza 1,6 MHz determina il massimo rapporto V_{OUT}/V_{IN} che il convertitore è in grado di raggiungere: quindi, il guadagno in tensione del circuito si imposta tramite il D.C. Per una data applicazione con il convertitore boost, il duty cycle si definisce con la formula seguente:

$$\text{Duty Cycle} = \frac{V_{OUT} + V_{DIODE} - V_{IN}}{V_{OUT} + V_{DIODE} - V_{SW}}$$

La frequenza tipica di commutazione dell' LM27313 è pari a 1,6 MHz, quindi il periodo di commutazione è uguale a 0,625 us.
Assumendo: V_{IN} = 5,5 V, V_{OUT} = 20 V, V_{DIODE} = 0,3 V, V_{SW} = 0,2 V, il duty cycle vale:

- D.C. = (20 + 0,3 – 5,5) / (20 + 0,3 – 0,2) = 73,6%.

Il periodo di tempo tipico in cui l'interruttore è ON ha durata:

- T_{ON} = (0,736 * 0,625) = 0,460 us.

Dimensionamento delle componenti passive
Il condensatore di ingresso C1 e quello di uscita C2 hanno lo scopo di assorbire i picchi di corrente che avvengono durante le commutazioni degli switch interni. In particolare, un valore sufficientemente elevato di C2 consente la stabilizzazione della tensione di uscita. Per la maggior parte delle applicazioni con l'LM27313 si consiglia per entrambi un valore compreso tra 4,7 e 10 uF.
In questo caso abbiamo scelto C1 = C2 = 4,7 uF.
La resistenza di retroazione Rs serve a fornire la corrente necessaria per l'alimentazione del sistema back light. Essa è interposta tra il pin 3 (FB) e GND. Poiché il piedino "Feedback" fornisce una tensione di riferimento tipica pari a 1,23 V, per ottenere i 23 mA di corrente necessari al sistema back light si calcola:

- Rs = V_{FB}/I_{BL} = 1,23 / 0,023 = 53,48 Ω.

Il valore commerciale più vicino a quello teorico è Rs = 56 Ω.
L'induttanza L1 si calcola tenendo conto che la massima corrente che l'integrato è in grado di sopportare è pari a 800 mA. L1 si ottiene con i seguenti passaggi:

- Si assumono, come per il calcolo del D.C., V_{IN} = 5,5 V, V_{OUT} = 20 V, V_{DIODE} = 0,3 V, V_{SW} = 0,2 V.

La frequenza nominale con cui il dispositivo funziona è 1,6 MHz. Quella minima è 1,15 MHz. Pertanto:

- $T_{ciclo(max)} = 1 / (1{,}15 * 10^6) = 0{,}87$ us.
- D.C. (dai calcoli precedenti) = 0,736.

Il tempo massimo in cui gli switch si trovano allo stato ON è:

- $T_{ON(max)}$ = 0,736 * 0,87 = 0,64 us.

Durante la fase T_{ON} la tensione ai capi di L1 è:

- V_{L1} = 5,5 – 0,2 = 5,3 V.

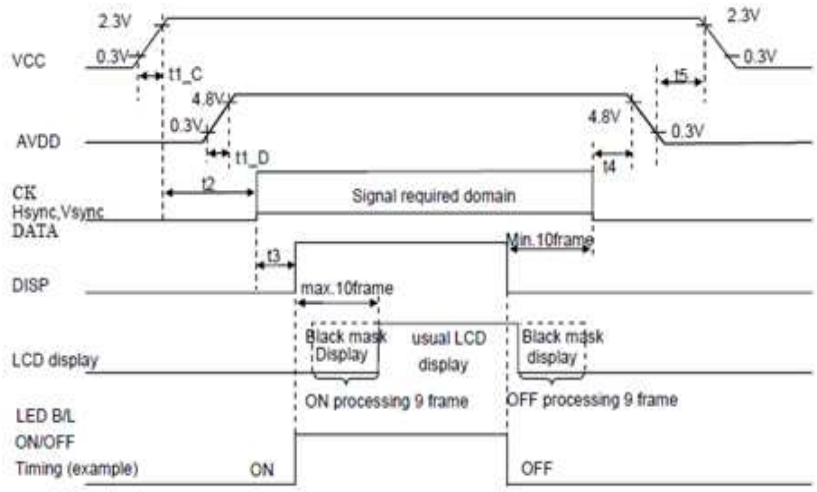
Il valore minimo di induttanza si ottiene a partire dalla legge di Lenz:

- L1 = V_{L1} * (dt/di_{MAX}) = 5,3 * (0,64 / 0.8) = 4,24 uH.

Per limitare ulteriormente i picchi di corrente abbiamo assunto un valore di poco maggiore, L1 = 4,7 uH.

2.9.4 L'alimentazione del TFT

Per alimentare il TFT servono due valori di tensione: 2,5 V (VCC) e 5 V (AVDD). Queste tensioni, ottenute da due circuiti distinti, devono essere fornite al display seguendo delle tempistiche prestabilite.
La sequenza delle alimentazioni e dei vari segnali che interagiscono nel display è raffigurata nell'immagine 2.25.



Legenda tempistiche

	MIN	TYP	MAX	unit
t1_C	0	–	10	ms
t1_D	0.5	–	10	ms
t2	50	–	–	ms
t3	0.5	–	–	ms
t4	0	–	–	ms
t5	0	–	–	ms

Fig. 2.25 – Segnali logici nel TFT

Il regolatore di tensione prescelto è il **TPS73701** della Texas Instrument. È un dispositivo con guadagno di tensione regolabile. Tale integrato è provvisto di un piedino di Enable (abilitazione), con il quale il microcontrollore gestisce le tempistiche di alimentazione appena descritte.

Le caratteristiche salienti del dispositivo sono:

- stabilità della tensione in uscita (grazie al condensatore in uscita C2 da 4,7 uF);
- range della tensione di ingresso: 2,2 ÷ 5,5 V;
- corrente inferiore a 20 nA quando non abilitato (shutdown);
- limitazione termica e sulla corrente massima;
- tensione di riferimento interna pari a 1,204 V;
- range di tensione di uscita compreso tra 1,2 e 5,5 V.

Il circuito completo è raffigurato nell'immagine seguente.

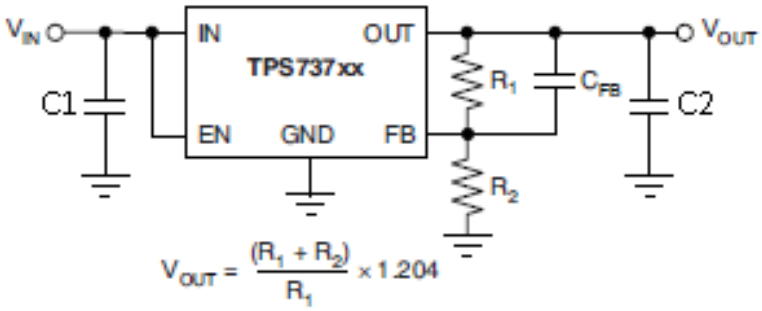
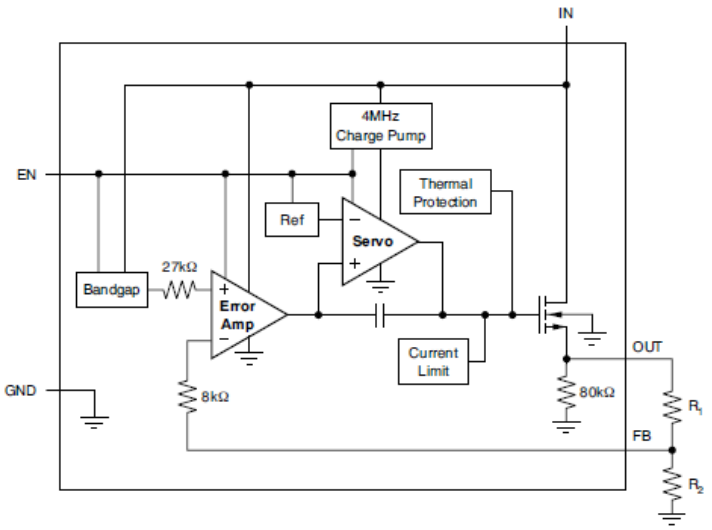


Fig. 2.26 – Il regolatore di tensione TPS73701

Schema interno e dimensionamento componenti



Standard 1% Resistor Values for Common Output Voltages

V_O	R_1	R_2
1.2V	Short	Open
1.5V	23.2kΩ	95.3kΩ
1.8V	28.0kΩ	56.2kΩ
2.5V	39.2kΩ	36.5kΩ
2.8V	44.2kΩ	33.2kΩ
3.0V	46.4kΩ	30.9kΩ
3.3V	52.3kΩ	30.1kΩ

NOTE: $V_{OUT} = (R_1 + R_2)/R_2 \times 1.204$; $R_1 \parallel R_2 = 19k\Omega$ for best accuracy.

Fig. 2.27 – Schema interno del TPS73701 e dimensionamento componenti in base alle tensioni di uscita

I valori delle due resistenze di precisione R_1 e R_2 possono essere calcolati con la formula sopra riportata per qualsiasi tensione in uscita, purché compresa nel range consentito.
Per una maggiore accuratezza, si raccomanda che il parallelo tra R_1 ed R_2 sia all'incirca 19 KΩ. Questo valore, sommato agli 8 KΩ del resistore interno, è in grado di compensare gli errori dovuti al disadattamento di impedenza.
Infatti, la resistenza in ingresso al piedino non invertente del primo stadio operazionale è pari a 27 KΩ.
Il pin invertente "vede" al proprio ingresso il parallelo tra R_1 ed R_2 in serie agli 8 KΩ. Quindi, se la relazione tra R_1 ed R_2 rispetta quanto consigliato, si è in presenza di carico adattato.
Le combinazioni tra R_1 ed R_2 tabulate sono state calcolate assumendo che l'amplificatore operazionale (Error Amp) sia ideale. In questo caso, la relazione tra le due resistenze è data da un semplice partitore di tensione (figura 2.28).

