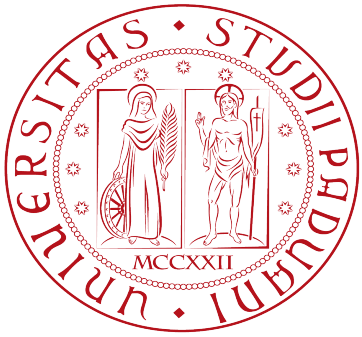


Università degli studi di Padova

FACOLTA' DI INGEGNERIA
Corso di Laurea Triennale in Ingegneria Elettronica

TESI DI LAUREA TRIENNALE
RELAZIONE DI TIROCINIO

Relatore:
Prof. Paolo Tenti

Candidato:
Bisenzi Andrea
Matr. 539487

Progetto e Realizzazione di una scheda a microcontrollore per la regolazione e la verifica dell'impianto frenante di un'automobile

Anno Accademico 2010-2011

Sommario

OGGETTO: Tirocinio presso Jofa di Padova

PROGETTO: Sviluppo di una scheda elettronica destinata al montaggio su attrezzo per collaudo e regolazione dell'impianto frenante di un'autovettura e interfacciamento della stessa con nuovo display OLED.

L'attrezzo non viene qui analizzato nello specifico ma viene data una semplice descrizione delle sue funzioni.
L'obiettivo principale dello stage riguarda la progettazione di una scheda di controllo digitale con micro MC9S08DZ128 della Freescale Semiconductor; esiste in azienda una versione della scheda perfettamente adatta all'utilizzo sugli odierni strumenti ma ormai obsoleta rispetto al resto della tecnologia utilizzata.
La richiesta di un'estetica e di una funzionalità evoluta inoltre porta allo studio e alla valutazione dell'eventuale utilizzo del display a matrice attiva uOLED-32028-P1T.

Indice

Elenco delle figure

Introduzione

1.1 L'Azienda

La Jofa srl è una società sita in Padova, in via della meccanica 8/10.
Nata nel 1972, inizia la propria produzione negli ambiti dell'automazione industriale e del collaudo, successivamente si specializza in elettronica industriale sviluppando e producendo strumenti digitali multicontrollo di analisi.
Nel 1980 introduce l'utilizzo di personal computer industriali interfacciati con elettronica remotata, diventando così un riferimento importante per sistemi di collaudo e diagnosi nelle più svariate applicazioni.
Nel 1985 raggiunge una quasi totale autonomia produttiva, realizzando anche le prime iniezioni elettroniche utilizzate da team di formula uno italiani.
Nel 1990 diviene fornitore partner delle più grandi case automobilistiche italiane esportando nelle sedi di tutto il mondo strumenti per il collaudo e l'automazione dei processi produttivi.
Ottiene la certificazione Sistema di Qualità RINA ISO90001 e inizia la collaborazione con l'università degli studi di Padova sviluppando in cooperazione con FIAR il sistema di accensione multiscintilla di un motociclo.
Negli anni 1995-2000 Jofa, assieme ad altre 4 aziende, fonda il Centro Ricerca Elettronica Industriale Veneto (CREI veneto, www.creiven.it) e realizza reti locali (connessioni multipunto wireless e via cavo) con Server/Concentratori di gestione di differenti collaudi indipendenti.
Negli anni 2000-2005 l'azienda ottiene la certificazione Vision2000 per il sistema qualità e realizza linee di produzione/assemblaggio completamente automatiche, pc traceability, server di gestione, concentratori, controlli di processo e sistemi di collaudo.
Ad oggi la società conta 29 dipendenti tra tecnici, disegnatori, progettisti e meccanici.
L'azienda lavora partendo dall'idea di un sistema automatico e sviluppa un sistema completo che lo implementa, passando per progettazione elettronica, meccanica e la realizzazione software appoggiandosi ad aziende esterne solo per piccoli ordini di materiale o di assemblaggio di schede elettroniche che comporterebbero una spesa maggiore se eseguiti in sede.

1.2 L'ambito lavorativo e i settori di sviluppo

Jofa inizia la sua storia principalmente nella progettazione elettronica industriale ma ben presto grazie all'eccellenza del suo lavoro espande il proprio ambito lavorativo in altri campi.
Aiuta molto, in questa positiva deviazione di rotta, il costante aggiornamento di conoscenze e tecnologie.
Oggi Jofa collabora con le maggiori aziende automobilistiche italiane quali FIAT, Ferrari, Lancia, IVECO e progetta strumenti e sistemi automatici a partire dall'idea di un processo di produzione.
I progetti e gli strumenti sono sviluppati per auto, motocicli, autocarri, macchine movimento terra, imbarcazioni, laboratori di ricerca, siti produttivi e circuiti ospedalieri.
I prodotti proposti dall'azienda sono molteplici e principalmente a struttura custom per personalizzarli e modellarli secondo le esigenze del cliente.

Le principali categorie sono:

- motion control
- test di tenuta
- rf wireless
- test di carico
- linee di montaggio (automatiche/semiautomatiche)
- ccd e vision

- gestione dati
- test ABS
- controlli di processo/prodotto

I prodotti Jofa sono installati direttamente presso gli stabilimenti di produzione del cliente.
La distribuzione del lavoro e dei prodotti è cosi composta:

Per Collaborazione:

- Il 60% della produzione è realizzato direttamente per le case automobilistiche di cui è fornitore partner (FIAT, Lancia, Alfa Romeo, Ferrari, Iveco, Komatsu, Pininfarina, Maserati Nissan...)
- Il 30% per le industrie che producono componenti per le case automobilistiche (Lear, Lasme, Magneti Marelli, Api, Gestamp...)
- Il 10% per settori diversi (Nautica, Centri di Ricerca, Università)

Per distribuzione del prodotto:
Il 40% della produzione in Italia (Torino, Milano, Brescia, Cassino, Melfi, Napoli, Maranello, Modena, Sulmona, Sevel)

- Il 30% in Europa (Spagna, Polonia, Turchia, Germania, Francia)
- Il rimanente 30% nel resto del mondo (Argentina, Brasile, Venezuela, Cina, Marocco, Sud Africa, Tailandia).

1.3 Il mio ruolo in azienda

Ho conosciuto Jofa cercando in rete alcune aziende di elettronica a Padova.
Ho contattato la segreteria dell'azienda, inviato il curriculum vitae e dopo qualche giorno ho incontrato colui che è poi divenuto il mio tutor aziendale.
Dopo aver visitato altre 2 aziende di elettronica ho scelto di svolgere il tirocinio presso Jofa.
Ho iniziato l'esperienza di stage il 2/11/2010.
Abbiamo da subito steso un programma contenente gli obiettivi e alcuni punti opzionali.
Il mio compito iniziale riguardava lo studio del circuito e del software di uno strumento realizzato in anni di attività e ancora oggi presente sul mercato; il successivo aggiornamento con sostituzione del microcontrollore e di vari componenti nonché l'inserimento di altre periferiche richieste dai clienti. Successivamente avrei dovuto studiare, programmare e valutare l'utilizzabilità di un nuovo tipo di display OLED touchscreen da montare eventualmente su alcuni strumenti. Mi è stato assegnato un pc in cui ho installato i programmi principali con cui lavorare tra i quali Orcad, Openoffice, Workshop 3.0, CodeWarrior, PSPad, CAM350 e Visual Basic 6.0.
Mi è stato inoltre garantito l'accesso ai banchi di lavoro dove ho creato alcune schedine utili per i primi test e le prime connessioni con i nuovi componenti.

Circuiti oggetto di studio

2.1 Microcontrollore MC68HC705B16

Il micro MC68HC705B16 è membro della famiglia Motorola 68HC05 di microcomputer economici a singolo chip. Le MCU appartenenti a questa famiglia sono a 8 bit e contengono un oscillatore interno, CPU , RAM, ROM, EEPROM, convertitore A/D, uscite PLM, pins di I/O, interfaccia di comunicazione seriale, timer programmabili e watchdog.
Nello specifico il micro utilizzato ha come caratteristiche:

- OTPROM(one-time programmable ROM)
- 15 kbytes EPROM
- 352 bytes di RAM
- 576 bytes bootstrap ROM
- Resistori di pull-down opzionali nei pin delle porte B e C.

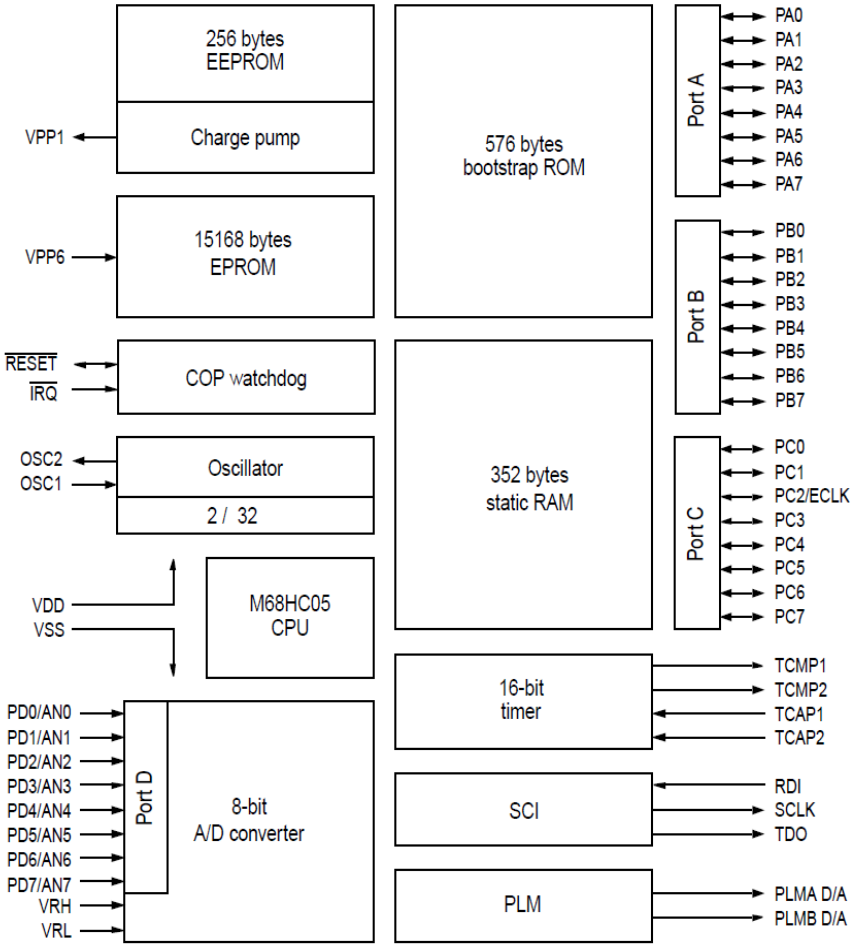


Figura 2.1.1

La figura 2.1.1 rappresenta lo schema a blocchi del micro montato su gran parte degli strumenti JOFA .

Osserviamo che ci sono 3 porte di I/O generici e una porta di acquisizione per segnali analogici connessi ad un convertitore analogico-digitale a 8 bit, un timer a 16 bit, un modulo seriale e un modulo PLM.
Il micro studiato viene utilizzato dall'azienda da circa 10 anni senza problemi e la scelta di sostituirlo con uno nuovo è stata fatta principalmente per la futura reperibilità del componente a rischio obsolescenza.
Purtroppo, data l'età commerciale del componente, non è possibile eseguire la programmazione on-board e con essa nemmeno un controllo real-time dell'esecuzione del programma.
Inoltre la tecnologia delle memorie interne del componente standard non permette la cancellazione ne elettronica ne in diversi modi della programmazione e quindi una volta programmato il componente non lo si può più riutilizzare.
In azienda è presente una discreta quantità di componenti one-time programmabili e solo 4 componenti nella versione cancellabile tramite irraggiamento UV.
La programmazione avviene tramite una basetta distribuita dall'azienda costruttrice riportata in figura 2.1.2

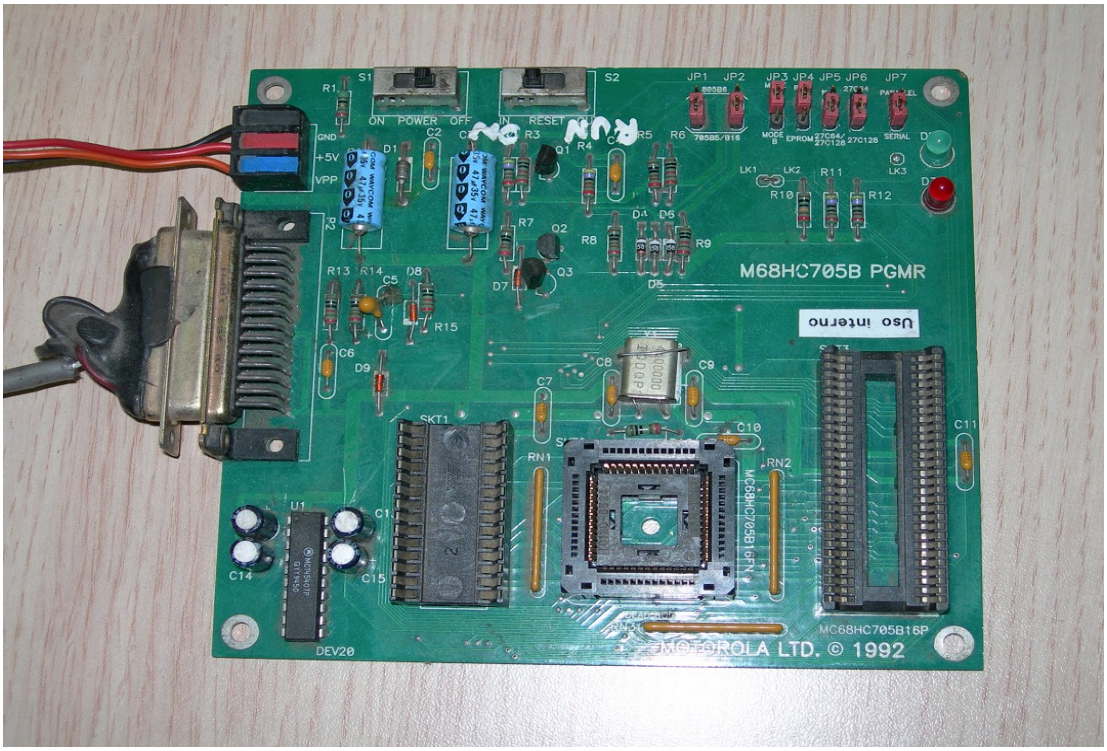


Figura 2.1.2

Il programmatore funziona in ambiente DOS con un'applicazione chiamata epbx32 ricevendo il programma già compilato e linkato con estensione OK.
Le schede analizzate sono state sviluppate per essere utilizzate in linee di produzione e studiate per uniformare e minimizzare i tempi di esecuzione dei controlli; inoltre quando sono utilizzati in grosse linee è importante che, in caso venga rilevato un difetto, la linea non si blocchi mai.
Tempi persi per eventuali difetti in un singolo processo corrispondono ad una perdita di denaro consistente perché bloccano l'intera linea di produzione.
E' importantissimo quindi dare continuità ai test e alle regolazioni in modo che in caso di eventuale insorgenza di problema, esso si risolva in modo autonomo .
Sono state utilizzate 2 schede prodotte da JOFA, la scheda FRENO che è stata poi modificata e aggiornata e la scheda COMFORT che è stata utilizzata soltanto a fini didattici per utilizzare componenti aggiuntivi quali ad esempio la RAM.