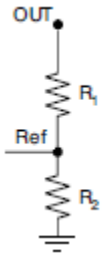


Fig. 2.28 – Dimensionamento rapido di R_1 ed R_2

Il riferimento è fisso e pari a 1,204 V. R_1 ed R_2 sono funzione esclusivamente della V_{OUT} che si desidera ottenere. Quindi:

- 5V5 to 2V5: R_1 = 36k5, R_2 = 33k2;
- 5V5 to 5V: R_1 = 86k6, R_2 = 27k4.

Nonostante il condensatore di ingresso C1 non sia indispensabile ai fini della stabilità del sistema, la sua presenza è consigliata per contrastare le induttanze parassite, migliorare la risposta ai transitori e la reiezione al rumore.
Il condensatore C1 impiegato in entrambi i TPS 73701 è pari a 10 uF.
C2 si utilizza per rendere stabile la tensione in uscita. Le specifiche consigliano un valore abbastanza elevato. Per la nostra applicazione abbiamo selezionato C2 = 4,7 uF.

Il condensatore di Feedback C_{FB} serve a ridurre il rumore in uscita, si consiglia di non superare i 100 nF. Per entrambi i TPS 73701, è stato scelto un valore di C_{FB} pari a 100 nF.

Piedinatura dell'integrato

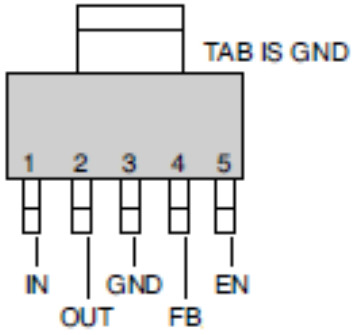


Fig. 2.29 – Piedinatura del TPS73701

Nome	**Pin**	**Descrizione**
IN	1	Ingresso
GND	3	Ground
EN	5	Se=1, accende il regolatore. Se=0, lo porta in shutdown
FB	4	Ingresso dell'anello di controllo. Serve a settare la VOUT
OUT	2	Uscita

2.9.5 Alimentazione di FPGA, SDRAM, Flash, MCU e modulo Telit

Anche in questo caso si adoperano due regolatori **TPS 73701.** Il dimensionamento delle varie capacità è già stato completamente definito. Cambiano solo i valori di R_1 ed R_2.

- 5V5 to 1V2: si utilizza direttamente il riferimento interno, le due resistenze non servono (precisamente, R_1 è un cortocircuito e R_2 un circuito aperto);
- 5V5 to 3V3: R_1 = 52k3, R_2 = 30k1.

CAPITOLO 3

IL SOFTWARE

Il software di questo sistema comprende le seguenti fasi:

- Programmazione del microcontrollore in linguaggio C;
- Programmazione dell'FPGA in linguaggio VHDL;
- Conversione delle foto da formato bmp a bit (tramite il software BmpTOBit per PC);
- Identificazione dei pulsanti digitati su tastiera del TFT;
- Scrittura su display del carattere digitato;
- Studio del Protocollo at del modulo Telit, utilizzando l'Hyper Terminal;
- Inserimento del Protocollo at nel codice sorgente C.

3.1 L'ambiente di sviluppo CodeWarrior

3.1.1 Introduzione

Per la programmazione del microcontrollore, si è utilizzato il software **CodeWarrior Development Studio** della Freescale Semiconductor. E' un ambiente di sviluppo integrato completo (IDE) che fornisce una interfaccia panoramica e automatizzata per velocizzare lo sviluppo di applicazioni che utilizzano microprocessori della Freescale.
Una delle caratteristiche di maggior pregio di questo ambiente è il suo supporto ai sistemi operativi più utilizzati dai programmatori, come Windows, Linux e Mac. L'IDE è incentrata su compilatori C e C++ e comprende tutti gli strumenti necessari per completare un software integrato: editor di testo, compilatori, simulatori, debugger, ecc.

3.1.2 La programmazione

Dopo il corretto completamento della fase di wizard, CodeWarrior visualizza una Project Window sul lato sinistro dell'ambiente ed un editor di testo a destra. Nella Project Window (fig. 3.1) si riconoscono i file di programmazione del micro e le librerie pre-caricate in automatico.
Infatti, tutti i registri sono definiti nel file MC9S12GC32.h: è sufficiente includere il file all'inizio della pagina del main per avere a disposizione tutti i registri e le porte settate e pronte all'utilizzo.

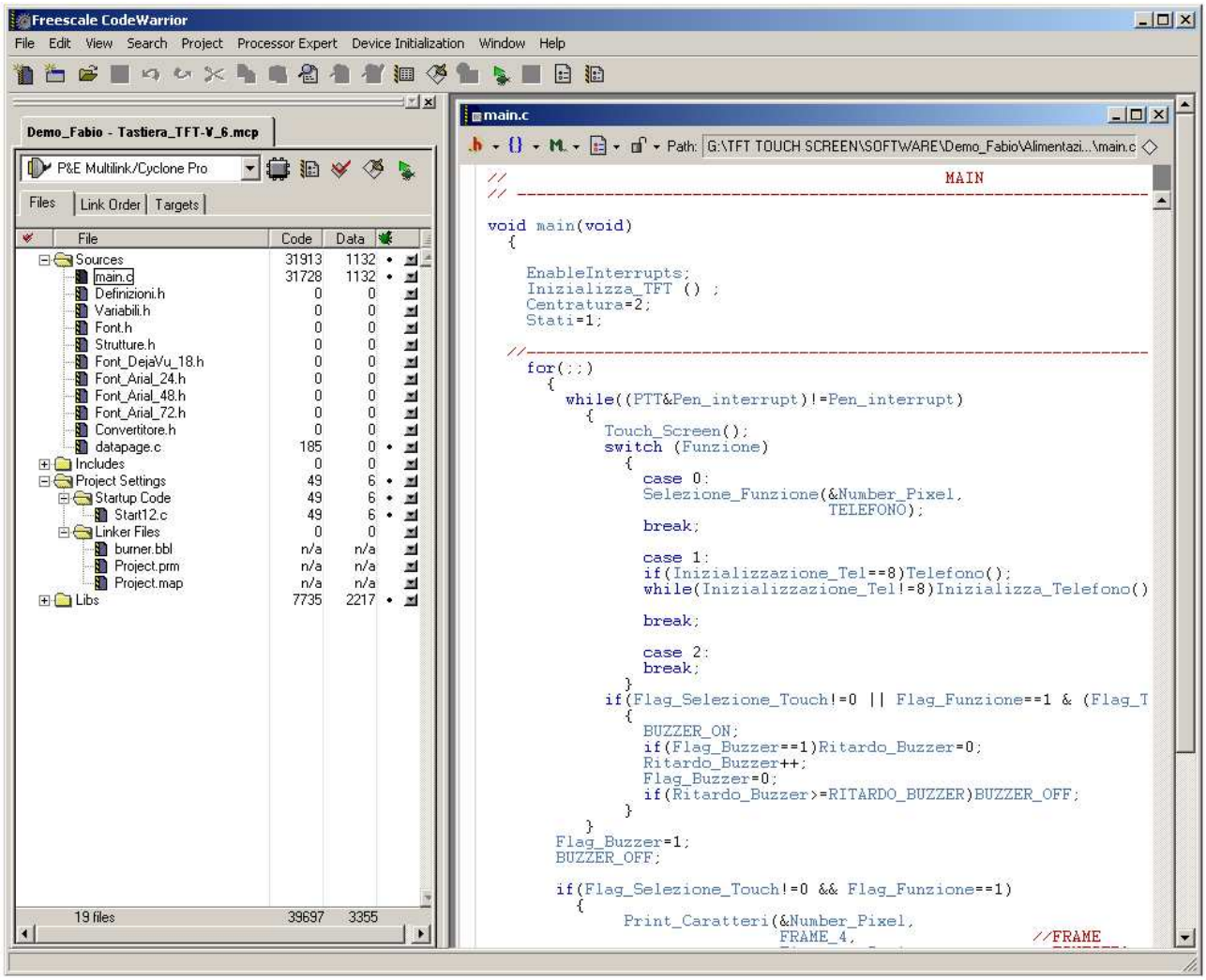


Fig. 3.1 – L'IDE di CodeWarrior

3.1.3 Il debug

Tramite questo strumento è possibile scaricare il programma precedentemente compilato all'interno del micro. Il download del programma avviene attraverso un apposito emulatore della **P&E Multilink** che ha il compito di interfacciare il programma CodeWarrior con il microcontrollore.
Una volta completata la procedura appare la schermata di debug.
Come si nota dalla figura 3.2, l'interfaccia è suddivisa in tre parti: a sinistra vi è il codice precedentemente compilato, a destra sono presenti tutte le variabili definite nel main ed il valore che esse assumono in tempo reale.
È possibile controllare passo per passo gli effetti di tutte le istruzioni per comprendere il motivo di un funzionamento indesiderato oppure risalire al punto nel quale il microcontrollore si arresta. Nel debug è possibile inserire dei break-point con i quali bloccare temporaneamente il codice. Inoltre si possono monitorare i valori assunti da tutte le variabili in ogni istante.

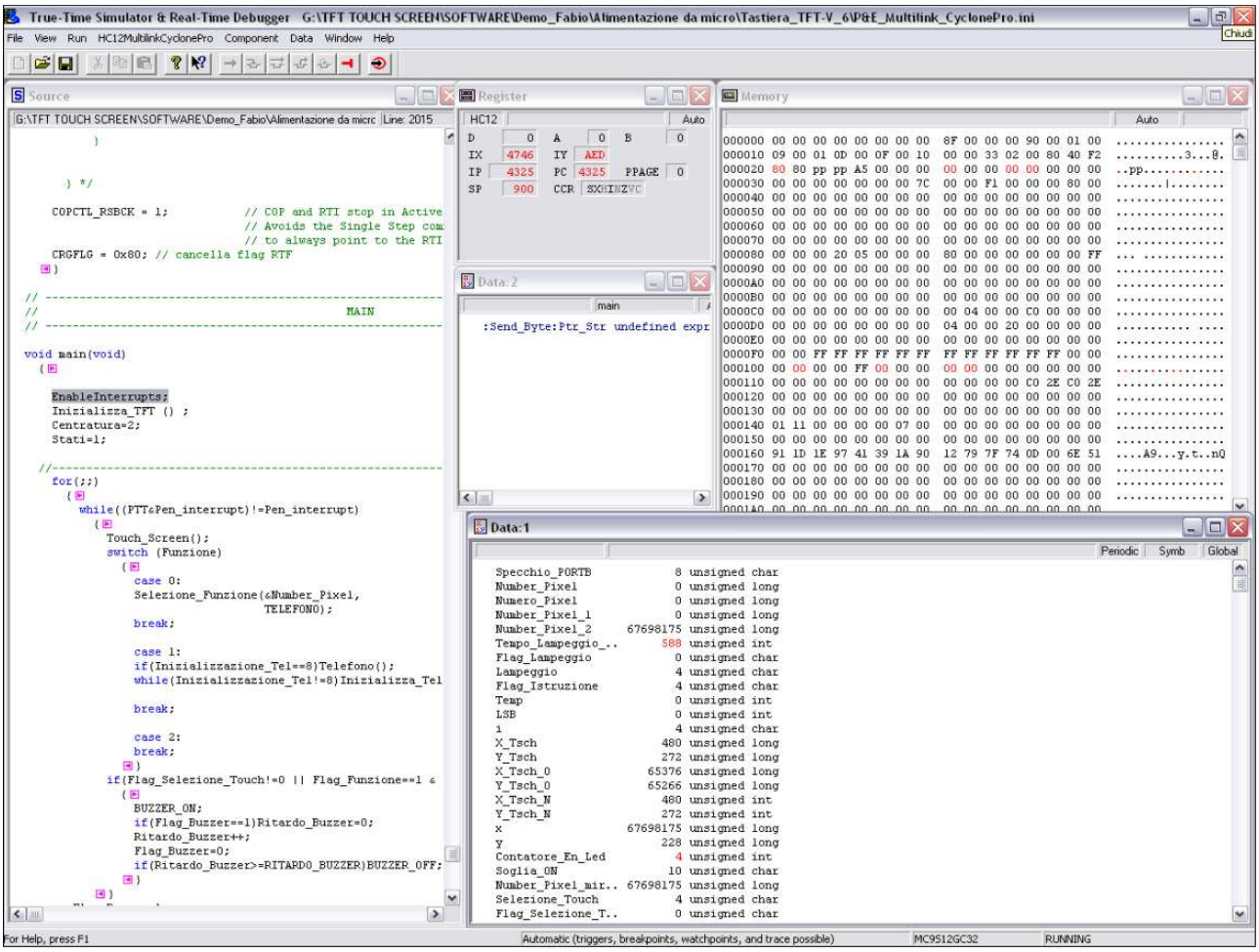


Fig. 3.2 – La schermata di Debug

3.2 La programmazione dell'FPGA in linguaggio VHDL

3.2.1 Il linguaggio VHDL: cenni

Gli HDL sono linguaggi di programmazione finalizzati a configurare l'hardware (non il software) di dispositivi programmabili. Per questo motivo presentano sostanziali differenze rispetto a linguaggi di programmazione tradizionali, come il C.
In particolare, vi sono due caratteristiche proprie degli HDL che sottolineano tale distinzione.

- **Concorrenza:** capacità di compiere più operazioni contemporaneamente (tipico dei dispositivi hardware). È previsto un sistema di esecuzione in parallelo;
- **Timing:** facoltà di gestire i tempi di propagazione dei segnali all'interno delle porte logiche nei circuiti digitali.

Poiché il risultato dell'elaborazione deve essere indipendente dalla sequenza con la quale le istruzioni sono state eseguite, tutte le elaborazioni devono essere effettuate in parallelo senza che un'istruzione, per essere eseguita, debba attendere il completamento di un'altra.

Entità definite in VHDL

Nel linguaggio VHDL sono disponibili entità (oggetti) alle quali è associato un tipo di dato ed un valore.
Le principali sono tre:

- Costanti;
- Variabili;
- Segnali.

Costanti: si tratta di entità che non possono modificare il proprio valore. Esattamente come nel linguaggio C, si utilizzano per migliorare la leggibilità del codice. La sintassi da adoperare è la seguente:

constant nome: **tipo** := valore;

Esempio: **constant** delay_interrupt: **time** := 20 ms;

Variabili: sono oggetti che possono cambiare il loro valore. È in uso la seguente sintassi:

variable nome: **tipo** := valore_iniziale;

Esempio: **variable** proposizione: **boolean** := false;

Le variabili non hanno valenza hardware, ovvero non significano nulla dal punto di vista elettrico.
Sono utilizzate per memorizzare valori temporanei.

Segnali: sono entità che possono cambiare il loro valore. Si definiscono in questo modo:

signal nome: **tipo**:= 'valore_iniziale';

Esempio: **signal** CS: **bit** := '0';

I segnali, diversamente dalle variabili, hanno un significato fisico ben preciso. Essi permettono di implementare l'hardware. Inoltre, ad ogni segnale è associata un'evoluzione nel tempo.

Tipi di dato predefiniti in VHDL

L'ambiente VHDL predefinisce i seguenti tipi di dato:

- integer;
- boolean ('TRUE', 'FALSE');
- bit ('0', '1');
- time;
- positive;
- natural.

Oltre a questi è possibile impiegarne di ulteriori, includendo librerie specifiche. È anche permesso crearne di nuovi tipi tramite l'istruzione **type**.

La programmazione dell'FPGA in linguaggio VHDL avviene utilizzando la connessione JTAG.

3.2.2 Il protocollo JTAG

La JTAG, acronimo di Joint Test Action Group, è un'associazione di circa 200 imprese costruttrici di circuiti integrati e stampati con lo scopo di definire un protocollo standard per testare i propri dispositivi, che col passare del tempo sono sempre più difficili da controllare.
Effettivamente, per quanto riguarda le schede elettroniche, il futuro che si profila è il seguente:

- Circuiti integrati sempre più piccoli e con numero di pin in continuo aumento;
- Circuiti stampati multistrato: sarà sempre più complesso accedere alle connessioni tra le componenti elettroniche sulle schede.

La soluzione proposta dal gruppo è stata quella di prevedere la possibilità, per alcuni pin degli integrati, di bloccare il funzionamento normale degli stessi e di portarsi ad una particolare configurazione che permetta il controllo di tutti i pin rimanenti.

Come già evidenziato, in fase di progettazione non utilizziamo la JTAG per i test dei dispositivi, bensì per programmare l'FPGA. La connessione JTAG di cui disponiamo prevede 6 pins, associati ad un preciso segnale. In altre configurazioni si può arrivare a 7, quando è presente un segnale di reset del test.
La piedinatura è la seguente:

- **TCK** (Test Clock): pin di Clock;
- **TMS** (Test Mode Select): selezione della modalità di test;
- **TDI** (Test Data In): piedino di ingresso dati della scheda elettronica. Questo segnale collega in catena i vari circuiti integrati;
- **TDO** (Test Data Out): pin di uscita dei dati;
- **VCC:** alimentazione, in questo caso 3,3 V;
- **GND:** massa.

Quando si impiega il protocollo JTAG per il test dei dispositivi, i circuiti integrati sono "percorsi" in modo seriale in catena (Daisy chain), poiché i segnali TDI e TDO trasferiscono i dati dall'ingresso della scheda tra un circuito integrato ed il successivo e vengono infine letti sul pin TDO in uscita. La trasmissione seriale è sincrona in quanto controllata da un segnale di Clock (TCK). La frequenza di Clock è determinata dal dispositivo più lento della catena. Il segnale TMS avvisa i circuiti integrati di sospendere il loro funzionamento normale e di riconoscere la modalità di test.

3.3 Come visualizzare un'immagine sul TFT

Per visualizzare su display un'immagine o una sequenza di figure sono necessari i seguenti passi:

1) Conversione del formato dell'immagine da bmp a bit;
2) *Merge* di tutte le foto *.bit ed il programma di boot dell'FPGA (creazione di un file *.mcs);
3) Download del file *.mcs utilizzando la connessione JTAG da PC a FPGA e contemporaneo trasferimento dei dati dall'FPGA alla Flash tramite protocollo SPI.