2.2 Componenti della scheda FRENO

La scheda FRENO si utilizza in un attrezzo che serve a testare e regolare il pedale del freno a bordo macchina. Lo strumento viene posizionato nella struttura dell'autovettura priva di comfort interni (sedili, volante, cruscotto, etc..) e fissato in modo tale che il braccio estendibile con vite senza fine prema il pedale per misurarne l'efficacia in frenata. L'attrezzo comprende un vasto assortimento di funzioni di diagnosi e test interni in modo da mantenere elevata la precisione nelle misurazioni; va quindi eseguita una calibrazione dello stesso ogni 4000/5000 esecuzioni. Lo strumento agisce con circa 30 pressioni sul pedale e contemporaneamente, se il conseguente aumento della pressione dell'olio nel circuito freni non è sufficiente o è troppo elevata, segnala di aggiungere o togliere fluido dal sistema. Queste misurazioni permettono di controllare e appurare anche la presenza di perdite del sistema, la presenza di sacche d'aria all'interno del circuito e appunto l'efficacia dell'impianto frenante. Questo sistema automatico di calibrazione è ampiamente usato nelle linee di produzione FIAT e di altre case produttrici in tutto il mondo perché permette di automatizzare un processo di controllo necessario sia per il comfort che per la sicurezza nella produzione di autovetture.
Uno strumento analogo è utilizzato per il test del pedale di frizione; in entrambi i casi lo strumento è interfacciato con un sistema di memorizzazione e analisi dati che permette di registrare tutte le singole prove e gli esiti dei singoli processi che portano al prodotto finito. Esiste quindi oltre allo strumento in se, tutto un sistema che gestisce un numero considerevole di apparati creati per ogni singola stazione di collaudo.
La scheda micro dello strumento FRENO ha i seguenti componenti base:

- Micro 68HC705B16/32
- Un circuito esterno di watchdog
- Buffer a flip flop per input e output
- Buffer ad array darlinghton per il comando del circuito di potenza
- Traslatore di livello per rendere il circuito compatibile con lo standard RS232
- Circuito di alimentazione con stabilizzatore integrato.
- Integrati optoisolatori
- Relay come switch

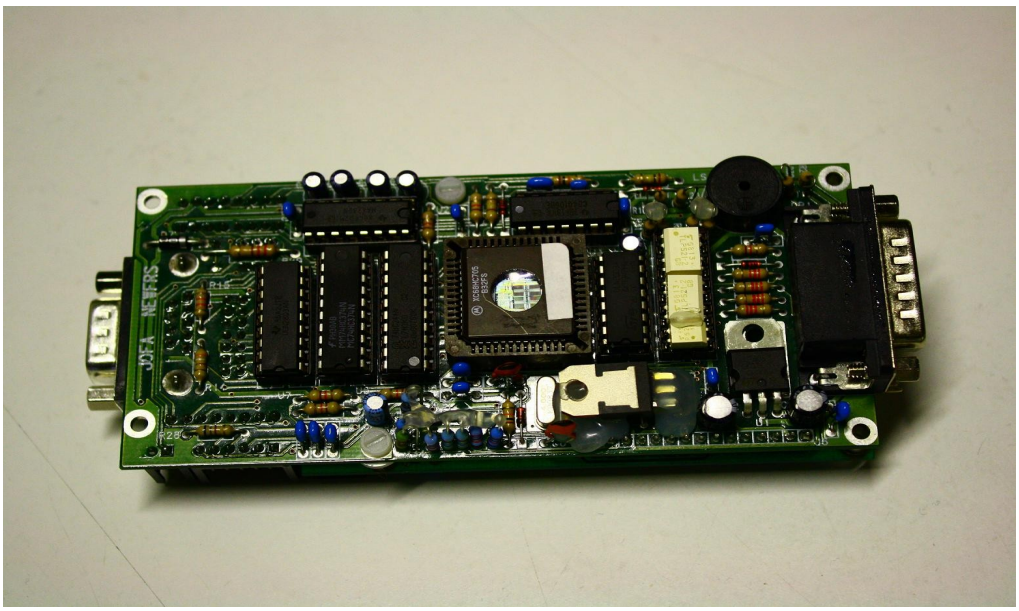


Figura 2.2.1

L'integrato mA78L05ACPK riceve in ingresso i 12V/14V delle batterie utilizzate per alimentare gli strumenti e fornisce in uscita una tensione stabile di 5V con un una corrente massima di 100mA, è dotato inoltre di limitatore per cortocircuiti e sovracorrenti. Esso è necessario per l'alimentazione della maggior parte dei componenti montati sulla scheda di controllo.
A monte dello stabilizzatore la scelta progettuale ha portato all'inserimento di un transistor pnp che funge da interruttore per l'accensione del circuito. Alla pressione del pulsante di accensione il transistor si porta in saturazione e alimenta lo stabilizzatore. Il comando di accensione viene mantenuto dal microcontrollore stesso, tramite un pin di output dedicato, la prima operazione eseguita dal micro infatti sarà quella di impostare il pin in modo da far condurre il transistor.
Come si può vedere in figura 2.2.1, si utilizza un oscillatore al quarzo esterno con frequenza caratteristica di 4 Mhz connesso in configurazione standard con 2 condensatori e una resistenza in modo da rendere stabile la frequenza in ingresso al micro

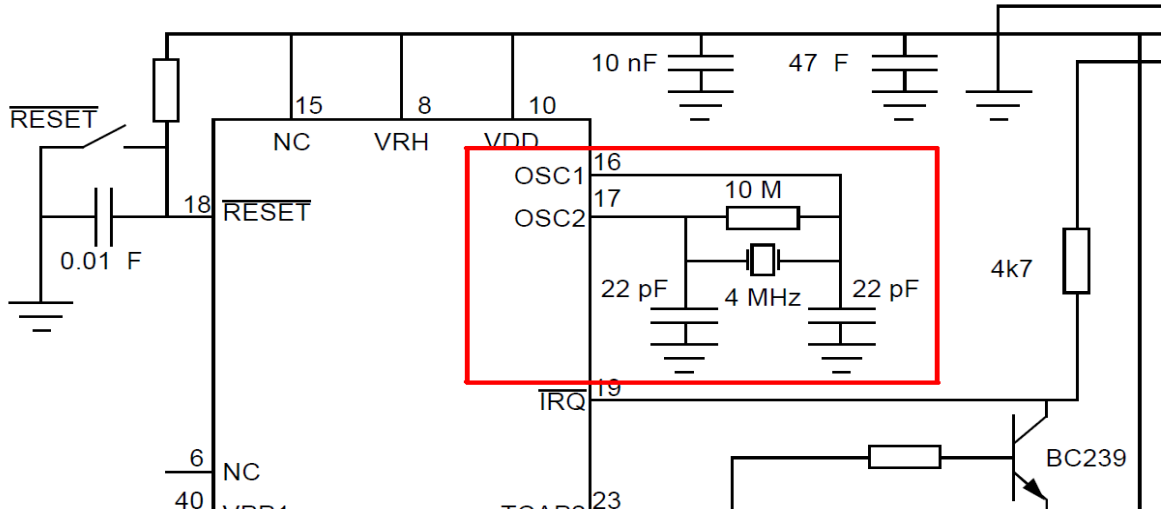


Figura 2.2.2

Il circuito di watchdog esterno è realizzato tramite un circuito astabile e una serie di porte inverter 40106 che resettano il microcontrollore nel caso l'ingresso al circuito in questione rimanga invariato nel tempo.
E' compito dell'utente invertire via software l'uscita del pin specifico per il watchdog ad una frequenza tale per cui non arrivi il comando di reset.
Si può osservare il circuito di WatchDog in figura 2.2.3.

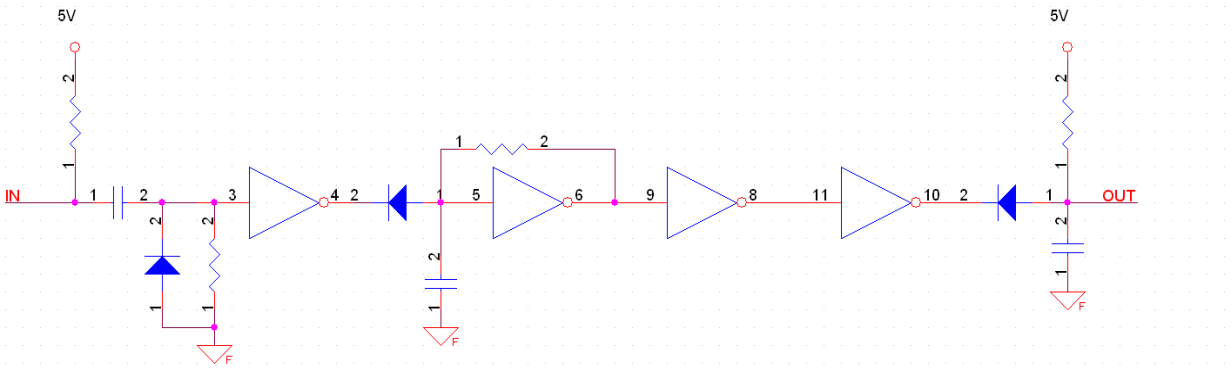


Figura 2.2.3

Il componente utilizzato per la traslazione di livello ai fini della comunicazione seriale è l'ICL232 (Figura 2.2.4)che consiste in un trasmettitore/ricevitore seriale alimentato a 5V per lo standard RS232

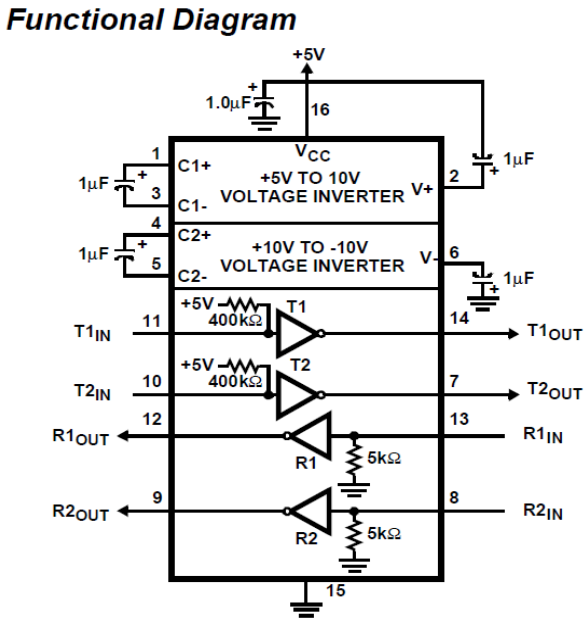


Figura 2.2.4

Lo standard di comunicazione infatti richiede i livelli +12 / -12V anziché i livelli 0 / 5V.
I circuiti di interfaccia input/output sono realizzati con i componenti 74HC374 e 74HC244.
Il primo è un buffer ottale a flip flop di tipo D (figura 2.2.5) mentre il secondo è un bus ottale con uscita 3-state (2.2.6).

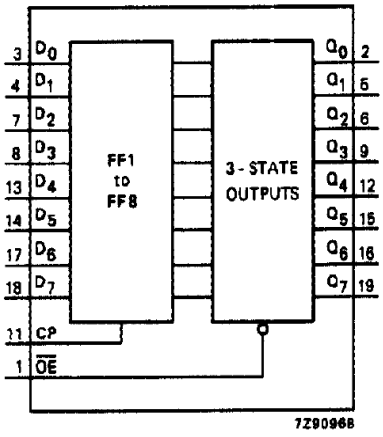


Figura 2.2.5

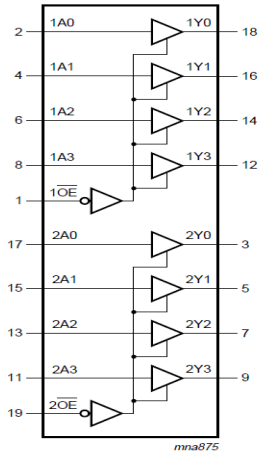


Figura 2.2.6

Per l'interfaccia con la parte di potenza occorre l'ULN2003, un array di darlinghton con corrente massima di 500mA e tensione massima open collector in configurazione ad emettitore comune di 50V in grado di comandare carichi induttivi, motori, relay.
Il diagramma funzionale è visibile in figura 2.2.7.

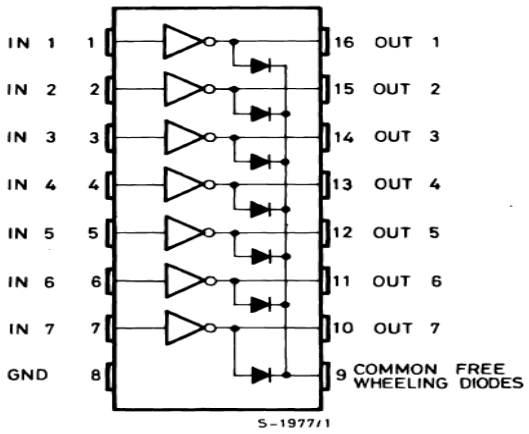


Figura 2.2.7

Le connessioni con finecorsa e altra strumentazione vengono fatte tramite optoisolatori assicurando le porte del microcontrollore da eventuali picchi di corrente o tensione nella parte di potenza.
Nel circuito si è inserito un piccolo cicalino per segnalazioni varie controllato sempre tramite l'integrato ULN2003.
La scheda comanda un motore in corrente continua per il braccio allungabile che si appoggia ai pedali.
Viene riportato in un ingresso analogico il valore della tensione di alimentazione del motore per un controllo della velocità. L'adattamento dei livelli di tensione viene effettuato tramite un partitore resistivo con 4 resistori i cui valori sono a precisione 1%.
Un relay permette di deviare il segnale della seriale ad un'antenna esterna o al connettore DB9 e di accendere o spegnere l'antenna stessa.
La scheda è predisposta per lavorare con un display a 2 linee da 16 caratteri tramite un connettore a 14 pin.