1) Le immagini salvate nei computer possiedono tipicamente i formati jpeg e bmp. L'FPGA, però, non accetta i file dotati di questa estensione. Per prima cosa è necessario adattare la risoluzione della foto ai pixel del display: è sufficiente usare un programma grafico che possiede la funzione *copy screen*, assegnando all'area di selezione il valore di 480*272 pixel. L'immagine va poi salvata con estensione bmp.
   Nel file *.bmp, i primi 54 byte contengono delle informazioni riguardanti il file stesso. I successivi 3 byte rappresentano il 1° pixel della foto, situato in basso a sinistra (in figura 3.3, è di colore azzurro). Gli ultimi 3 byte del file, invece, definiscono il pixel in alto a destra (fucsia). Per il TFT, come si è visto dal suo protocollo di interfacciamento, i primi 3 byte del file rappresentano il 1° pixel, collocato in alto a sinistra (rosso), mentre i 3 byte finali definiscono l'ultimo pixel (verde).

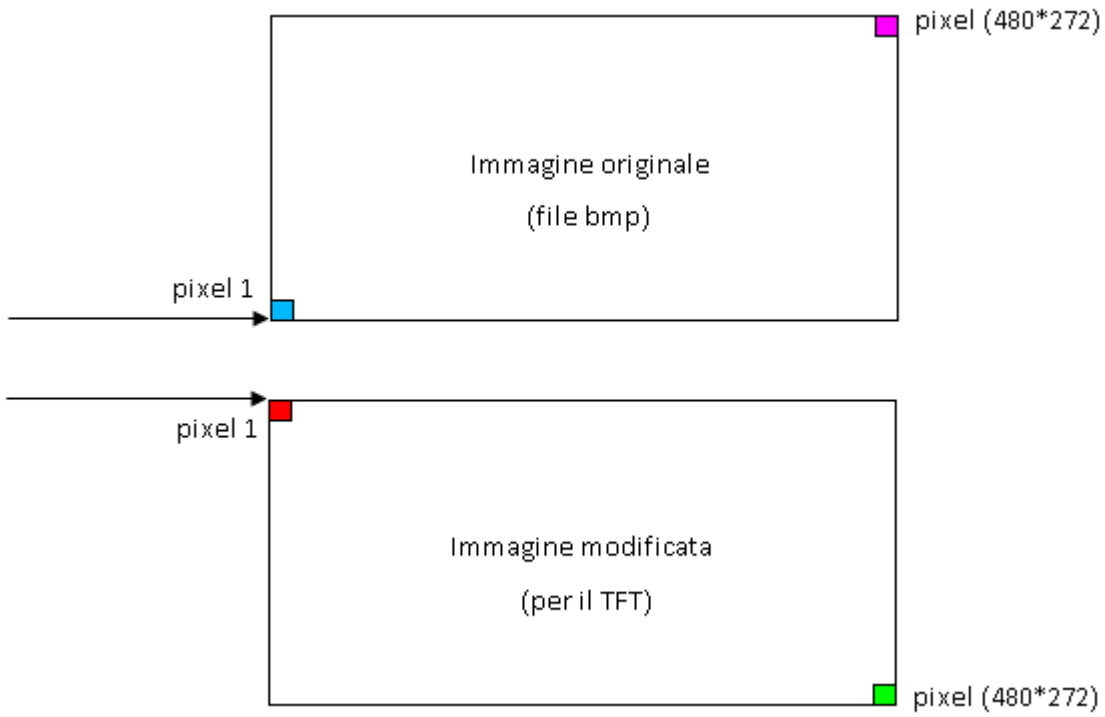


Fig. 3.3 – Distinzione tra il 1° pixel del file bmp ed il 1°pixel del protocollo di interfacciamento del TFT

Si ricorda che un pixel LCD è composto dalla "fusione" dei colori R, G, B, ciascuno dei quali è definito a propria volta dalla combinazione di 8 bit, quindi da 256 possibili valori. Si realizza che:

- 256*256*256 = 16'777'216 colori possibili.

Un pixel bmp è invece il risultato della combinazione opposta, cioè B, G, R.
Per ogni singolo pixel occorre quindi invertire gli 8 bit del colore R con quelli del B, senza spostare i dati del colore G.
Inoltre, come già sottolineato, è obbligatorio convertire l'estensione bmp in un formato che possa essere gestito dall'FPGA. Occorre quindi servirsi di un software che esegua questi tre algoritmi. Il programma adoperato è stato concepito internamente alla ditta EAS. Esso è il **BmpTOBit**, realizzato in Visual Basic ed ideato specificamente per le applicazioni con moduli LCD. Il file che esso produce è di tipo bit.
Il software esegue i seguenti passaggi:

a) Rimuove dal file *.bmp i primi 54 byte, che rappresentano delle informazioni riguardanti il formato bmp;
b) Procede al salvataggio della foto tenendo conto di quanto è stato specificato;
c) Inverte la componente R con la B di ogni pixel, e viceversa.

La figura 3.4 illustra il programma **BmpTOBit** a modifiche effettuate.

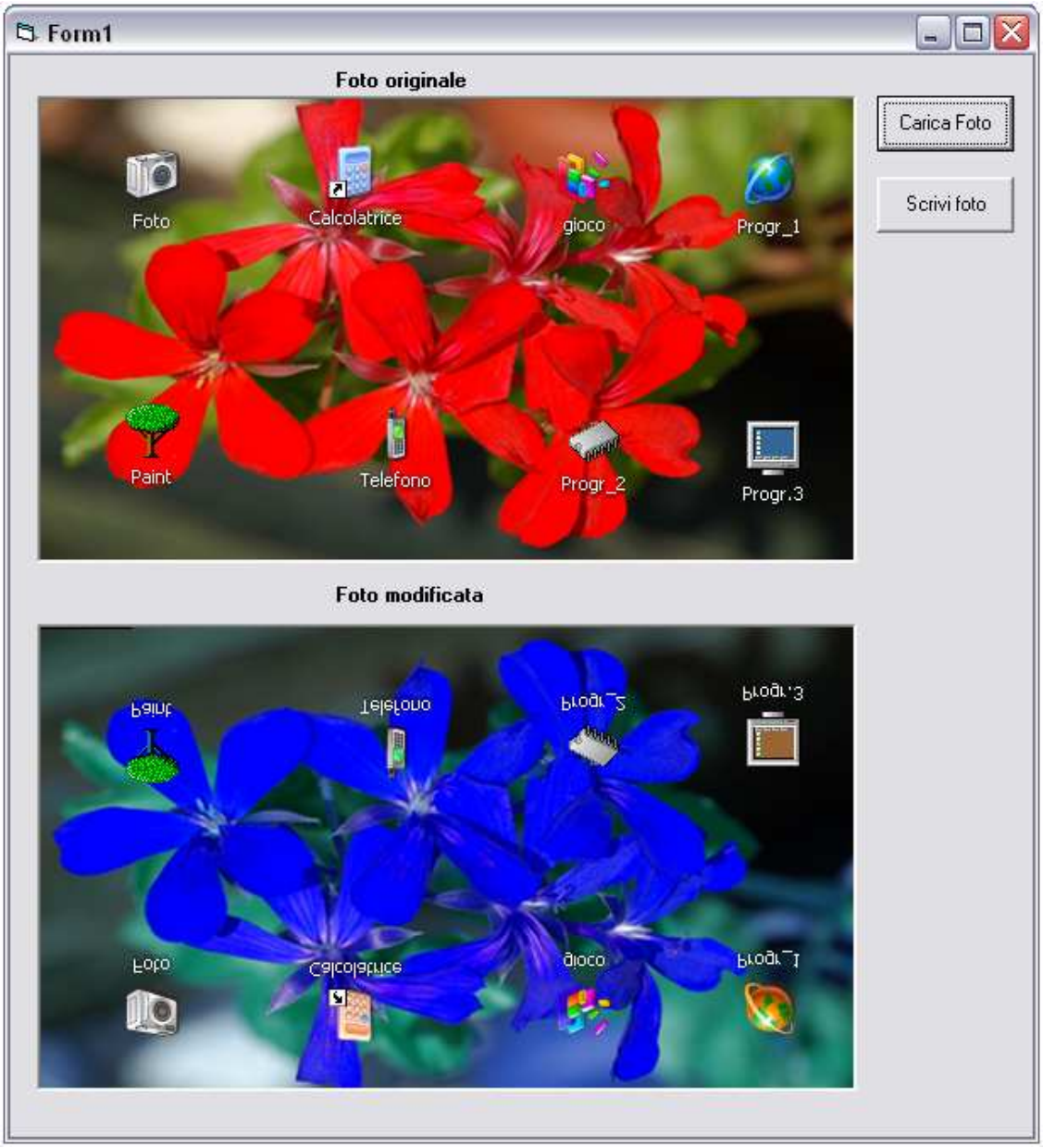


Fig. 3.4 – L'interfaccia del software BmpTOBit

2) Dopo aver scelto i frame da importare (al massimo 16), si adopera un software che realizza la *merge* (l'unione) dei vari file *.bit con il programma di boot dell'FPGA. Il software è l'**IspUFW,** della Lattice**.** Il file da esso generato ha estensione mcs.
3) Per caricare il file *.mcs nell'FPGA si usa il software denominato **IspVM System.** Il trasferimento dei dati avviene per mezzo della connessione JTAG. Contemporaneamente l'FPGA, sempre tramite connessione JTAG, trasferisce i frames alla memoria Flash utilizzando il protocollo SPI.