2.3 Componenti della scheda FRENO\FRIZIONE SIDCA

La scheda SIDCA si monta su attrezzi per la rilevazione di parametri con i quali poi settare altri strumenti di controllo, si utilizza per valutare eventuali difetti di un vasto numero di automatismi a bordo macchina come alzacristalli, tettucci apribili, freno a mano, pedali freno e frizione.
Lo strumento in questione è connesso con un potenziometro e una cella di carico.
Le tensioni di uscita di questi trasduttori (opportunamente adattate) vengono elaborate dal micro attraverso dei pin di ingresso analogico e la scheda invia dati al pc-host connesso in seriale che segnala eventuali malfunzionamenti o stampa report dettagliati sui vari test.
La scheda micro dello strumento COMFORT ha i seguenti componenti base:

- Micro 68HC705B16/32
- Un circuito esterno di watchdog
- Traslatore di livello per rendere il circuito compatibile con lo standard RS232
- Circuito di alimentazione con stabilizzatore integrato.

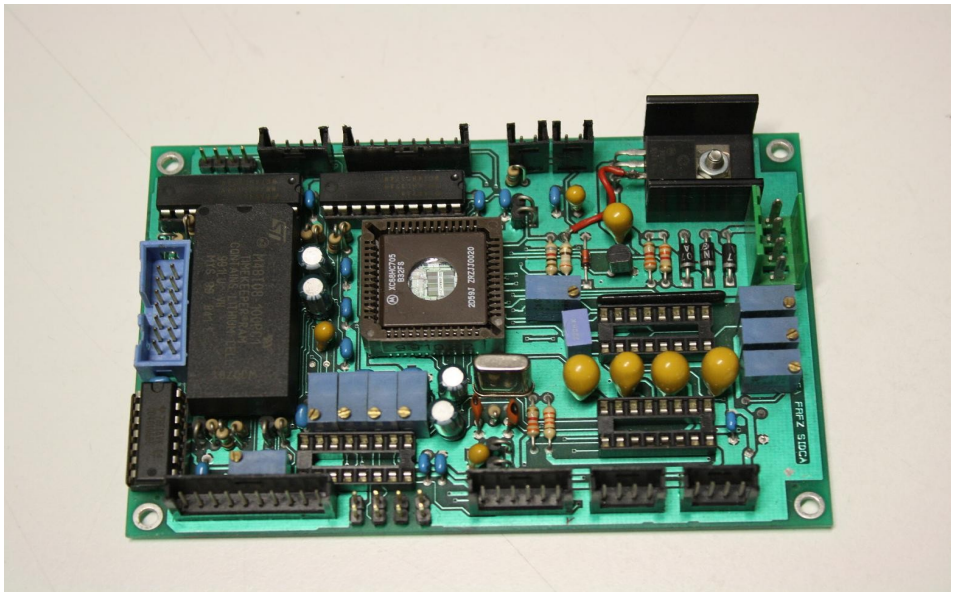


Figura 2.3.1

2.4 Programmazione

Il codice sorgente scritto con un editor di testo viene compilato e linkato tramite 2 file eseguibili: XAMS e XLNK.
La programmazione del microcontrollore avviene, come accennato in precedenza, tramite una scheda fornita da Motorola per specifici integrati.
Il codice sorgente scritto in assembly è stato fornito dall'azienda. Il programma piuttosto complesso si compone di circa 10000 righe secondo la sintassi e le direttive RASM05.
La CPU contiene 5 registri:

- Accumulator 8bit
- Index Register 8bit
- Program Counter 16bit(i 3 bit più significativi non sono utilizzati e mantenuti a 0)
- Stack Pointer 16bit
- Condition Code Register 5bit

Il codice per il microcontrollore in azienda viene sviluppato tramite un editor di testo in ambiente DOS forse un po' scomodo da utilizzare. Per questo motivo le modifiche al sorgente apportate in fase di test sono state eseguite con il supporto di un editor di testo: il PSPad, da me scelto e arricchito di un file di registro per l'evidenziazione della sintassi specifica assembly del micro.

2.5 Circuito di potenza

La parte di potenza della scheda elettronica sebbene oggetto di studio preliminare perché controllata tramite micro è sviluppata a parte e rimane la stessa da anni. Essa si interfaccia alla scheda di controllo tramite dei connettori a pettine. Sostanzialmente si tratta di un circuito che controlla la velocità e la direzione di un motore a corrente continua utilizzando 5 mosfet di potenza in configurazione a ponte, la velocità dello stesso è regolata da una apposita linea PWM. La scheda è dotata inoltre di 2 convertitori DC/DC integrati per regolare la tensione di alimentazione delle batterie e adattarla alle esigenze del circuito.

2.6 Scelta di una scheda custom

Dopo aver testato alcune tecnologie “all-in-one” di diverse case madri si è deciso di sviluppare la scheda internamente in JOFA.
Le soluzioni all-in-one hanno un elevato potenziale e un'elevata funzionalità su diversi strumenti ma il problema fondamentale oltre al costo è dato dal fatto che si è dipendenti da tecnologie che nel tempo sono destinate ad essere obsolete e quindi al ritiro dal mercato. La scheda custom essendo sviluppata e montata per ogni singolo strumento diviene più performante e modificabile nel tempo, in quanto nasce proprio per la funzione che andrà a svolgere.

Linguaggio Assembly RASM05

3.1 Codice in uso

Per lo studio del linguaggio si utilizzano 2 sorgenti distinte per 2 strumenti diversi.
I codici si compongono circa di 11000 righe in linguaggio assembly.
L'instruction set per architetture della famiglia 6805 si compone di circa 63 comandi divisi in 5 gruppi funzione: registrazione/memorizzazione, lettura/modifica/scrittura, branch, manipolazione bit e controllo e in 31 ulteriori direttive.
L'elenco completo dei comandi è visibile in appendice E.
E' necessario capire a fondo l'architettura 6805 al fine di poter sviluppare un codice efficiente, essa è organizzata in 5 registri: accumulatore e index register che sono a 8 bit e il condition code register che è a 5 bit, gli altri 2 registri, il Program counter e lo stack, hanno una grandezza variabile a seconda della versione del controllore.
Il condition code register o CC contiene dei bit di controllo indispensabili per alcune funzioni di branch ed è importante capire se e come ogni istruzione chiamata modifichi il registro.
I bit che lo compongono sono:

- Carry/borrow
  Questo bit viene settato se un riporto avviene durante l'ultima operazione aritmetica e in presenza di shift, rotazioni e istruzioni di bit test;
- Zero
  Questo bit viene settato se il risultato dell'ultima operazione di manipolazione, aritmetica o logica è zero;
- Negative
  Questo bit viene settato se il settimo bit del risultato dell'ultima istruzione è negativo
- Interrupt Mask;
  Quando questo bit viene settato l'interrupt esterno e l'interrupt del timer vengono disabilitati;
- Half Carry
  Questo bit viene settato se avviene un riporto tra i bit 3 e 4 durante le istruzioni ADD e ADC. Questa flag viene usata nelle subroutine di addizione BCD.

Vi sono vari metodi di indirizzamento: immediate, relative, extended, indexed, inherent, bit set/clear, bit test/branch e direct. Questi metodi permettono un accesso alla memoria con varie modalità a seconda dell'istruzione che viene eseguita.
Il codice in uso è il frutto di anni di perfezionamenti e di modifiche alle strumentazioni e agli stessi prodotti Jofa ed è quindi difficile fare riferimento ad ogni singolo pezzo di codice.
In generale il micro è ampiamente sfruttato per funzioni quali la lettura e conversione di segnali analogici, la comunicazione con host pc, periferiche seriali e linea I²C, comando PWM e comandi a driver di potenza.
Entrambi i codici studiati hanno struttura simile con una definizione preliminare di stringhe e variabili temporanee seguite dallo sviluppo delle varie routine e in fine la tabella dei vettori.

3.2 Codice sviluppato per la scheda COMFORT

Per apprendere e sfruttare le potenzialità del linguaggio assembly viene studiato e poi modificato il codice sorgente per la scheda COMFORT. Vengono create routine per alcuni passaggi delicati del programma quali la scrittura/lettura su RAM tampone e la visualizzazione di stringhe prelevate dalla memoria sul display 16 segmenti.
Per la scrittura su RAM il problema principale sta nel fatto che si comunica in parallelo tra il sistema micro a 8bit e il sistema memoria a 13 bit. Occorre quindi effettuare l'indirizzamento per scrittura/lettura in 2 tempi con 2 latch e 2 pin di controllo.

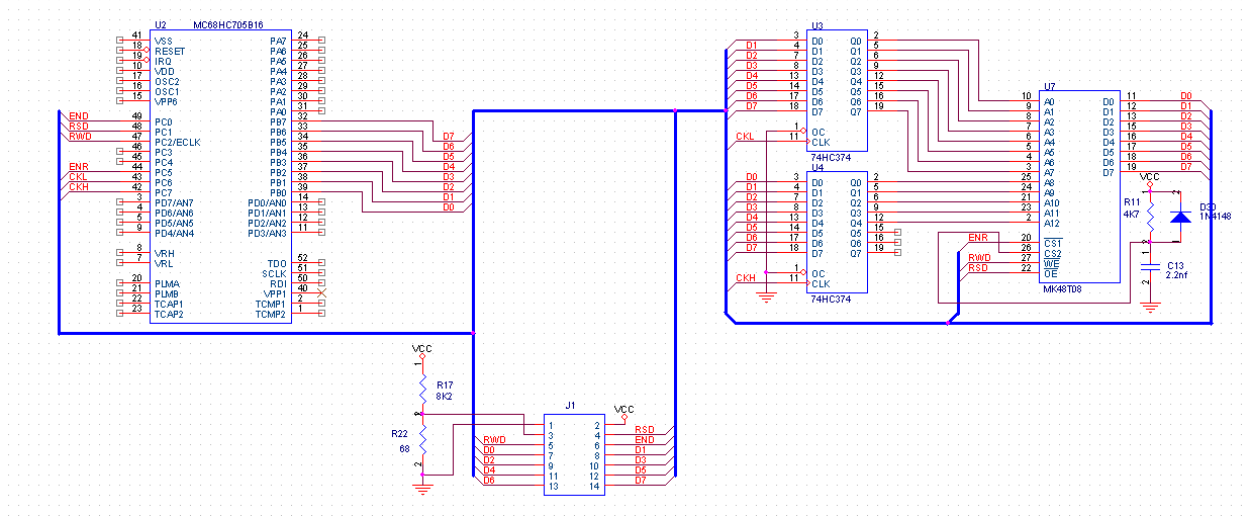


Figura 3.2.1

Come si può vedere dal circuito in appendice D relativo alla scheda COMFORT e in figura 3.2.1 disponiamo di un bus condiviso con il display 16x2 che utilizzeremo per scrittura e lettura da memoria.
Il bus comprende anche i bit per il controllo CKH, CKL, RWD, RSD, END, ENR.

3.2.1 Lettura e scrittura in RAM esterna, Tasti, Display 16x2

Si fa qui riferimento al codice presente in appendice C dove sono riportate per esteso le routine create.
Si intende far eseguire alla scheda la seguente serie di operazioni:

- Scrivere un menu con le opzioni a seconda dei tasti premuti
- attendere la pressione di un tasto
- eseguire la routine di scrittura su RAM
- eseguire la routine di lettura da RAM e visualizzazione su display

Per fare ciò utilizziamo una stringa già memorizzata nel programma contenente la scelta delle opzioni e la visualizziamo nella seconda riga del display.
Sfrutteremo le routine di visualizzazione già create da Jofa per evitare di passare troppo tempo a studiare dei componenti che verranno sostituiti nel nuovo progetto.

Avviamo la routine di controllo tasti che controlla in modo ciclico i bit legati ai pulsanti esterni di start e stop che sono connessi al micro tramite pin usati come input. Se il tasto premuto è lo start, il programma continua l'esecuzione; se il tasto premuto è lo stop l'esecuzione del programma riparte dall'inizio stampando nuovamente il menu; se non viene premuto nulla il programma cicla sul controllo tasti.
La visualizzazione sul display viene effettuata modificando gli 8 bit della parte dati del bus e settando adeguatamente i bit di controllo per attivare la scrittura.
Se viene premuto il tasto start, il micro inizia l'indirizzamento della RAM per la scrittura memorizzando la parte bassa dell'indirizzo su cui scrivere nel bus e imponendo un fronte di salita nel primo latch 74HC374. Segue quindi la scrittura della parte alta dell'indirizzo nel secondo latch. In fine viene abilitata la scrittura nella memoria attraverso gli ingressi WE, OE, CS1 e CS2. Seguono i singoli byte del dato da memorizzare e, contemporaneamente,di volta in volta la parte bassa dell'indirizzo deve essere aggiornata per passare alla successiva cella byte della memoria finché non viene generato un overflow. In quel caso la parte alta viene aggiornata e la parte bassa viene azzerata.
La lettura avviene in modo analogo: viene utilizzata la stessa tecnica per l'indirizzamento ma viene abilitato l'output della memoria e impostata in modalità di lettura modificando adeguatamente gli ingressi WE, OE, CS1 e CS2, aggiornando di volta in volta l'indirizzo e salvando il contenuto in un buffer temporaneo chiamato DSBUF. Dopo la lettura viene chiamata la visualizzazione in riga uno del display della stringa appena salvata in memoria.
La visualizzazione nel display si effettua con la chiamata a 2 possibili funzioni: WRITE1 e WRITE2 rispettivamente per scrivere sulla prima o sulla seconda riga del display.
Le funzioni utilizzano come parametri di ingresso un vettore di caratteri che verrà stampato nella riga selezionata e il numero di caratteri del vettore stesso.
Nello specifico le funzioni WRITE1 e WRITE2 muovono il cursore del display a inizio riga (a seconda della funzione nella prima o nella seconda riga) salvano il numero caratteri in una variabile contatore e inviano ogni singolo carattere al display.
Il test della scheda è stato fatto con questa configurazione e successivamente con una diversa configurazione hardware e software per interfacciare il nuovo display alla scheda COMFORT. E' stato sufficiente modificare la porta di comunicazione per la visualizzazione e sviluppare una nuova routine per la scrittura via seriale in modo tale da inviare le stringhe direttamente al display OLED.

3.2.2 Test della scheda con il nuovo software

Dopo alcuni tentativi il test del software ha dato gli esiti sperati e la visualizzazione avviene correttamente
I tempi di scrittura di un codice assembly in verità non sono elevati una volta assunta una certa disinvoltura nella stesura ma ci sono dei tempi abbastanza lunghi in fase di debug perché la densità di istruzioni per una singola routine risulta grande e piccoli errori possono essere frequenti.

3.3 Scrittura del codice

Per la scrittura del codice viene utilizzato un text editor reperibile gratuitamente al sito http://www.pspad.com/it/.

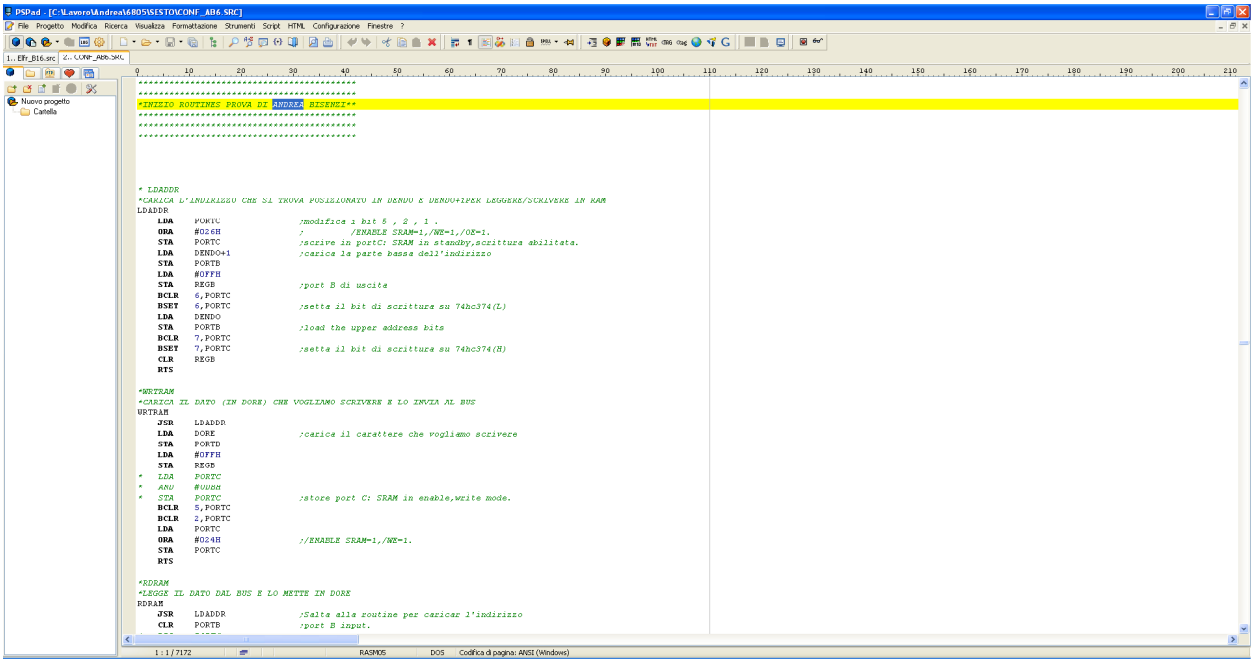


Figura 3.3.1

Questo programma permette una scrittura del codice facilitata dall'uso di diverse colorazioni per direttive, istruzioni, commenti e parole chiave relative ad un linguaggio di programmazione.
Nel caso non sia già caricato un profilo per il linguaggio di programmazione che ci interessa è possibile crearne uno di personalizzato, salvando in un file *.INI l'insieme di istruzioni da evidenziare scegliendo anche colore e font.
Si riassumono qui le principali caratteristiche:

- Gestione di Progetti
- Modifica di file di testo di Dimensione qualsiasi
- Apertura contemporanea di diversi documenti (MDI)
- Possibilità di salvare lo stato del desktop per future sessioni
- Client FTP per modificare i files in remoto (direttamente sul WEB)
- Registratore di macro per registrare, memorizzare e riprodurre sequenze di operazioni ripetitive
- Ricerca e Sostituzione di stringhe nei files
- Evidenziazione colorata delle differenze di testo
- Uso di Modelli (HTML tags, script, modelli sintattici...)
- L'installazione contiene modelli per HTML, PHP, Pascal, JScript, VBScript, MySQL, MS-Dos, Perl,...
- Evidenziazione di Sintassi definibile dall'utente, per ambienti "personalizzati"
- Riconoscimento automatico dell'Evidenziazione di Sintassi in base al tipo di file
- Correzione Automatica
- Editor ESADECIMALE completo
- Definizione di programmi esterni, in funzione dell'ambiente selezionato

- Compilatore esterno con cattura dei messaggi di uscita, finestra di log, analizzatore sintattico di log per ogni ambiente
- Stampa colorata della sintassi con anteprima di stampa
- Libreria TiDy integrata per formattare e verificare il codice HTML e per convertirlo in CSS, XML, XHTML
- Versione libera integrata dell'editor CSS TopStyle Lite
- Esportazione con sintassi evidenziata nei formati RTF, HTML, TeX in file o Appunti
- Selezione a colonna, segnalibri, numeri di linea, ...
- Riformattazione e compressione codice HTML, conversione MAIUSCOLA/minuscola dei tags
- Ordinamento delle linee in base a una colonna selezionata, con possibilità di eliminare i duplicati
- Tabella caratteri ASCII con entità HTML
- Esploratore di Codice per Pascal, INI, HTML, XML, PHP ed altri linguaggi, in futuro
- Controllo Ortografico
- Browser WEB interno con supporto per APACHE
- Evidenziazione coppie di parentesi corrispondenti

Nel nostro caso è necessario inserire nel file tutto l'instruction set relativo all'architettura della famiglia 6805.
Il file generato ha nome 6805 e permette appunto una lettura e una comprensione del codice maggiore rispetto ad un normale editor di testo standard.

Il Nuovo Circuito

4.1 Microcontrollore MC9S08DZ128

Il microcontrollore acquistato per i nuovi progetti di Jofa si compone di una CPU a 8-Bit con frequenza massima di lavoro a 40MHz e frequenza del bus massima di 20MHz.

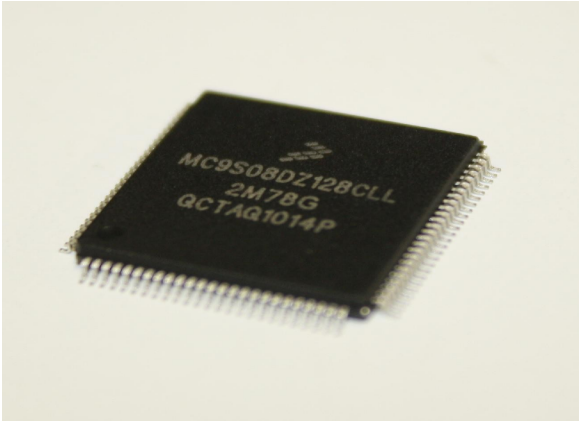


Figura 4.1.1

La scelta di questo micro comporta un grosso balzo in avanti per quanto riguarda la parte elettronica degli strumenti Jofa.
Sebbene il circuito fin qui utilizzato funzioni molto bene, è necessario aggiungere caratteristiche al prodotto finito che richiedono un calcolatore più efficiente e dalle maggiori potenzialità.
Il micro MC9S08DZ128 è stato scelto anche per la compatibilità dell'instruction set dei vecchi progetti, in quanto la scrittura e i test del codice richiedono tempi di sviluppo elevati e un comune insieme di istruzioni permette di utilizzare (previe modifiche alle definizioni) codici sorgente già presenti in archivio Jofa.

Figura 4.1.2

Le caratteristiche principali del componente per gruppi di competenza sono:

8-Bit HCS08 U (CPU)
- 40-MHz HCS08 CPU (20-MHz bus)
- HC08 instruction set with added BGND instruction
- Support for up to 32 interrupt/reset sources

On-Chip Memory
- FLASH read/program/erase over full operating voltage and temperature
- EEPROM in-circuit programmable memory; 8-byte single-page or 4-byte dual-page erase sector; Program and Erase while executing FLASH; Erase abort
- Random-access memory (RAM)

Clock Source Options
- Oscillator (XOSC) — Loop-control Pierce oscillator; Crystal or ceramic resonator range of 31.25 kHz to 38.4 kHz or 1 MHz to 16 MHz
- Multi-purpose Clock Generator (MCG) — PLL and FLL modes; reference clock with nonvolatile trim (0.2% resolution, 1.5% tolerance over temperature with internal temperature compensation); External reference with oscillator/resonator options

System Protection
- Watchdog computer operating properly (COP) reset with option to run from backup dedicated 1-kHz internal clock source or bus clock
- Low-voltage detection with reset or interrupt; selectable trip points
- Illegal opcode detection with reset
- Illegal address detection with reset
- FLASH and EEPROM block protect
- Loss-of-lock protection

Development Support
- Single-wire background debug interface
- On-chip, in-circuit emulation (ICE) with real-time bus capture

Peripherals
- ADC — 24-channel, 12-bit resolution, 2.5 µs conversion time, automatic compare function, temperature sensor, internal bandgap reference channel
- ACMPx — Two analog comparators with selectable interrupt on rising, falling, or either edge of comparator output; compare option to fixed internal bandgap reference voltage; runs in stop3 mode
- MSCAN—CAN protocol - Version 2.0 A, B; standard and extended data frames; Support for remote frames; Five receive buffers with FIFO storage scheme; Flexible identifier acceptance filters programmable as: 2 x 32-bit, 4 x 16-bit, or 8 x 8-bit
- SCIx — Two SCIs supporting LIN 2.0 Protocol and SAE J2602 protocols; Full duplex non-return to zero (NRZ); Master extended break generation; Slave extended break detection; Wakeup on active edge
- SPIx — Up to two SPIs; Full-duplex or single-wire bidirectional; Double-buffered transmit and receive; Master or Slave mode; MSB-first or LSB-first shifting
- IICx—Up to two IICs; Up to 100 kbps with maximum bus loading; Multi-master operation; Programmable slave address; Interrupt driven byte-by-byte data transfer

- TPMx — One 6-channel (TPM1), one 2-channel (TPM2) and one 4-channel (TPM3); Selectable input capture, output compare, or buffered edge- and center-aligned PWM on each channel.
- RTC—(Real-time counter) 8-bit modulus counter with binary or decimal based prescaler; Real-time clock capabilities and RTC for precise time base, time-of-day, calendar or task scheduling functions.

Input/Output

- Up to 87 general-purpose input/output (I/O) pins and 1 input-only pin
- Up to 32 interrupt pins with selectable polarity on each pin
- Hysteresis and configurable pull device on all input pins.
- Configurable slew rate and drive strength on all output pins.

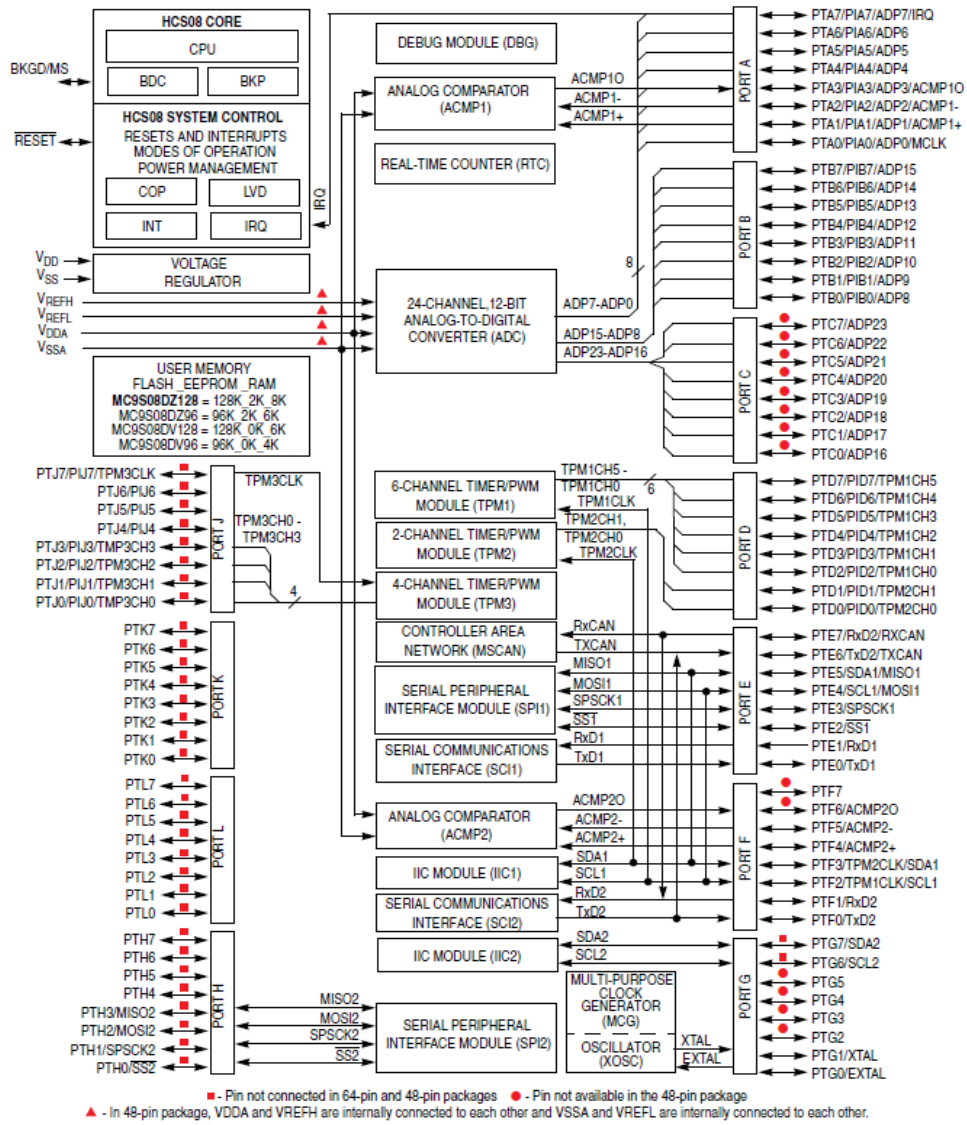


Figura 4.1.3

4.2 Nuovi componenti della scheda

La struttura di base della scheda rimane invariata rispetto alla vecchia versione ma troviamo alcuni componenti e alcune interfacce nuove.
Tra le novità troviamo:

- Connettore per la programmazione e il debug real-time:
  Un connettore che permette la programmazione del micro in modo semplice e facilita il debug tramite un'esecuzione controllata del software.