Le locazioni della Flash (ad indirizzi esadecimali) sono riempite come in figura 3.5:

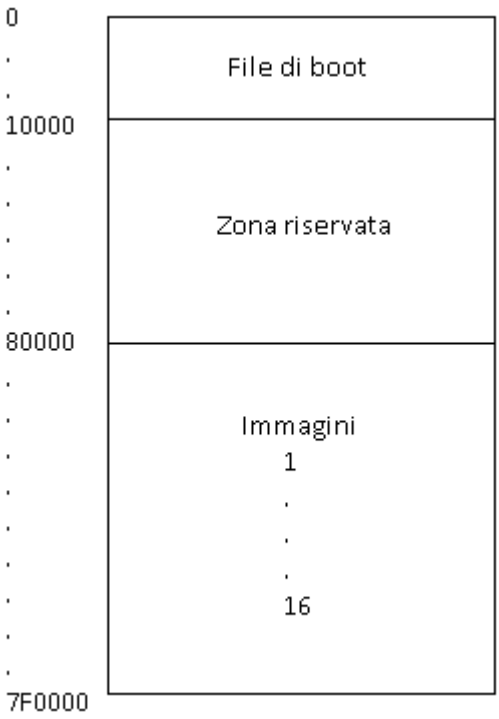


Fig. 3.5 – Suddivisione dei dati nella memoria Flash

All'accensione del modulo, la Flash carica in SDRAM le varie immagini che vengono poi visualizzate.
La prima delle foto da noi inserite, che permane circa 6 secondi (il tempo necessario a caricare i dati successivi in SDRAM), rappresenta il logo della ditta.
In seguito si visualizza il frame della figura 3.4. Se si digita l'icona "Telefono", compare l'immagine di una tastiera QWERTY (figura 3.6), che consente la scrittura di messaggi di testo.

Fig. 3.6 – La tastiera QWERTY visualizzata nel TFT

3.4 Riconoscimento del tasto digitato

Ogni pulsante della tastiera digitale è composto da un certo numero di pixel che occupano un'area definita. Se viene premuto un tasto, il microcontrollore analizza i pixel che sono stati sfiorati e controlla di quale area essi fanno parte. Quando risale alla delimitazione interessata, il carattere corrispondente appare nella parte superiore del frame.
Nella figura seguente, si osservano i singoli pixel che compongono il tasto "q". essi sono 39 in direzione x e 29 in direzione y. L'area del tasto occupa quindi 1'131 pixel quadrati.

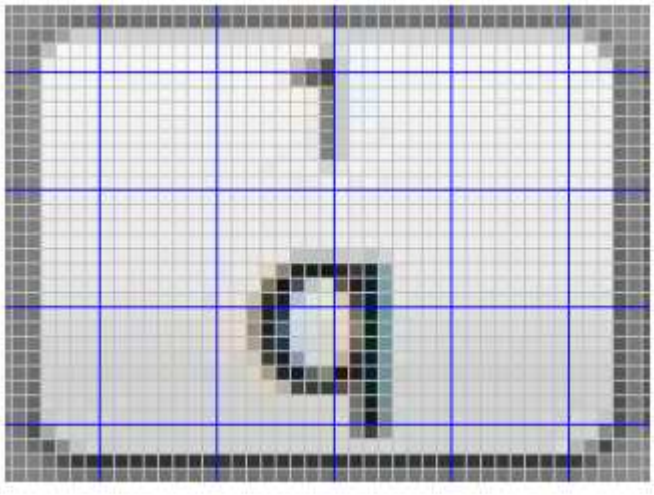


Fig. 3.7 – I pixel costituenti il tasto "q"

Anche la lettera od il numero che compare in seguito alla digitazione occupa un certo numero di pixel, in un'area ampia 45 byte (esattamente 360 pixel quadrati: altezza 24 pixel, larghezza 15 pixel). In questa zona sono presenti solamente pixel bianchi o neri.
Per recuperare la codifica binaria di ogni carattere abbiamo utilizzato un programma, il **BitFont Creator.** Esso è in grado di estrarre la codifica binaria di qualsiasi font proveniente da programmi di gestione testi. Il font da noi selezionato è il `DejaVu Sans Mono` di Microsoft Word, a grandezza 18.
Nel BitFont abbiamo impostato che la scrittura sul display del carattere avvenga per colonne, partendo dall'alto e scendendo: dopo aver raggiunto l'ultima riga si torna nel punto più alto e si percorre la colonna immediatamente a destra.

L'immagine 3.8 rappresenta l'interfaccia del BitFont Creator, con le impostazioni di formattazione appena descritte. Si noti, in basso a sinistra, la codifica esadecimale del singolo carattere per il linguaggio C: ogni carattere, secondo il proprio ordine nel codice ASCII, è stato collocato in una tabella di elementi denominata **array**, nel programma sorgente in linguaggio C.
Un array può essere assimilato ad un "insieme organizzato di oggetti", che si sviluppa verso il basso. Il concetto di "insieme" implica che tali oggetti siano dello stesso tipo: per esempio si può definire un array di "cubi", il quale non può contenere nessun oggetto "sfera".
"Organizzato" implica che sia possibile identificare senza ambiguità tutti gli oggetti dell'array in modo sistematico; nel codice C questa procedura si effettua utilizzando indici numerici che, in un array di dimensione n (cioè contenente n oggetti), vanno da 0 ad n-1.

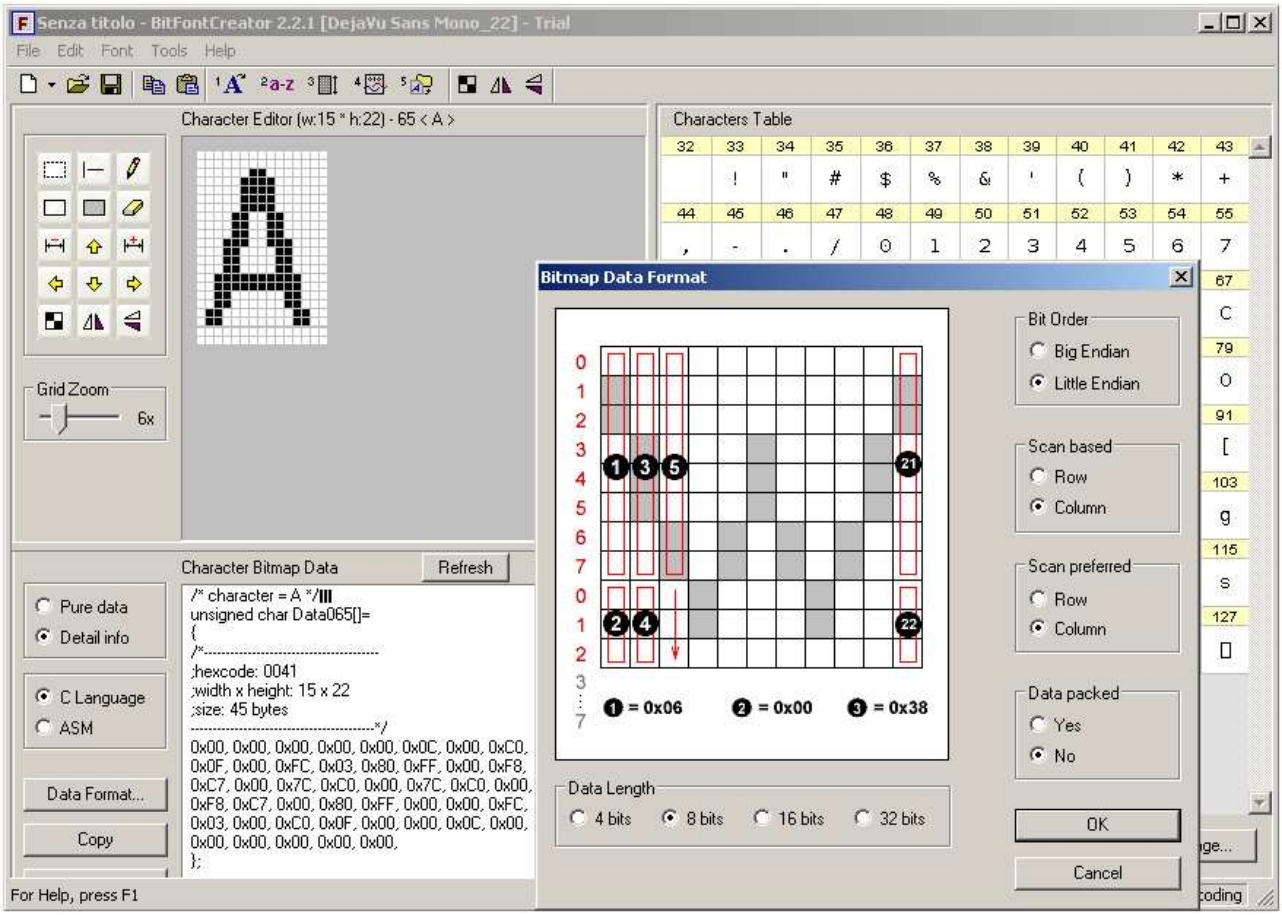


Fig. 3.8 – L'interfaccia del programma BitFont Creator

Il successivo paragrafo illustra in modo più dettagliato la procedura impiegata per la comparsa del carattere digitato su display.

3.5 Scrittura di un carattere su display

Quando viene premuto un tasto sul Touch Screen, il software (SW) ricerca il carattere che è stato sfiorato. L'algoritmo, come già accennato, confronta la superficie premuta con l'area di ogni pulsante che è stata predefinita nel codice C. In questo modo si risale alle coordinate precise del tasto.
Inoltre nel codice sorgente C è stata creata una struttura che, date le coordinate del tasto, è in grado di estrarre dalla tabella contenente la codifica ASCII il carattere corrispondente.