- Diodi di protezione su tutti i pin di ingresso analogico:
  Ogni ingresso analogico è stato messo in sicurezza contro tensioni negative o maggiori di 5V attraverso diodi

- Integrati Infineon BTS3408G:
  Sono switch di protezione utilizzati per comandare principalmente la parte di potenza ma anche per il buzzer e per il controllo dei relay di selezione

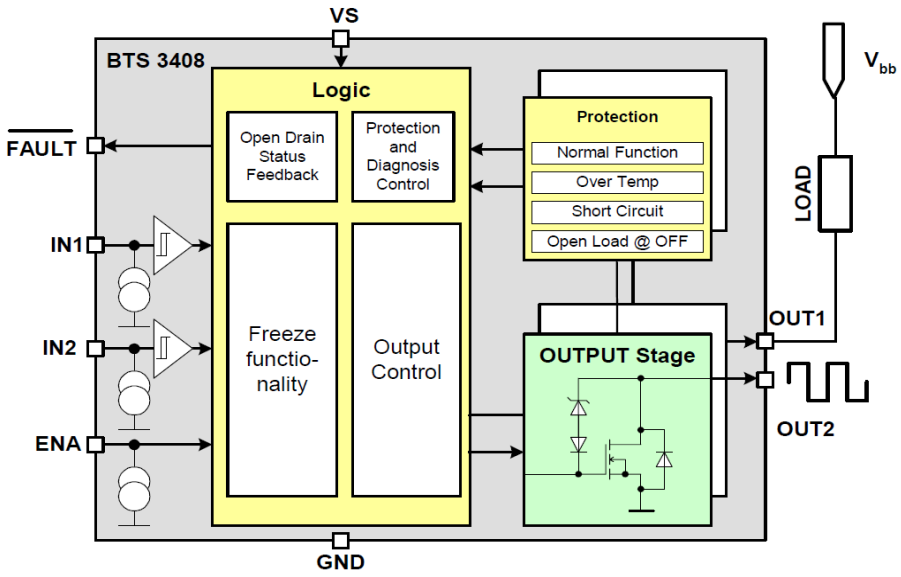


Figura 4.2.1

- Integrato max3232 in sostituzione dell'ICL232:
  E' un driver/ricevitore di linea multicanale, interfaccia una seriale a 5/3,3V con una seriale con standard RS232

- Ulteriore stabilizzatore integrato per non sovraccaricare il primo:
  L'alimentazione del display OLED richiede mediamente 90mA, per questo si è deciso di inserire uno stabilizzatore dedicato

- Connettore display OLED e circuito di interfaccia:
  Il display OLED comunica attraverso una linea seriale con livello alto a 3.3V, è stata quindi inserita una coppia di inverter HTC per ripristinare il segnale a 5V.

- Connettore e relay di selezione per la periferica Barcode:
  I nuovi strumenti montano un lettore di Barcode secondo le richieste di Fiat per il controllo del processo produttivo

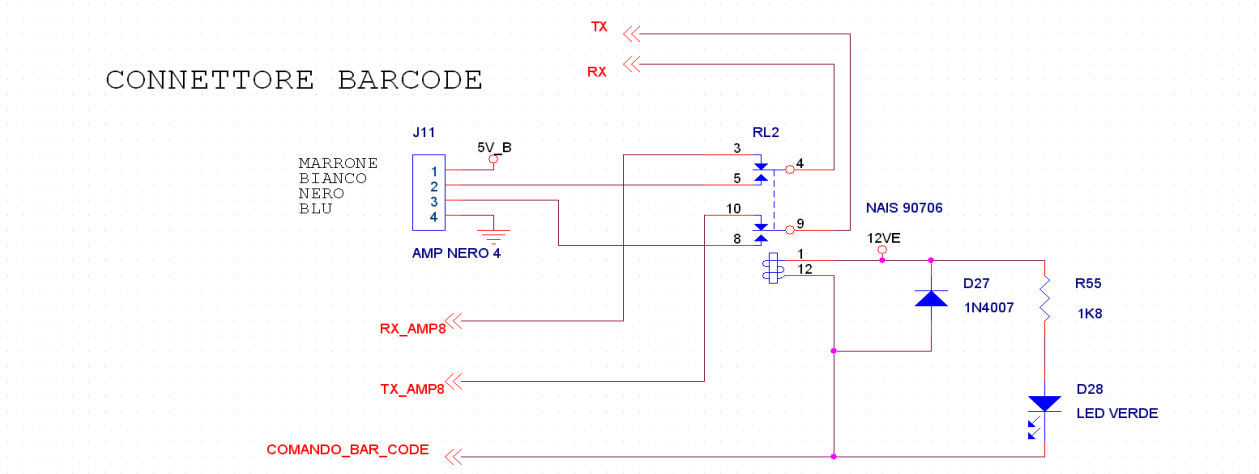


Figura 4.2.2

4.3 SMT e passaggio al multistrato

Il gran numero di componenti e le nuove interfacce rendono necessario un passaggio quasi integrale alla tecnologia di montaggio superficiale.
La scheda progettata infatti deve rispettare una determinata geometria per essere facilmente montata nei vecchi strumenti. Il passaggio al montaggio superficiale non è totale, sono stati lasciati in montaggio tradizionale gli stabilizzatori, i relay, il buzzer, gli optoisolatori, i led, i connettori e alcuni condensatori.
Purtroppo la scelta seppur necessaria e innovativa rende meno “flessibile” la scheda.
Infatti per applicazioni su strumenti custom spesso è utile sostituire componenti o modificare il circuito anche in occasione di interventi di riparazione: mentre ad oggi eventuali riparazioni vengono effettuate anche dagli stessi clienti con semplicità, con il passaggio ad un montaggio superficiale con componenti con passo (distanza tra i pin) di 0.5mm la sostituzione diventa difficile se non impossibile.
Il collegamento dei componenti alla scheda viene lasciato ad una ditta esterna di cui Jofa è cliente: la Elpi elettronica di Padova.
Per la parte SMD l'azienda richiede un file CAD con le specifiche geometriche della scheda dotato delle posizioni dei componenti e una lamina specifica per il deposito di pasta saldante. I valori vengono gestiti da un posizionatore automatizzato che poi invia al forno per eseguire la saldatura.
Oltre al passaggio all'SMD la ditta per il disegno del master ci ha consigliato un passaggio al multistrato con una notevole riduzione di eventuali disturbi nati da un affollamento eccessivo di piste nel tradizionale circuito a 2 facce.

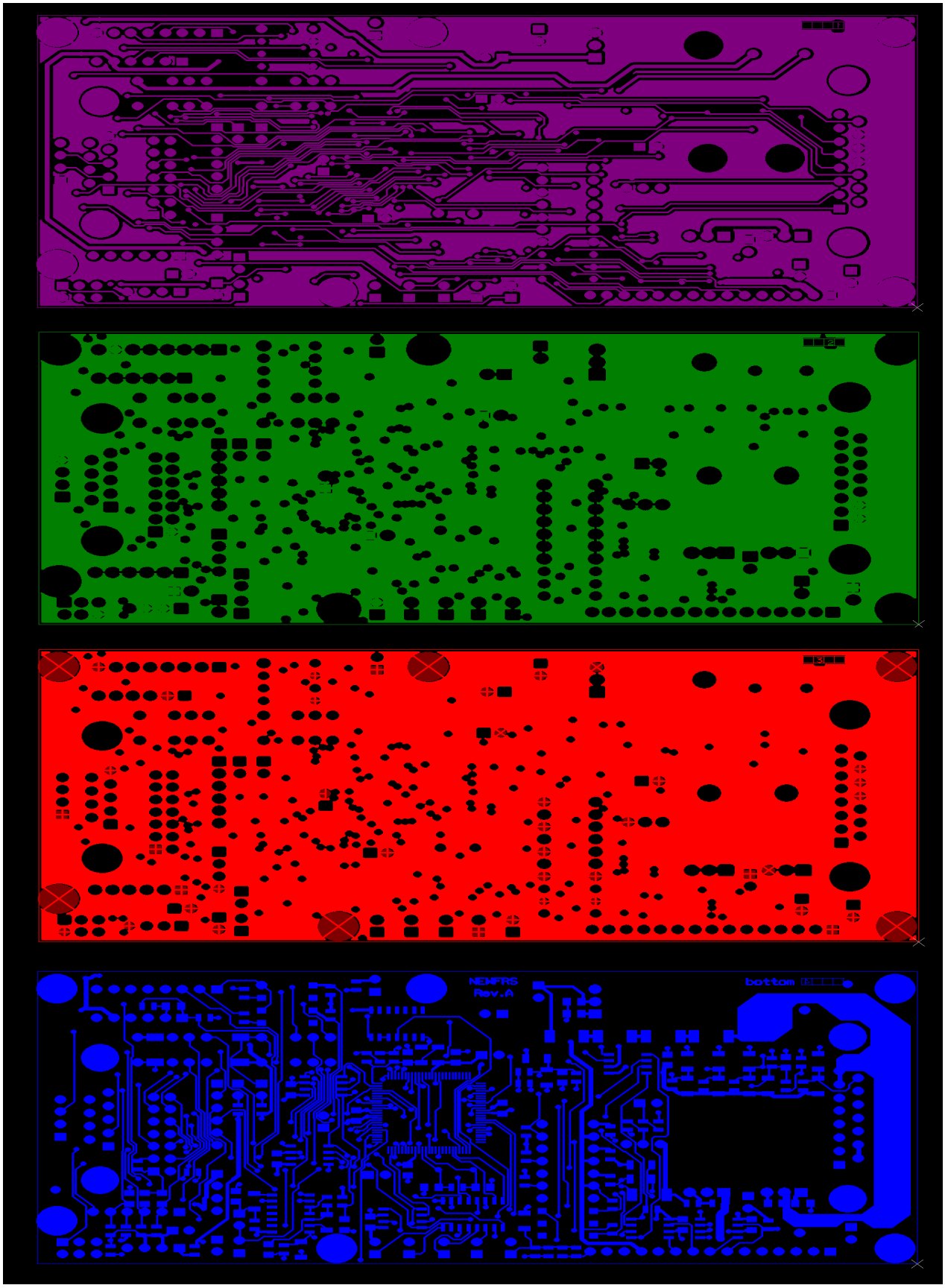


Figura 4.3.1

4.4 Programmazione

Mentre in precedenza il micro veniva programmato attraverso una scheda dedicata e a componente scollegato, ora la programmazione avviene attraverso un connettore a 6 pin come mostrato in figura.

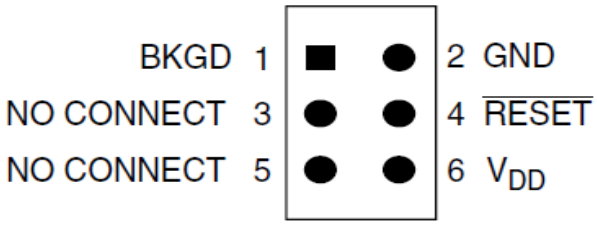


Figura 4.4.1

Il pin BKGD realizza una comunicazione seriale bidirezionale che scambia istruzioni e dati attraverso un protocollo custom descritto nella sezione “Communication Details” del datasheet relativo al Microcontrollore.

Figura 4.4.2

La connessione con l'host PC tramite l'interfaccia in figura 4.4.2 eseguita in questo modo permette non solo la programmazione con la scheda alimentata e non (se la scheda non è alimentata esiste in commercio un programmatore che provvede a fornire l'alimentazione), ma permette di effettuare un'esecuzione controllata del software (background active mode) o di controllare un'esecuzione automatica del software caricato in flash. Nel caso un host richieda di comunicare con una MCU con un segnale di clock non definito, un segnale di sincronizzazione viene inviato sempre tramite il pin BKGD per determinare la velocità di comunicazione.
La linea di reset permette di resettare il micro e serve anch'essa per l'invio di istruzioni e per il sincronismo.
Il codice sorgente e il codice generato vengono sviluppati tramite un IDE, il Codewarrior versione 5.9.0.

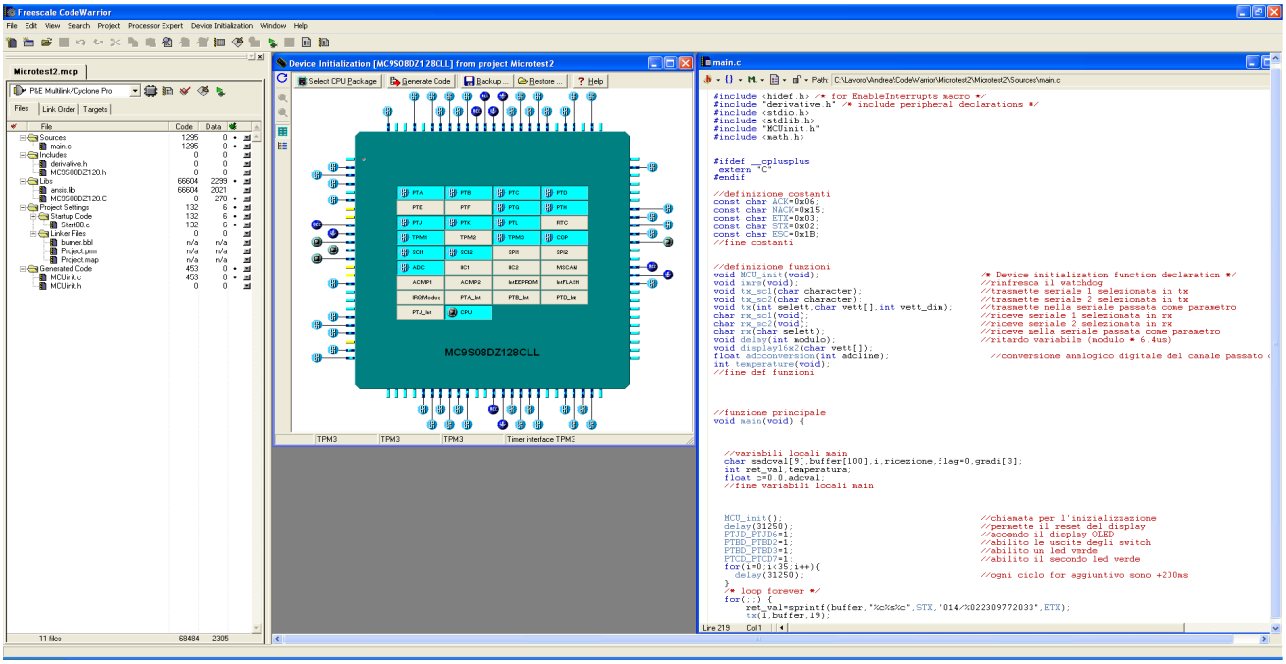


Figura 4.4.3

L'IDE è in grado di compilare e linkare sorgenti in C,C++, assembly (compatibile con l'instruction set RASM05) e codice misto. Noi lavoreremo con il linguaggio C per i test ma la sostituzione del codice avverrà in modo graduale (verrà modificato il codice assembly utilizzato sui micro MC68HC705). Questo IDE risulta molto pratico grazie anche ad un applicativo che permette di generare i file MCUnit.c e MCUnit.h che rappresentano l'inizializzazione del microcontrollore. In essi è contenuto il codice per le prime istruzioni di abilitazione e di impostazione per i registri di tutte le periferiche interne.
Le operazioni di inizializzazione e configurazione sono cosi sviluppate in automatico dall'IDE dopo aver compilato delle schede specifiche per ogni periferica in un ambiente grafico peraltro anche molto intuitivo.
Il file generato comprende anche i vettori di inizializzazione e di reset necessari all'avvio del programma.
Il programma completo è riportato in appendice B.
Il primo codice generato controlla alcuni pin della scheda per accendere/spegnere dei led, controlla una trasmissione seriale per la scrittura di stringhe sul display e accende/spegne un buzzer.
Vengono anche scritte la routine imre(), che “rinfresca” il segnale in uscita al pin PTL3 relativo al circuito di watchdog, e la routine delay(), che utilizza il contatore TPM1.
La routine delay() accetta come parametro un numero che sarà il modulo del contatore (il numero di conteggi che genera overflow) la quale richiamando al suo interno la funzione imre() e controllando la flag di overflow in polling, genera un ritardo.
E' importante aggiornare questo segnale per far in modo di non resettare il micro stesso; abbiamo analizzato nello specifico il circuito di watchdog nel paragrafo 2.2.
Procedendo con la programmazione si è verificato il corretto funzionamento della maggior parte delle periferiche on-board alla scheda o al micro.
Il secondo collaudo riguarda la comunicazione seriale del micro; la linea scelta per il test è la stessa dedicata alla comunicazione con il display OLED e lo stesso verrà utilizzato per verificare il corretto funzionamento della trasmissione.
Il modulo seriale del micro viene fatto operare a 9615,39 baud con un formato dati di 8bit, un bit di start e un bit di stop.

La routine per la comunicazione viene scritta seguendo lo schema:

- controllo buffer di trasmissione, attende che sia vuoto
- chiamata alla routine di watchdog
- copia del byte nel buffer di invio
- attesa per il completamento della trasmissione

Il codice è stato scritto con una funzione comune a entrambe le linee seriali del micro in modo da uniformare il codice sorgente; è quindi possibile selezionare una delle due linee semplicemente passando come parametro il numero 1 o il numero 2 rispettivamente per la linea 1 o la linea 2.
Questa funzione, che prende il nome di tx(), accetta come altri parametri un buffer contenente i byte da inviare come caratteri e la dimensione dello stesso.
Secondo lo stesso principio sono state scritte le funzioni per la ricezione.
Nel passo successivo abbiamo verificato il corretto funzionamento del convertitore analogico digitale, molto importante per le operazioni che il circuito è destinato a eseguire.
L'ADC a 8/12 bit è un convertitore ad approssimazioni successive in grado di lavorare con varie sorgenti di clock. Scegliamo di convertire segnali analogici con una frequenza di 2,5KHz in modalità a singola conversione.
La funzione di conversione scritta è unica per tutte le 28 linee di ingresso analogico, accetta come parametro il numero corrispondente alla linea di nostro interesse e restituisce il valore come numero di quanti corrispondenti alla conversione. E' compito del programmatore convertire in tensione (o nella grandezza che ha generato il segnale) il parametro restituito dalla funzione.
Un ulteriore routine creata è relativa alla linea di trasmissione I²C, utilizzata negli strumenti per comunicare con più periferiche tramite un unico bus a 2 linee.
I parametri scelti sono l'indirizzamento a 7 bit e la frequenza di trasmissione di 125KHz.
La comunicazione avviene con una memoria esterna della Microchip I²C seriale a 16Kbit cancellabile elettronicamente.
La funzione implementata usa il seguente schema:

- abilitazione modulo I²C
- imposto la trasmissione e master mode
- invio l'indirizzo di memoria con la selezione del pacchetto e il bit di scrittura
- inizio ad inviare i byte dell'informazione da scrivere
- attendendo la memorizzazione del byte inviato
- continuo con i successivi
- invio un segnale di stop per la conclusione della comunicazione e disabilito il master mode.

E' stata scritta un'ulteriore funzione, esclusivamente per fini didattici, per utilizzare il sensore di temperatura interno al micro.
Per usare questa periferica la funzione creata deve:

- Configurare l'ADC in “long sample” con una frequenza massima di 1MHz
- Convertire il riferimento di tensione bandgap canale 27
- convertire il canale 26 relativo al sensore di temperatura
- calcolare la temperatura esatta in gradi centigradi secondo l'equazione Eqn 10-1 del datasheet per il microcontrollore.

4.5 Semplificazioni necessarie

Per motivi di spazio, e al fine di evitare la generazione di rumore o disturbi, si è stati costretti a modificare il circuito. In particolare si è cercato di spostare i connettori AMP perché necessariamente occupano entrambe le facce visibili della scheda rendendo inutilizzabile una delle due facce per il posizionamento dei componenti e comprimendo fra loro le piste.
La prima modifica sostanziale effettuata riguarda il circuito di interfaccia con il display.
La configurazione da noi scelta realizzava la traslazione di livello da 5V a 3.3V attraverso un regolatore di tensione e un traslatore di livello per una linea seriale. Il primo serviva esclusivamente per generare un riferimento di tensione utile al traslatore di linea.
Abbiamo quindi scelto una configurazione che limitasse lo spazio occupato dai componenti per dare “respiro” alle piste già abbastanza fitte.
La seconda modifica riguarda l'inserimento di un bar code e relativo circuito con un relay per l'attivazione e la selezione della linea di comunicazione.

Il Display uOLED-32028-P1T

5.1 Presentazione Display 4D System

4D System realizza display LCD per l'elettronica embedded, con caratteristiche avanzate e specializzate tali da agevolare l'integrazione dei propri display in tutti i progetti in cui si necessita di un'interazione visiva.
La gamma di Display LCD è estremamente vasta e copre le più svariate esigenze. Si parte con display piccoli di soli 0,96" idonei in tutti i contesti dove si devono conservare rigorosamente le ridotte dimensioni, si giunge a soluzioni molto più permissive con una diagonale di 2,83", touch-screen sensor e caratteristiche tecniche avanzate. Tutti i display hanno elevate profondità di colore selezionabile in molti casi tra 256 e 262k, true-to-life-colours, enhanced AMOLED screen, permettendo riproduzioni di immagini molto fedeli, il contrasto tra bianco e nero è ottimo essendo assente la retroilluminazione, la scala dei grigi è lineare e non soggetta a false approssimazioni con la riproduzione di un simil-nero, in molti casi grigio scuro, ed infine la disponibilità per tutti gli utenti dell'ambienti di programmazione integrato IDE e del linguaggio 4DGL, disponibile gratuitamente, rende molto interessante questi display noti anche per le elevate capacità logiche di cui sono dotati.
La 4D System concentrando i propri sforzi sull'innovazione ha reso i propri display molto competitivi, sopratutto in termini di funzionalità e di dimensioni, a beneficio dell'applicazione finale, diversamente improponibile con i display TFT legati strettamente ad una fonte permanente di retroilluminazione, e quindi da circuiti di step-up e pilotaggio delle tensioni, spesso molto elevate.

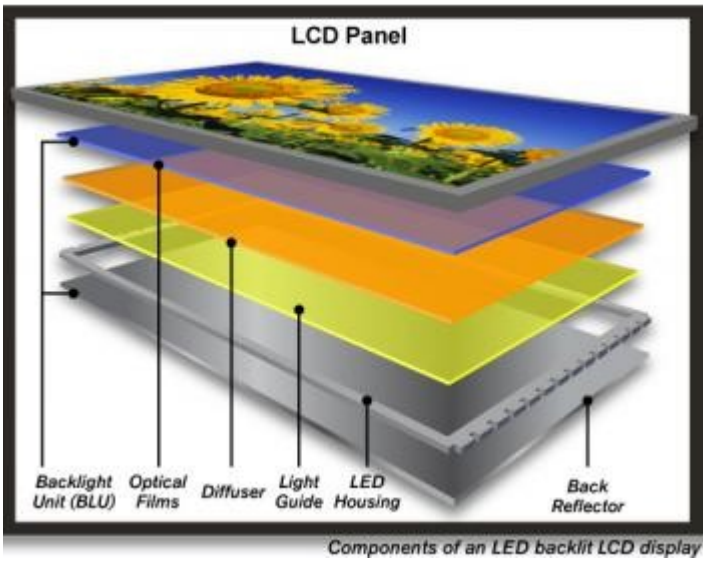


Figura 5.1.1

Con la tecnologia OLED e sue derivati è possibile eliminare la retroilluminazione permanente, fornendo ai pixel la capacità di emettere luce, ciò implica anche l'assenza di circuiti complessi e specializzati che contribuisce a ridurne le dimensioni in pochi millimetri di spessore, in più l'assenza di lampade fluorescenti permette di lavorare con tensioni dell'ordine delle decine di volt, spesso solo 5V sono sufficienti, di conseguenza meno pericolose in caso di guasti.

5.2 Panoramica sulla tecnologia OLED

I display della 4D System sono notoriamente realizzati con tecnologia OLED, dove l'acronimo sta per Organic Light Emitting Diode, nel dettaglio Light Emitting Diode non deve trarre in inganno, i display OLED non sono realizzati con LED o con matrici di minuscoli LED, il termine è impropriamente adoperato. Il fascio di luce è generato da sostanze organiche, più precisamente da sostanze plastiche, rese conduttive con particolari processi che rendono la natura della struttura cristallina mono-polare, conferendogli in questo modo la capacità di condurre corrente solo in un verso. Il comportamento ottenuto è analogo a un diodo e per questo motivo i display realizzati con strutture organiche prendono il nome generale di O-LED, ricordando la similitudine con i LED, diversamente realizzati con silicio drogato.

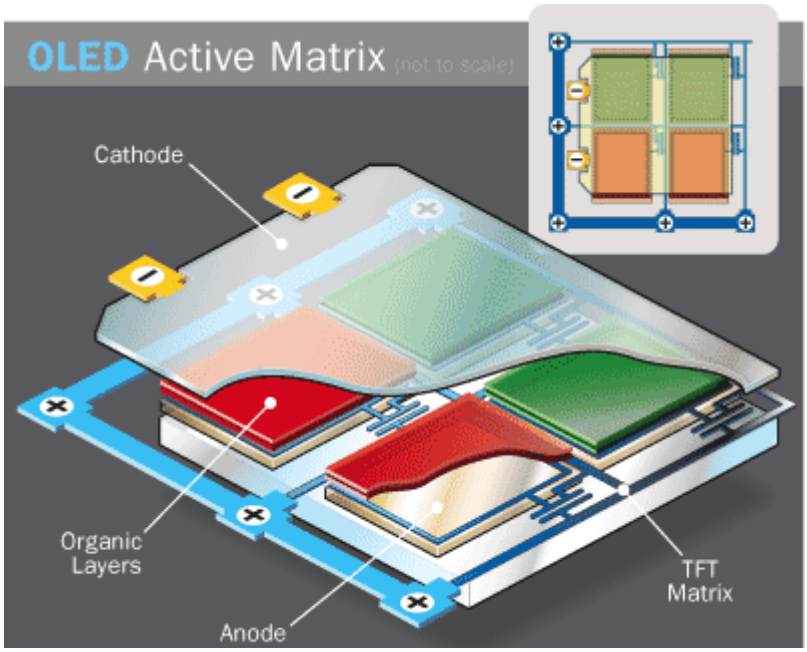


Figura 5.2.1

Il termine OLED, nella pratica è molto ampio, esso racchiude un grande insieme di famiglie di display realizzati con la capacità di emettere luce propria, questi possono anche essere definiti attivi, differenziandoli dai TFT dove il pannello ha un comportamento puramente passivo, filtrando e polarizzando il fascio di luce. Nella grande famiglie degli OLED nascono diversi sotto prodotti, differenti per tipo di materiale organico o per tipologia di matrice costruttiva, utilizzata per raggruppare i pixel, ad esempio sulla differenza di materiale organico si possono trovare queste sigle P-LED, PHO-LED, SM-LED e così via, distinguendo per lo più le tecniche produttive o il tipo di legame “drogaggio” tra gli elementi. Allo stesso tempo si hanno distinzioni tra il tipo di matrice, passiva o attiva, in questo campo una classe molto importante, derivante dalla famiglia degli OLED, è quella degli AMOLED, questi si differenziano per avere una matrice attiva, ovvero una matrice con integrati almeno due transistor per pixel, che conferisce la possibilità di pilotare autonomamente il singolo pixel tra una scansione e l'altra, come risultato si hanno immagini ben definite e molto luminose.
Con il termine Display si intende essenzialmente un quadrante di varia natura utile per comunicare delle informazioni. In elettronica la necessità di comunicare informazioni all'utente è molto sentita si potrebbe affermare con buona approssimazione che tutta l'interazione uomo-macchina si basa sulla comunicazione per mezzo visivo.

Le potenze di calcolo maggiori, le necessità di comunicare più informazioni possibili in un sol colpo e la necessità di modellare gli oggetti elettronici secondo l'ambiente che li circonda, ha spinto l'evoluzione verso schermi di visualizzazione sempre più performanti, coprendo in molti casi la totalità dell'oggetto, basti pensare agli ultimi cellulari dove il display è poco più piccolo dell'intera superficie, le performance in certi versi possono essere considerate come le capacità di adattare un componente elettronico, fornendogli semplicità d'uso e capacità espressive, chi di noi oggi si sognerebbe di usare un telefono privo di display LCD colorato, oggi anche i sistemi più banali, necessitano di comunicare e rendere interattiva la propria presenza, basti pensare come molti strumenti che sono stati per anni puramente meccanici, oggi si affacciano sul mondo dell'elettronica programmata e di conseguenza richiedono display multifunzionali per comunicare, esempi classici sono la lavatrice, il forno, il frigo e così via sono tutti elettrodomestici dove oggi l'utente si aspetta interazione, in più questi meccanismi di interazione non devono essere invasivi.

Figura 5.2.2

La prima caratteristica richiesta diventa la capacità accessibilità e familiarità con l'oggetto.
La storia dell'evoluzione dei display nasce proprio dai presupposti descritti fin ora, il termine display ha catturato in pieno questi aspetti e oggi pronunciando la parola display si immagina facilmente un display LCD super sottile da migliaia di colori. È bene conoscere qualche dettaglio in più a tal proposito, i display non deve necessariamente individuare dispositivi LCD, spesso l'uso comune tende a confondere i due significati, dimenticando che display racchiude una famiglia molto vasta, come il tubo catodico, il proiettore e tra questi i ben famigerati LCD.
Ovviamente lo schermo LCD collegato al computer o l'ultima televisione super-sottile sono frutto di anni di evoluzione in ambito tecnologico, riguardando un po' la storia si scopre facilmente che i display LCD per arrivare ad essere colorati e super-sottili e dalle mille dimensioni ne hanno fatta di strada, basti ricordare la comparsa dei primi display LCD in applicazioni low-cost, come ad esempio calcolatrici ed agli orologi da polso, dove il compito del display era molto semplice, la funzionalità era solo quella di mostrare numeri o simboli prestabiliti, la logica di controllo aveva il compito di interagire con pochissimi “pixel” o segmenti nel caso di display numerici, la scarsità di pixel ed il colore monocromatico permetteva di usare logiche molto scarne con capacità di calcolo limitate. Le applicazioni embedded spinte da nuovi micro controllori sono state le prime a sentire la necessità di comunicare numeri e simboli, per questo sono stati introdotti a buon mercato i classici HD44780, display molto versatili di tipo alpha-numerico semplici da programmare. È compito della logica interna convertire il carattere ASCII nella rappresentazione visiva sulla matrice di pixel. Questi display nonostante il basso costo e la semplicità d'uso si sono presto rilevati obsoleti in situazioni più complesse, ed infine l'assenza di capacità grafiche ne ha limitato l'uso sconsigliandolo in nuove applicazioni.