Il codice ASCII (American Standard Code for Information Interchange) associa un numero progressivo decimale da 0 a 255 ad ogni suo carattere, corrispondente alla posizione in tabella del carattere stesso.
Poiché non utilizziamo tutta la codifica ASCII ma solo una parte, abbiamo definito un offset pari a 32: per esempio il carattere spazio, che nel codice ASCII corrisponde al numero 32, nel nostro caso è associato all'indirizzo numero 0.
La nostra tabella "personalizzata", chiamata "CustomizedASCII", è stata inserita nel programma C. Dopodichè si è provveduto a creare una secondo array più specifico, denominato "FontSystem", contenente

la codifica esadecimale dei byte componenti ogni singolo carattere. La codifica di ogni carattere è stata ottenuta utilizzando il BitFont Creator, come in figura 3.8.

Si suppone di digitare su tastiera il carattere “0”. Esso occupa la 48ma posizione nella codifica ASCII, che corrisponde alla posizione 16 dell’array CustomizedASCII. Poiché ogni carattere è ampio 45 byte, per risalire al primo byte del carattere “0” il microcontrollore effettua il seguente calcolo:

- 16 * 45 = 720

Ossia il primo byte del carattere “0” occupa la 720ma posizione di FontSystem.

Successivamente il microcontrollore calcola il pixel esatto da cui cominciare a scrivere il carattere (a partire dall’alto a sinistra). Dopo ogni scrittura o cancellazione di caratteri, il micro aggiorna in automatico la posizione da cui scrivere il carattere successivo.
Dopodichè inizia la scrittura del carattere per colonne, da sinistra verso destra.

Poiché una colonna occupa 24 pixel, essa è ampia 3 byte. La sua scrittura ha inizio dal byte più in alto e prosegue verso il basso. Si ricorda che, per recarsi da un pixel collocato nella n-esima riga a quello immediatamente inferiore, l’incremento necessario allo spostamento non è 1 bensì 480, poiché il pixel inferiore è situato nella riga n+1.

Quando i 3 byte di una colonna sono stati scritti, per procedere ai successivi si torna sempre al 1° byte della prima colonna. A questo punto, per portarsi alla 1^{a} colonna vuota a destra, è necessario sommare un valore pari al numero di colonne che sono già state completate.

Il diagramma di flusso seguente (figura 3.9) illustra con maggior semplicità l’algoritmo appena descritto.

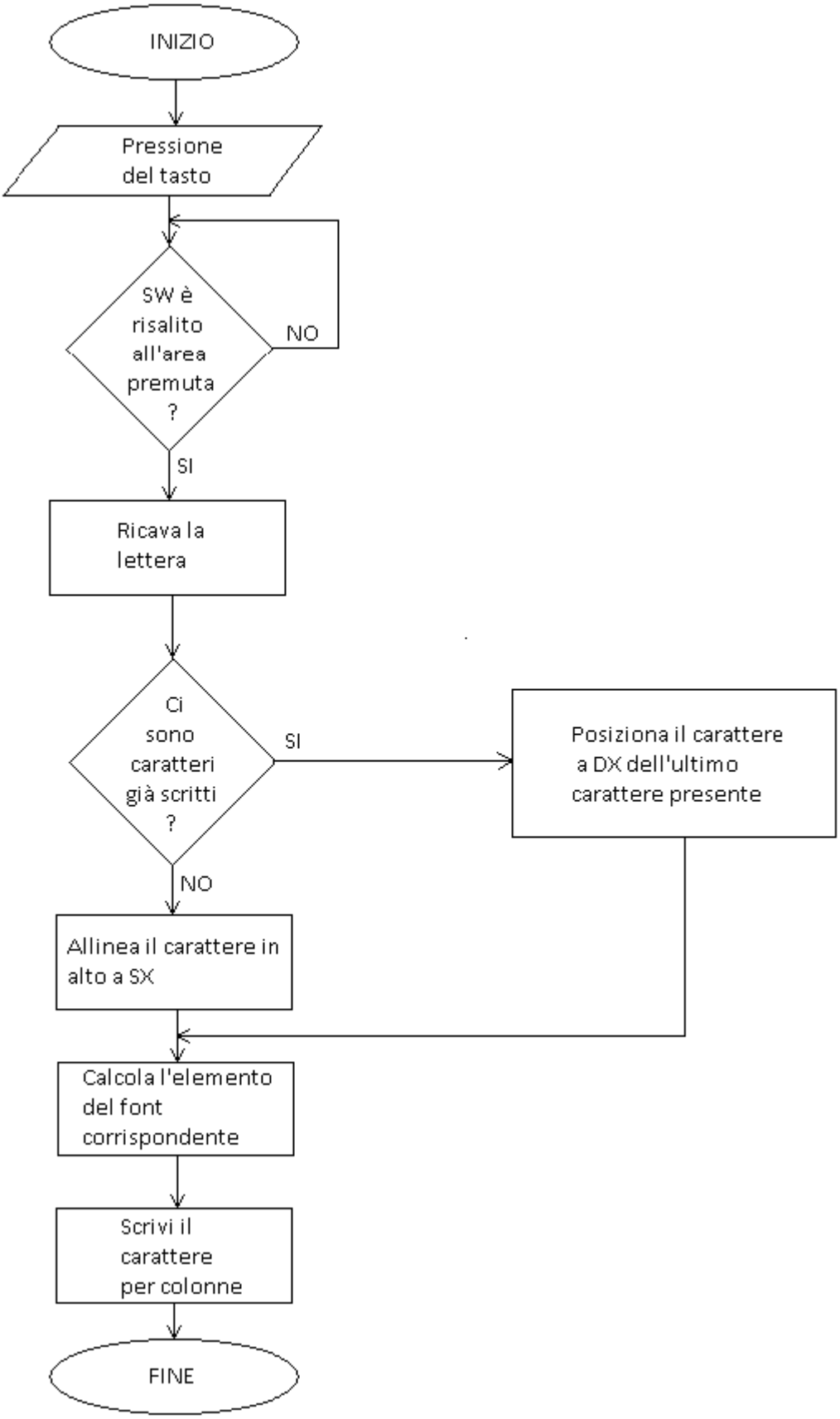


Fig. 3.9 – Flow chart che riassume la scrittura di un carattere sul display

3.6 Il modulo Telit: sintassi dei comandi principali del Protocollo at

3.6.1 Introduzione

Come sottolineato in precedenza, il modulo GSM accetta solamente istruzioni previste dalla sintassi del Protocollo at, pena la restituzione di messaggi di errore nel Terminal.
Tutti i comandi devono iniziare con la stringa di testo:

at

infatti, una prima verifica della sintonizzazione può consistere nell'inviare la seguente stringa:

at<cr>

L'acronimo <cr> significa "carriage return", letteralmente "riportare indietro il carrello". Ciò avviene premendo il tasto Invio: in questo modo il cursore del Terminal si abbassa di una riga e si allinea a sinistra. Se trasmissione e ricezione sono sintonizzate, sul Terminal appare il messaggio OK, altrimenti ERROR. Ci sono due tipi di comandi at:

- Parametrici: possono essere di lettura (per visualizzare dei dati), di settaggio (per impostare dei valori) o di test (per determinare una gamma di valori supportati).
- Di azione: possono essere di esecuzione (per invocare funzioni particolari o dispendiose da punto di vista della memoria) o di test.

3.6.2 I comandi di utilità generale più importanti

In seguito riporto il set dei principali comandi at, senza soffermarmi sulla sezione delle chiamate, in quanto esulano dagli obiettivi finali del progetto.

1) Controllo della presenza della carta SIM (tramite codice PIN):

at+cpin?<cr>

Il Terminal può fornire le seguenti risposte:

Risposta	**Motivazione**
+CPIN:SIM PIN	SIM presente, è richiesto il codice PIN
+CPIN:SIM PUK	SIM presente ma, dopo 3 inserimenti di codici PIN scorretti, è richiesto il PUK
+CPIN:READY	SIM presente, non è richiesto il codice PIN
+CME ERROR:10	SIM non presente
+CME ERROR:13	SIM difettosa
+CME ERROR:14	SIM occupata, riprovare più tardi
+CME ERROR:15	SIM sbagliata

Per i primi due casi in tabella, si prosegue in questo modo:
Comando **at+cpin=****<cr>** per fornire PIN SIM se richiesto.
Comando **at+cpin=*******,<newpin><cr>** per fornire PUK SIM se richiesto e nuovo PIN.