Oggi i display grafici colorati si trovano in moltissimi settori e la loro presenza è favorita dai costi accessibili a molte applicazioni, il problema che continua ancora a sussistere è la scarsa integrabilità in termini di dimensioni dovute per lo più alla presenza di particolari circuiti di pilotaggio, problema sorvolato dai display della 4D System con costruzione OLED.
Senza descrivere nel dettaglio il funzionamento di un display LCD, si può facilmente riassumere che esso è un oggetto passivo, differentemente da quanto capitava con i tubi a raggi catodici, dove il fascio di elettroni colpendo lo schermo fluorescente generava fasci luminosi, il pannello LCD ha un funzionamento opposto, non è in grado di emettere luce bensì può solo diventare opaco alla sorgente retrostante, per questo motivo tutti i display hanno bisogno della retroilluminazione, questa spesso viene fornita da lampada ad alta tensione, o in caso i display siano molto piccoli da LED permettendo di avere tensioni di esercizio non eccessive, quindi anche rischi minori. La presenza della retroilluminazione non è la soluzione ottimale, prima di tutto aumenta le dimensioni del display che per forza di cose dovrà essere più spesso, secondo la retroilluminazione dovrà essere sempre fissa il che comporta consumi costanti anche quando si sta riproducendo solo il colore nero, tutte queste esigenze hanno spinto allo studio di nuove tecniche in grado di far generare autonomamente ad ogni pixel la luce necessaria per ricreare l'immagine sul display. É il caso della tecnologia OLED che sfrutta le proprietà di materiali organici per emettere luce, eliminando per sempre la necessità della retroilluminazione, e di complessi circuiti di pilotaggio.

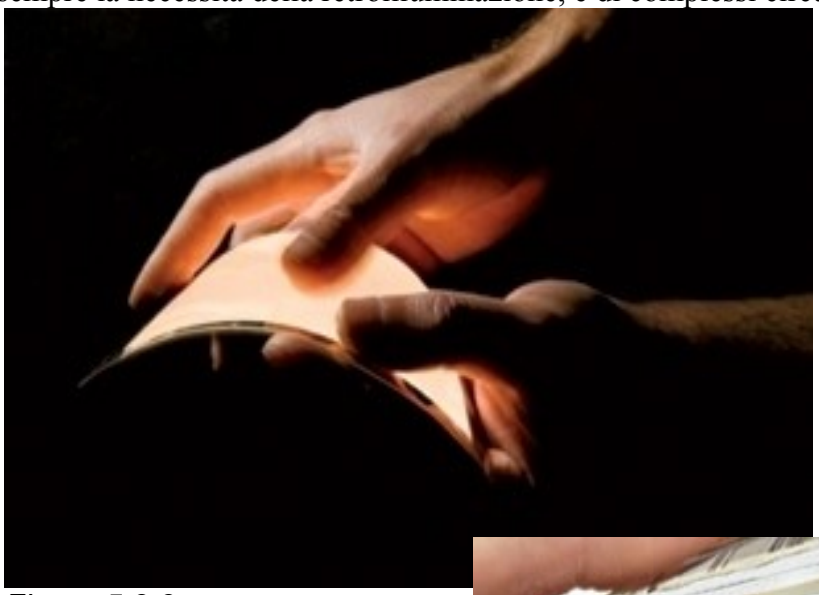


Figura 5.2.3

Figura 5.2.4

5.3 Il Display

Come già detto si è scelto sostituire il vecchio display a due righe e sedici caratteri con un display della 4D Systems, il uOLED-32028P1T.

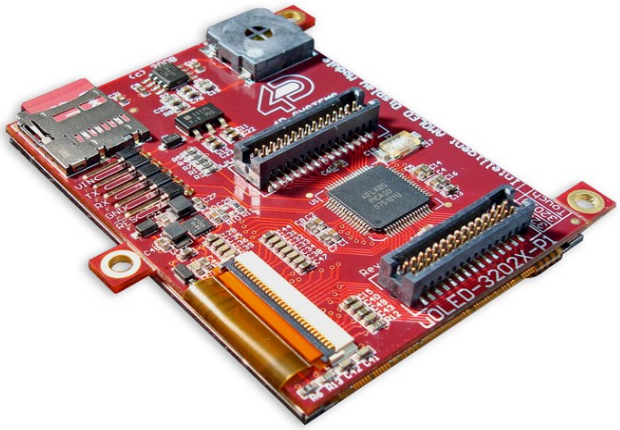


Figura 5.3.1

Il display appena citato si compone di uno schermo a matrice attiva a led(AMOLED) con un controller grafico PICASO-GFX che ne permette un utilizzo altamente funzionale, grazie anche al vasto numero di funzioni di libreria.

Figura 5.3.3

Il display viene programmato con un linguaggio orientato al grafico simile a molti linguaggi di programmazione popolari come il C il VISUAL e il Pascal e controllato nonché interfacciato al sistema che lo utilizza tramite la porta seriale.
Con questa scelta garantiamo l'indipendenza della scheda di controllo dalla periferica display. Questo dettaglio risulta molto importante in quanto la scheda di controllo non deve dipendere da tecnologie commerciali come questo tipo di display, destinate ad un processo di evoluzione tecnologica molto più rapido dei sistemi di controllo che si vanno ad implementare.
La periferica ha altri sistemi di interfaccia oltre la seriale, quali un lettore di schede uSD e dei connettori 30 pin per espansioni e interfacce alternative.
Lo sviluppo del software destinato alla periferica avviene tramite un IDE reperibile gratuitamente al sito della casa produttrice del display stesso.

Mentre il display è, come già detto, funzionale, i software per lo sviluppo del codice anche se semplici e intuitivi da usare non hanno dei sistemi di debug che renderebbero la scrittura e la revisione del software più immediata.

Figura 5.3.3

Il download del codice nel modulo uOLED-32028-P1T avviene tramite un connettore USB (venduto separatamente) figura 5.1.3.
Il dispositivo testato è anche dotato di uno schermo touchscreen che studieremo esclusivamente ai fini di prendere manualità con la stesura del codice, non verrà utilizzato infatti negli strumenti creati da JOFA in quanto non risulterebbe compatibile con le sollecitazioni applicate in fase di utilizzo.

Specifiche Tecniche

- Diagonal : 2.83”
- Screen Outline : 49.1 x 67.3 mm
- Active Area : 43.2 x 57.6 mm
- PCB Size: 49.1 x 67.3 x 11.0mm
- Resistive touch screen
- Easy 5 pin user interface (VCC, TX, RX, GND, RESET) to any 4D micro-USB module such as the µUSB-MB5 or the µUSB-CE5.
- Voltage supply from 4.5V to 5.5V, current @ 90mA nominal when using a 5.0V supply source
- Onboard micro-SD (µSD) memory card adaptor with full FAT16 file support for storing and executing 4DGL programs, files, icons, images, animations, video clips and audio wave files. 64Mb to 2Gig µSD memory cards can be purchased separately
- 2 x 30 pin headers for I/O expansion and future plug-in daughter boards
- Audio amplifier with a tiny 8 Ohms speaker for sound generation and wave file playback.
- QVGA 240 x RGB x 320 pixel resolution with 256, 65K or 262K true to life colours enhanced AMOLED screen.
- RAM Size : 1280 signed words (2560 bytes), CODE Size : 12288 bytes

5.4 Le problematiche di interfacciamento

La connessione con l'esterno avviene come abbiamo detto attraverso la linea seriale e nello specifico tramite un connettore lineare a cinque poli.

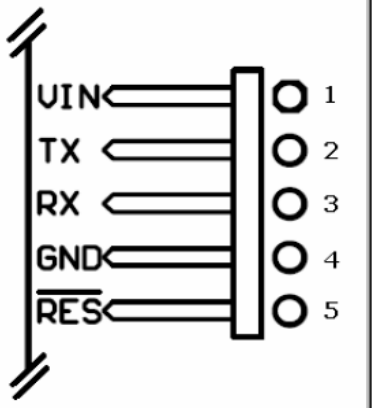
Power, Serial and micro-USB Interface			
Pin	**Function**	**Description**	
1	VIN	Main Power Supply input 4.5Volts to 5.5Volts. Nominal @ 5Volts.	
2	TX	Serial Transmit Pin (Data Out), COM0 TX. CMOS levels 0V to 3.3V	
3	RX	Serial Receive Pin (Data In), COM0 RX. CMOS levels 0V to VIN.	
4	GND	Ground.	
5	RES	External RESET signal for the module and PICASO chip. Pull this pin Low for 20µsec or longer to Reset the module. Not required for normal usage.	

Figura 5.4.1

I livelli di tensione in ricezione e per l'alimentazione non sono per noi un problema, in quanto risutano totalmente compatibili con il nostro sistema a 5V.
Il livello alto di tensione in uscita dal display invece risulta essere di 3,3V e non è quindi compatibile con il micro da noi utilizzato. La prima scelta da noi effettuata è stata quella di utilizzare degli integrati specifici per la traslazione del livello di tensione sia per l'alimentazione che per i livelli della linea seriale.
Per la ricerca dei componenti si sono consultati i cataloghi RS e DISTRELEC.
Gli integrati scelti per il circuito sono il MIC5205-3.3 e il PCA9603 che consentono rispettivamente di avere un riferimento di 3.3V e di traslare dal livello di 5V al livello di 3.3V e viceversa nella comunicazione seriale.
Il MIC5205-3.3 è un regolatore di tensione che accetta in ingresso una tensione compresa tra 2.5V e 16V e genera in uscita un riferimento di tensione di 3.3V con corrente massima di 320mA
il PCA9603 è un traslatore di livello per linee seriali I²C e SMBus compatibile con le nostre applicazioni. Sebbene con un discreto numero di resistori esterni adatta linee seriali di varie tensioni con una linea a 3.3 V
Questa soluzione risulta semplice e permette di interfacciarci perfettamente con un sistema a basse tensioni.

Questa versione comporta lo svantaggio in termini di dimensioni del circuito e di numero di componenti infatti oltre ai due integrati aggiuntivi si rendono necessarie resistenze di pull-up e condensatori aggiuntivi per il pilotaggio dei traslatori.

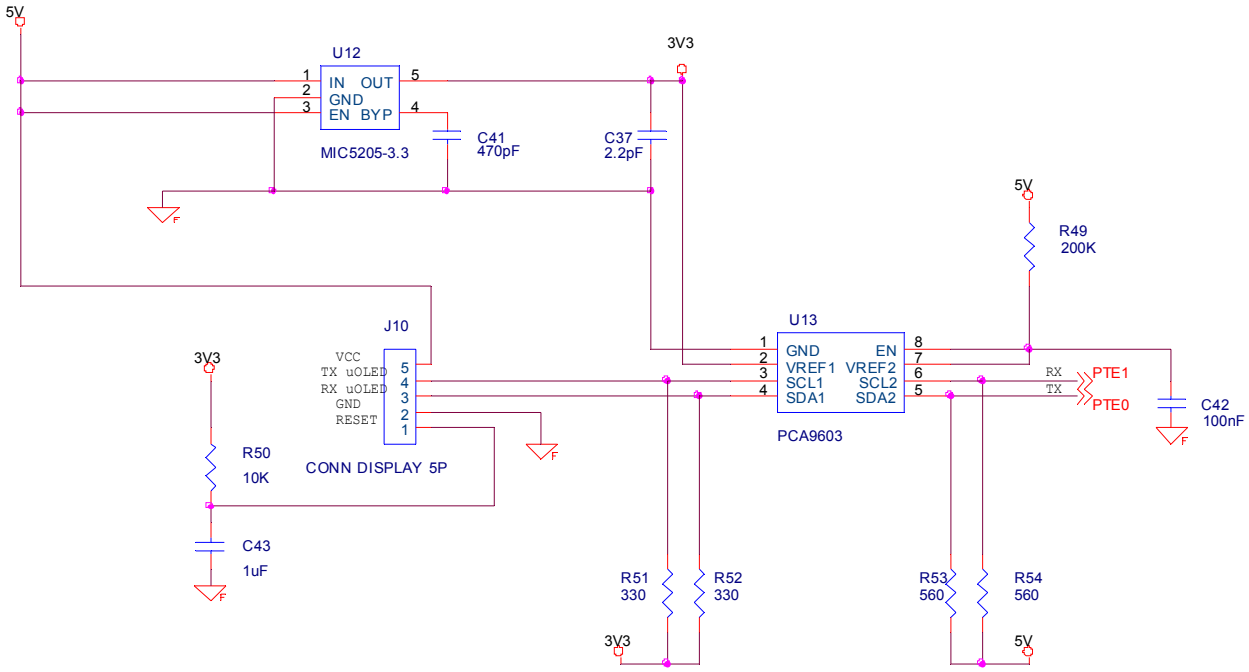


Figura 5.4.2

Sfruttando quindi una caratteristica di ingresso (in ricezione) del modulo display che accetta anche tensioni dell'ordine di 5V si è scelto di usare un integrato per rendere compatibile il solo segnale di trasmissione del display (in ricezione quindi alla nostra scheda).
Come vedremo in seguito (paragrafo 5.4) sfruttando semplicemente la caratteristica ingresso/uscita di porte NOT della serie HTC si risolveranno i problemi di compatibilità tra le nostre schede e il nuovo display OLED.

5.5 Il linguaggio di programmazione

Il 4DGL è un linguaggio di programmazione di alto livello orientato alla grafica che permette di sviluppare applicazioni anche complesse e di avviarle direttamente nel processore PICASO nei display 4D Systems.

Figura 5.5.1

Una estesa raccolta di funzioni (circa 230) per grafica, testi ed elaborazione di file, nonché la semplicità di un linguaggio “C like” simile anche al Basic o al Pascal, permettono una rapida programmazione e rendono inutile l'utilizzo di un controller host e di un'interfaccia seriale indipendenti.
Il codice 4DGL compilato viene eseguito direttamente sul processore PICASO ad una velocità superiore ai 50MIPS, più che soddisfacente per le nostre applicazioni.
La casa produttrice mette a disposizione dei tools per lo sviluppo di software quali il Workshop v3.0 il GraphicsComposer e il PmmCLoader.
L'intera scrittura del codice avviene tramite l'utilizzo dell'IDE Workshop reperibile al sito della casa madre www.4dsystems.com.au/developers/4dgl-download.php .
Workshop è un ambiente di sviluppo molto semplice da utilizzare ma anche molto scarso in quanto a tools per il debug e per il controllo del codice.
La libreria raccoglie un elevato numero di funzioni riguardanti grafica, testi, elaborazioni di file, gestione del tocco nonché il controllo seriale.
Un documento molto utile ed esplicativo riguardante le funzioni interne del processore PICASO si può trovare all'indirizzo www.4dsystems.com.au/developers/4dgl-download.php .
Il programma GraphicsComposer è utile se si vogliono inserire delle immagini nella scheda uSD. Genera dei pacchetti che vengono direttamente copiati nella scheda di memoria e contengono un insieme di foto, filmati e suoni che si intendono utilizzare nel display.
Per richiamare questi file il programma del display deve sapere l'indirizzo del file da leggere, informazione che ci da il GraphicsComposer .
Lo strumento PmmCLoader viene utilizzato esclusivamente per l'aggiornamento del firmware del modulo display, il che deve avvenire solo alla prima programmazione del modulo e per eventuali aggiornamenti del software IDE.
I primi codici generati riguardano la scrittura di stringhe attraverso la gestione del colore, del font e della posizione ma anche l'utilizzo del touch e della creazione di icone e pulsanti grafici.
Abbiamo un esempio del primo software creato nell'appendice C di questo documento.
In un secondo momento si è scelto, vista la disponibilità di tempo, di sviluppare un software compatibile e utilizzabile ad oggi prima dell'uscita in commercio degli strumenti con la nuova scheda elettronica completa (si è pensato infatti di utilizzare in modo temporaneo, nell'attesa che la nuova scheda venga montata e testata, il solo display in sostituzione del vecchio modulo a 2 righe e 16 caratteri).