2) Controllo dello stato della rete:

at+creg?<cr>

Le risposte possibili sono:

Risposta	Motivazione
+CME ERROR:10	SIM non presente o danneggiata
+CME ERROR:11	SIM presente ma è richiesto il PIN
+CREG: 0,0 +CREG: 1,0	Non è stata trovata nessuna rete GSM
+CREG: 0,1 +CREG: 1,1	Registrato nella rete
+CREG: 0,2 +CREG: 1,2	Non è registrato in nessuna rete ma sta cercando una rete disponibile per connettersi
+CREG: 0,3 +CREG: 1,3	Ha trovato una rete ma non è riuscito a registrarsi in quanto in quella rete non è permesso il roaming
+CREG: 0,4 +CREG: 1,4	Ha trovato reti disponibili ma non è riuscito a registrarsi in nessuna di esse
+CREG: 0,5 +CREG: 1,5	Ha trovato delle reti ma è gia registrato in una di esse

3) Identificazione dell'operatore di rete:

at+cops?<cr>

Il modulo risponde con:

+COPS: (<stat>,"Nome Network","Format")

<stat>	0 rete sconosciuta	
	1 rete disponibile	
	2 rete corrente	
	3 rete proibita	

Format	5 numeri	primi 3	identificativo nazione
		ultimi 2	codice della rete

Esempio: *+COPS: (2,"I VODAFONE Omni","22210")*

4) Ricezione e qualità del segnale

at+csq<cr>

Risposta:

+CSQ: <rssi>,<bear>

<rssi>	Intero tra 0 e 99 che indica la "forza" del segnale (associato ad un valore in dBm)
<bear>	Intero tra 0 e 7 che riporta la qualità del segnale ricevuto

5) Controllo veloce dello stato di rete

A registrazione in rete avvenuta, può essere utile conoscere la forza del segnale e la rete a cui si è connessi. Per accedere contemporaneamente a queste informazioni il comando da adoperare è:

at#moni?<cr>

In seguito compare la stringa:

#MONI: <netname> BSIC:<bsic> RxQual:<qual> LAC:<lac> Id:<id> ARFCN:<arfcn>
PWR:<dBm> TA:<timadv>
OK

netname	Nome dell'operatore di rete
bsic	Codice identificativo della stazione di base
qual	Qualità della ricezione (0:7)
lac	Codice di localizzazione dell'area
id	Identificativo della cella
arfcn	Frequenza del canale radio assegnata
dBm	Forza del segnale in dBm
timadv	Ritardo di sincronizzazione

Esempio: *#MONI: I WIND BSIC:74 RxQual:7 LAC:5DDF Id:5687 ARFCN:748*
PWR:-74dbm TA:0
OK

6) Accesso alla rubrica

at+cpbs=<PB><cr>

PB	SM: Rubrica SIM
	FD: Tastierino telefonico
	LD: Elenco ultime chiamate effettuate
	MC: Elenco chiamate perse
	RC: Elenco chiamate ricevute

Possibili risposte:

Risposta	**Motivazione**
OK	PB selezionato è attivo
ERROR	Errore avvenuto
+CME ERROR:10	SIM non presente
+CMS ERROR:310	SIM non presente
+CME ERROR:11	SIM presente ma necessita del PIN per continuare
+CMS ERROR:311	SIM presente ma necessita del PIN per continuare
+CME ERROR:12	SIM presente ma necessita del PUK per continuare
+CMS ERROR:316	SIM presente ma necessita del PUK per continuare
+CME ERROR:13	SIM difettosa
+CMS ERROR:313	SIM difettosa
+CME ERROR:14	SIM occupata
+CMS ERROR:314	SIM occupata
+CME ERROR:15	SIM inserita non è del tipo giusto
+CMS ERROR:315	SIM inserita non è del tipo giusto
+CME ERROR:17	Richiesto PIN2 per continuare

3.6.3 I comandi per la gestione del servizio SMS

Gli SMS (Short Message Service) possono avere due tipi di formato:

- PDU;
- Testo.

Il PDU (Protocol Data Unit) è il protocollo numerico con cui l'SMS si "affaccia" al network durante la fase di invio.
La modalità Testo, invece, è quella a noi familiare.
Il Telit prevede il PDU come impostazione di default. Quindi, se si legge un ipotetico messaggio in entrata senza aver prima inizializzato la modalità Testo, ciò che si vede è una sequenza di numeri apparentemente inspiegabile.

1) Scelta del tipo di formato SMS:

at+cmgf=<mode><cr>

<mode> è appunto il tipo di formato, associato a due valori numerici:

0- PDU;
1- Testo.

Pertanto, la stringa che digitiamo è:

at+cmgf=1

A impostazione avvenuta, sul Terminal appare il messaggio OK.

Quando dal Telit si manda un SMS, prima di giungere a destinazione esso è inviato ad un Numero Centro Servizi (SMSC). Da qui il messaggio viene spedito verso la meta o rimane in sospeso fino a che la consegna può avvenire. Per garantire un comportamento corretto del servizio, il Numero Centro Servizi deve essere supportato dall'operatore SIM.

2) Controllo del Numero Centro SMS:

at+csca?<cr>

La risposta che si visualizza nel Terminal è:
+CSCA: <number><type>
OK
Dove:
<number> = Prefisso della Nazione (*Esempio: +39*)
<type> = Numero telefonico del Centro Servizi (*Esempio: 3492000200*)

3) Aggiunta del numero Centro SMS (solo se richiesto):

at+csca=<number><type><cr>

Dopo qualche istante compare il messaggio OK.

Il Protocollo at definisce un'istruzione per impostare il comportamento del telefono nel caso si presenti un SMS in entrata. Questo comando è stato concepito per poter gestire e far fronte ad eventuali situazioni indesiderate con maggior facilità.

4) Trattamento del nuovo SMS in entrata:

at+cnmi=<mode>,<mt>,<bm>,<ds>,<bfr><cr>

Dove:

<mode>

Definisce una serie di opzioni per la gestione degli eventi non desiderati.

0- Nel caso in cui ci sia un messaggio in entrata e la memoria messaggi è esaurita, l'SMS che è memorizzato da più tempo può essere eliminato e sostituito con il nuovo messaggio ricevuto.
1- L'indicazione sull'evento indesiderato è rifiutata ed il codice annesso a tale risultato è respinto quando il Terminal è riservato (per esempio durante una chiamata), altrimenti, se è libero, il codice è trasmesso direttamente al Terminal.
2- Nel caso in cui si verifichi un avvenimento indesiderato ed il Terminal è occupato, i codici vengono mandati al Terminal previa "prenotazione". Se è libero, vi entrano direttamente.
3- Se <mt> è impostato a 1 (mentre il modulo è in modalità GPRS online), quando si riceve un SMS, la linea sonora di hardware è abilitata per 1 secondo.

<mt>

Parametro che permette o meno di visualizzare i risultati di avvenuta ricezione degli SMS in arrivo.

0- Al Terminal non viene inviato nessun segnale di avvenuta ricezione.
1- Il dato sulla posizione in memoria del nuovo messaggio è inviato al Terminal, da cui può essere visualizzato, tramite la sintassi seguente:
+CMTI: <memr>,<index>
Dove:
<memr> =tipo di memoria in cui il nuovo SMS è immagazzinato (di solito nella SIM)
<index>= posizione in memoria del nuovo SMS

<bm>

Segnalazione delle opzioni sulla trasmissione degli SMS.

0- I messaggi trasmessi non vengono inviati al Terminal;
2- I messaggi trasmessi vengono inviati al Terminal.

<ds>

Opzioni sullo Status Report degli SMS. Nel caso di nuovo messaggio in entrata, esso è "ricevuto e non letto" (Received Unread). Dopo che lo si è visualizzato almeno una volta, esso risulta "ricevuto e letto" (Received Read).

0- Lo Status Report non viene comunicato al Terminal;
1- Lo Status Report viene comunicato al Terminal.

<bfr>

Metodo manuale di gestione del buffer in merito ai codici degli eventi.

0- Il buffer contenente i codici degli avvenimenti non desiderati (ERROR) è inviato al Terminal quando il parametro <mode> è inizializzato con i valori 1, 2 o 3.
1- Il buffer è svuotato quando <mode> = 1, 2 o 3.

L’istruzione è stata da noi impostata nel modo seguente:

at+cnmi=2,1,2,1,0

Ora è possibile inviare e ricevere SMS senza il verificarsi di situazioni ambigue.
La gestione dei messaggi di testo è completa con l’utilizzo dei seguenti 5 comandi.

3.6.4 I comandi per l’invio e la ricezione degli SMS

5) Scrivere un nuovo SMS da memorizzare

at+cmgw=“<da>”<cr>

Dove:
<da> = destination address (*Esempio: +393401234567*)
Dopo aver premuto il tasto Invio

- Attendere la comparsa del carattere “>”;
- Scrivere il messaggio (al massimo 160 caratteri);
- Terminare l’SMS con la combinazione “Ctrl+Z” o annullarlo con il pulsante “esc”;
- Attendere la risposta.

Risposta	**Motivazione**	**Passo successivo**
AT+CMGW: <index> OK	L’SMS è salvato in posizione <index>	Procedere
ERROR	Errore generico	Ritorno del tipo di errore
AT+CMS ERROR: 330	Indirizzo SMSC sconosciuto	Inserire indirizzo SMSC
AT+CMS ERROR: 322	Memoria piena	Liberare parte di memoria SMS e riprovare

6) Inviare un SMS precedentemente memorizzato

at+cmss=<index><cr>

Gli errori più frequenti sono legati all’assenza di rete. In tal caso, è sufficiente effettuare un controllo sulla potenza e la qualità del segnale.
La restituzione della stringa ERROR: 96 implica la mancanza di informazioni obbligatorie.

7) Inviare un SMS senza memorizzarlo

at+cmgs=“<da>”<cr>

Il messaggio deve terminare sempre con “Ctrl+Z”.

8) Cancellare un SMS

at+cmgd=<index><cr>

L’eventuale comparsa della scritta ERROR: 321 segnala che l’indice di memoria non è valido oppure risulta già vuoto.

9) Leggere un SMS

at+cmgr=<index><cr>

Il messaggio compare dopo la stringa AT+CMGR: insieme ad altre informazioni selezionate con l’istruzione at+cnmi.

3.7 Inserimento ed utilizzo del Protocollo at nel codice sorgente C

Si è visto come dall'Hyper Terminal si possano mandare le istruzioni che si desiderano far compiere al Telit. Quando il sistema funziona autonomamente, non si utilizza più il Terminal e i comandi del Protocollo at previsti sono "lanciati" dal microcontrollore (programmato in linguaggio C).
In particolare le istruzioni prescelte avvengono in modo sequenziale: se l'istruzione i-esima produce un errore (gli errori possibili sono stati definiti nella sezione 3.6), il successivo comando non viene eseguito e si "salta" ad una routine di gestione degli errori.

Lo schema adottato è la creazione di due array di elementi char (caratteri, ossia facenti parte del codice ASCII), denominati "BufferTX" e "BufferRX", inizialmente vuoti. Gli elementi interessati di BufferTX conterranno un singolo carattere dell'istruzione at, mentre in BufferRX comparirà la stringa con la sola risposta del modulo. Poiché la stringa più lunga che possa verificarsi è la scrittura o la lettura di un SMS di 160 caratteri, l'array è stato dichiarato di 160 elementi (n=1,….,159).
Quando si esegue il programma, la prima istruzione che esso invia è il semplice test

at

che riempie i primi 2 elementi di BufferTX.
Se il Telit è pronto, la risposta che compare è OK, ed essa occupa i primi 2 elementi del secondo array.
Per essere sicuri che l'istruzione abbia esito positivo, il metodo scelto è il seguente: si inizializza un controllo nell'ultimo elemento di BufferRX e lo si fa risalire finché esso percepisce una K e, all'indice immediatamente minore, una O. In questo caso significa che il modulo funziona correttamente. Se il primo elemento trovato nell'indice x fosse una R e nell'indice x-1 si incontrasse una O, si sarebbe in presenza di ERROR ed il modulo non potrebbe inviare né ricevere messaggi.
Il comando successivo, per la verifica della rete, è il seguente:

at+creg?

Se il telefono è registrato in rete, la seconda delle due cifre restituite in BufferRX è "1". In questo caso, per poter passare all'istruzione successiva, è sufficiente controllare che l'ultimo elemento riempito di BufferRX sia "1" e quello ad indice immediatamente inferiore sia "," (per non creare ambiguità nel caso si verificasse l'evento +CME ERROR: 11).
Una procedura analoga è stata applicata anche ai successivi comandi:

at+cmgf=1, per l'inserimento della modalità Testo;

at+cnmi=2,1,2,1,0, per la gestione di eventuali SMS in entrata.

L'algoritmo di programmazione è progettato in modo che, non appena l'immagine della tastiera è caricata, compare anche la frase "Ricerca Rete…" in alto a sinistra. Subito dopo appare il testo "Inserire messaggio:". In seguito alla scrittura dell'SMS, dopo la pressione del tasto Invio, compare la scritta "Inserire numero telefonico:". Per disporre della modalità numerica si preme il riquadro "12#" (fig. 3.6).
Dopo aver confermato, si materializza la stringa "Invio in corso" e, qualche istante più tardi, "Messaggio inviato". La procedura appena indicata è stata costruita compatibilmente con l'istruzione

at+cmgs="<da>"

Nel caso di SMS in arrivo, il testo del messaggio appare direttamente sullo schermo, previa gestione del comando di lettura

at+cmgr=<index>

I messaggi riconosciuti (quindi pre-impostati nel codice sorgente C) sono:
"Attiva Rele";

“Disattiva Rele”.
L’invio di questi due SMS ad una serie di relè ne causa rispettivamente l’eccitazione e lo spegnimento.
Nel caso in cui la richiesta in corso sia di attivare i relè, qualora essa produca l’effetto desiderato, sul display compare la scritta “Rele Attivato”. Quando i relè sono stati disattivati con successo appare la stringa “Rele Disattivato”.

CAPITOLO 4

CONCLUSIONI

4.1 Obiettivi raggiunti

Il progetto è stato portato a termine nei tempi prestabiliti.
Attualmente, il sistema è in grado di:

- Scrivere ed inviare SMS a qualsiasi telefono GSM;
- Ricevere messaggi da un telefono GSM e visualizzarli sul display TFT;
- Riconoscere i messaggi in entrata “Attiva Rele” e “Disattiva Rele”, eseguire l’incarico e segnalare su TFT rispettivamente “Rele Attivato” e “Rele Disattivato”.

4.2 Documentazione dei risultati ottenuti

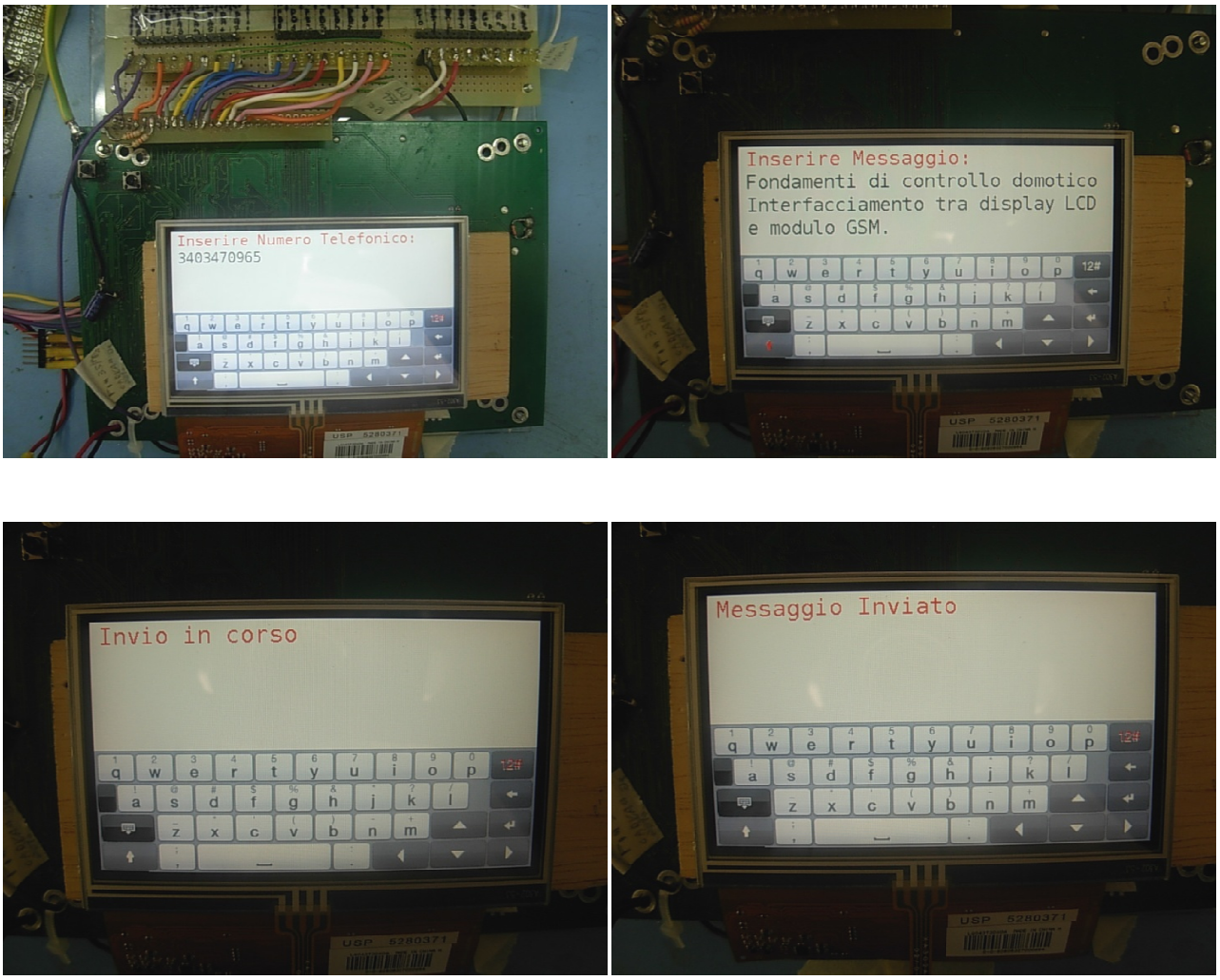


Fig. 4.1 – Procedimento "step by step" per l'invio di un SMS. Nella settima ed ultima immagine è ben visibile la configurazione a 4 fili del Touch Screen resistivo. Nella terza foto, in basso a sinistra, si noti la connessione JTAG.

Fig. 4.2 – Segnalazione sul display di nuovo messaggio ricevuto.

4.3 Obiettivi futuri

I prossimi obiettivi da raggiungere sono:

- Visualizzare su TFT la temperatura all'interno di una stanza, rilevata con l'impiego di sensori di temperatura;
- Riconoscere i messaggi in entrata "Apri Serrande" e "Chiudi Serrande", eseguire il comando e segnalare su display rispettivamente "Serrande Aperte" e "Serrande Chiuse".
- Riconoscere i messaggi in entrata "Attiva Impianto Irrigazione" e "Disattiva Impianto Irrigazione", eseguire il comando e segnalare su display rispettivamente "Irrigazione Attivata" e "Irrigazione Disattivata".

RINGRAZIAMENTI

Ringrazio in primo luogo il titolare della EAS Elettronica, il sig. Giorgio Geronazzo, il quale mi ha permesso di svolgere un'esperienza fondamentale sia sotto il profilo della conoscenza, sia per quanto riguarda i rapporti e gli scambi di idee con i colleghi di lavoro.

A questo proposito ringrazio tutto il personale della sezione "Ricerca e Sviluppo" ed in particolare il mio tutor aziendale, il sig. Carlo Fongaro, per i suoi preziosi insegnamenti in merito all'elettronica di potenza e alla programmazione informatica, oltre che per la pazienza con cui mi ha seguito durante tutto lo stage.

Un enorme "grazie" va ai miei genitori, che mi hanno sostenuto economicamente e moralmente durante il mio percorso di studi.

Ringrazio infine i miei migliori amici: quelli che mi dicono SEMPRE di non mollare MAI.