Il software creato si basa su un protocollo di trasmissione/ricezione definito nel codice stesso che permette di comunicare con il modulo display tramite l'invio di un buffer di dimensione variabile.
Le potenzialità di questo sistema sono molteplici. Prima di tutto l'uniformità delle istruzioni e la versatilità nella comunicazione.
Con la stessa struttura infatti possiamo inviare pacchetti di dati che visualizzano stringhe modificandone il colore, la posizione e il font o visualizzano grafici a spezzate per qualsiasi grandezza.
Un esempio di comunicazione con il display è il seguente:

STX + "021" + "/+02P" + " PROVA SCRITTURA" + ETX

Il messaggio chiede al display la scrittura di una nuova stringa in riga 02 in colore rosa contenente il messaggio " PROVA SCRITTURA".
Ulteriori informazioni riguardanti il protocollo e il software definitivo sono contenute nel paragrafo 5.5.
Si è pensato di rendere un po più personalizzato lo strumento in cui viene montato il display inserendo un video o un'immagine di Jofa come sfondo di avvio dello strumento stesso.
Per fare in modo di visualizzarla è necessario caricarla nella memoria uSD tramite il Graphic Composer e richiamare l'indirizzo di memoria della uSD con una funzione di libreria da software.

5.6 Connessione del nuovo display con il vecchio circuito

Un passo intermedio tra lo studio della vecchia scheda e la progettazione della nuova ci ha portato ad interfacciare la scheda da anni in uso presso l'azienda con il display uOLED-32028-P1T. Questo è stato possibile grazie a due semplici realizzazioni circuitali:
In prima battuta la connessione tra il microcontrollore MC68HC705B16 e il modulo seriale del display la quale è stata effettuata connettendo adeguatamente alcuni punti della scheda per bypassare sostanzialmente un integrato (un ICL232) che adatta la trasmissione/ricezione del micro allo standard RS232 della seriale.
Si è reso quindi disponibile il connettore, che portava alla linea seriale esterna, collegato direttamente all'uscita/ingresso dell'ICL232 ed è stato utilizzato per la comunicazione con il display.
In seconda battuta sono state modificate le configurazioni nel programma del MC68HC705B16 per utilizzare il connettore (14 pin destinato al display 16x2) creando una seriale (via software) in una porta di I/O generico.
In entrambi i casi si è utilizzata una doppia porta inverter della serie HTC per riportare il livello alto della trasmissione al display ai 5V desiderati come accennato nel paragrafo 5.2.

5.6.1 Realizzazione del circuito

Utilizzando una piccola parte di una basetta mille fori, uno zoccolo DIL da 14 pin un 74HTC04 una resistenza e un condensatore è stato realizzato il piccolo circuito riportato in figura 5.4.1.1.

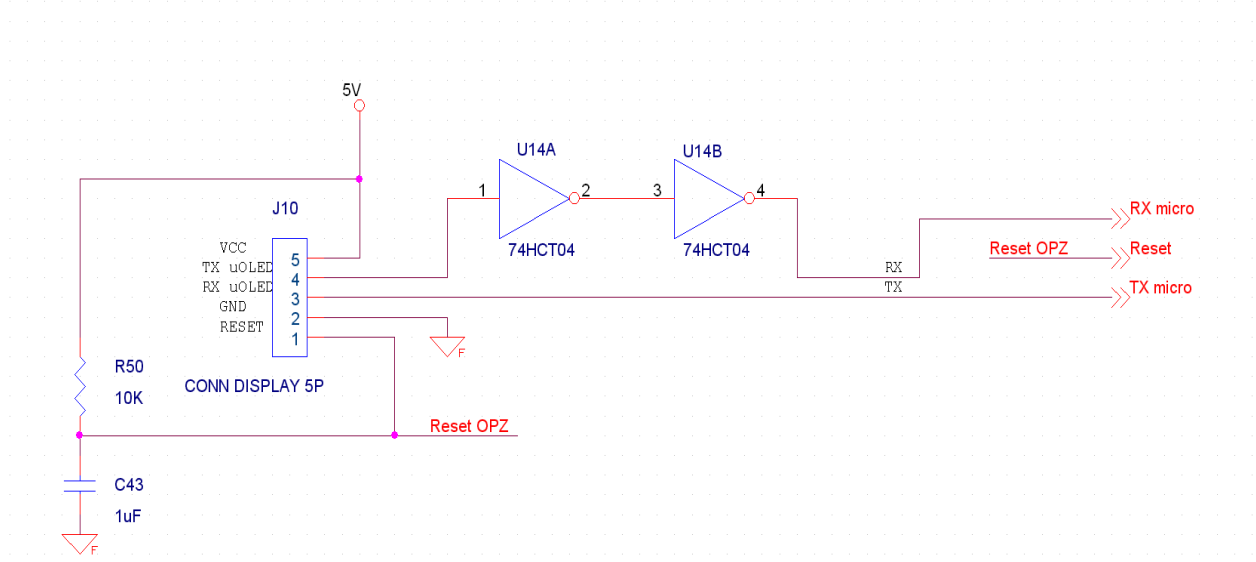


Figura 5.6.1.1

Per accendere effettivamente il display oltre ad alimentarlo è conveniente inserire un circuito di reset, in questo caso realizzato attraverso una resistenza ed un condensatore (nel circuito completo si vedrà di collegare il reset ad un pin del micro).
Il display infatti si resetta se il pin 5 viene mantenuto al livello basso per un tempo maggiore di 50us.
Fanno al caso nostro un condensatore da 1uF e una resistenza da 10KΩ scelti in modo tale che il tempo necessario per attivare il display sia di circa 50ms .
Le porte 74HCT04 hanno un tempo di transizione di circa 20ns non esistono quindi problemi per la velocità di comunicazione. Questi integrati ci permettono attraverso la compatibilità degli ingressi sia con lo standard TTL che con lo standard CMOS di avere in uscita un livello di tensione di 5V a fronte di un livello di 3.3V in uscita dal display.

5.6.2 Realizzazione del software

Per il micro risulta complicato intervenire sulla velocità di trasmissione/ricezione i pin attraverso i quali realizziamo la seriale sono pin generici, bisogna quindi essere accorti nell'usare le istruzioni assembly per temporizzare l'uscita di dati.
Abbiamo modificato il codice assembly per il microcontrollore e creato delle routine in 4DGL per scrivere le stringhe ricevute via seriale sul display.
Le routines realizzare per il micro leggono gli indirizzi di variabili stringhe e, carattere per carattere, inviano tramite seriale al display.

La routine LDST1 tramite le variabili PROVV e SCRA2 controlla se la stringa è stata letta per intero e in caso negativo invia il carattere successivo.
La routine LDST2 legge e invia da una stringa diversa dalla prima.
La routine TX8BB invia bit a bit il carattere caricato nella variabile DATX preceduto da 1 bit di start e seguito da un bit di stop come da protocollo standard RS232.
In particolare se analizziamo la figura 5.4.2.1 e osserviamo la routine TX8BB sulla destra ci sono i tempi di ciascuna istruzione assembly (da manuale nonché instruction set RASM05 relativo al micro MC68HC705B16) e la loro somma (ovvero la somma dei tempi di esecuzione delle istruzioni eseguite in un intero ciclo per tutti gli 8 bit più il bit di start e di stop) corrisponde a circa 104 us che è il periodo di tempo per l'invio di un bit se trasmesso ad una velocità di 9600 Baud.
Il display ha tra le sue funzioni di libreria una routine per il settaggio della velocità di trasmissione, risulta quindi semplice con l'istruzione setbaud(BAUD_9600); configurare il modulo seriale del display.
Abbiamo costituito una prima bozza di protocollo di trasmissione, risulta semplice inviare dei caratteri al display il quale a seconda del carattere ricevuto esegue una determinata operazione, tra le quali la scrittura in una riga, la modifica del colore e la cancellazione di una riga nonché la cancellazione dell'intero schermo.
Possiamo vedere il codice in appendice A.
Come possiamo notare il programma eseguito dal display rimane in attesa di comandi seriali e in base al carattere ricevuto salta all'esecuzione di determinate funzioni, attendendo l'invio di altri caratteri.

5.6.3 Test

La fase di test ha dato da subito gli esiti sperati.
La scrittura sul display avviene correttamente dopo l'invio da parte del microcontrollore delle stringhe JOFA1 e JOFA3 dichiarate come costanti nel codice assembly.
Un grosso dubbio riguarda la corrente assorbita dal display e il limite imposto da un circuito costruito modificando la scheda nata con scopi diversi.
Il vecchio display infatti assorbe una corrente di gran lunga inferiore, perché la corrente destinata alla sua illuminazione viene normalmente prelevata in un altro punto del circuito.
Con il circuito modificato la corrente deve ragionevolmente arrivare dallo stesso connettore per la connessione seriale.

5.7 Protocollo Tx/Rx definitivo e implementazione software

Dopo aver preso la decisione di montare il nuovo display sui nuovi strumenti creati da JOFA si è resa necessaria una certa uniformità nel controllo del display in quanto gli attrezzi sono tra loro diversi e hanno differenti necessità di visualizzazione.
Per questo è stato pensato e realizzato un protocollo di trasmissione in cui i due soggetti, microcontrollore e modulo uOLED-32028-P1T si sincronizzano con un comando di STX e concludono il colloquio con un comando di ETX.
Tra i quali vi è uno schema ben preciso descritto di seguito.

[STX] → [3BYTE (numero byte pacchetto)] → [1BYTE (comando principale)] → [1BYTE (comando secondario)] → [4BYTE (opzionali) (comandi specifici)] → [nBYTE (informazioni)] → [ETX]

In questo modo riusciamo ad inviare comandi per scrittura, cancellazione o disegno tramite un buffer con dimensione massima di 999 byte.
Il sistema uOLED controlla la corretta ricezione del buffer, invia il carattere di conferma ACK o il carattere di errore NAK in base al numero di byte ricevuti e ai tempi di ricezione.
Il secondo blocco contenente i 3 Byte con la grandezza del pacchetto, deve tener conto della grandezza dell'informazione da inviare nonché della somma dei byte per le istruzioni.
Risulterà quindi un numero a 3 cifre da 000 a 999.
Il terzo blocco contiene il comando principale è di un solo byte, generalmente qui sono contenuti caratteri che vanno utilizzati una sola volta nella trasmissione di un pacchetto e sono usati per la selezione della funzione principale, che può essere (fin qui) di scrittura, di visualizzazione foto, video o di rappresentazione di una curva spezzata per l'andamento di determinate grandezze.
Il quarto blocco contenente il comando secondario serve per selezionare la funzione secondaria, vale a dire, la scrittura di una nuova riga, la cancellazione di una riga, un nuovo punto in un grafico etc.
Il quinto blocco contenente i comandi specifici serve a particolarizzare la scelta fin qui fatta. Per esempio nel caso di scrittura (comando principale) con l'inserimento di una nuova riga (comando secondario) posso scegliere dove stampare la stringa e di che colore sarà la stessa (comandi specifici).
Il sesto blocco contenente l'informazione vera e propria, può essere composto dai caratteri della stringa da stampare, o dai punti del grafico su cui scrivere o ancora dall'indirizzo in memoria della foto da visualizzare.
Per la generazione del grafico sono state utilizzate delle funzioni grafiche per la generazione di righe e di forme; le funzioni sono gfx_Vline() e gfx_Hline() per la traccia delle rette e gfx_Triangle() per la traccia delle frecce degli assi.
Il nome di ascisse e ordinate passato tramite seriale dal micro viene stampato come testo.
Il software che realizza questo protocollo consiste in circa 300 righe di codice effettivo.
È inserito in appendice C.

5.8 Connessione del nuovo display con il nuovo circuito

Il nuovo circuito è già predisposto per la connessione con il display OLED; abbiamo infatti inserito un connettore AMP a 5 poli specifico con le linee di alimentazione, di trasmissione e di ricezione. Le tensioni sono ora compatibili grazie alla coppia di porte inverter inserite nella linea di ricezione del microcontrollore come visto nel paragrafo 5.2.

5.8.1 Realizzazione software

Il software realizzato in C per il micro MC9S08DZ128 è stato implementato tenendo conto del protocollo di cui abbiamo appena trattato in paragrafo 5.5.
Il sorgente C realizza vari test tra i quali la scrittura di varie stringhe di diversi colori e grandezze stampate in diverse posizioni dello schermo.
Per la creazione del buffer completo si utilizza la funzione sprintf() della libreria “stdio.h”.
Come parametri espliciti alla funzione inseriamo il buffer di destinazione, i tipi di dati da stampare e i vari pezzi della stringa da creare.
Di seguito un esempio di utilizzo della funzione.

```
sprintf ( buffer ," %c%s%c ", STX ,"029/+02A  questo e'il messaggio" , ETX ) ;
```

Il buffer risultato si presenta come nella seguente tabella:

Stx	0	2	9	/	+	0	2	A		Q	U	E	S	T	O		E	'	I	L		M	E	S	S	A	G	G	I	O	Etx

Il buffer viene inviato tramite una routine che accetta come parametri l'indirizzo del buffer e il numero di caratteri da inviare.
Sono state create anche delle routine per la visualizzazione di un grafico con diverse curve sovrapposte di diverso colore; queste richiedono l'invio di una sequenza di coordinate e il display si occupa della creazione di una curva spezzata. E' stata prevista la creazione di label per gli assi in modo tale che la visualizzazione possa avvenire per svariati tipi di variabili.
Il grafico visualizzato durante i test rappresenta l'andamento di 2 tensioni nel tempo.

5.8.2 Test

I test della comunicazione con il display sono andati a buon fine assieme ai test per il funzionamento generale della scheda freno. La visualizzazione avviene correttamente e con i vari formati e colori decisi da programma. Il grafico si presenta semplice ma esplicativo come si può vedere in figura.

Figura 5.8.2.1

Realizzazione dello stampato e collegamento dei componenti

6.1 BF engineering, masterizzazione scheda

La BF ENGINEERING è una ditta sita in Saonara, tecnicamente aggiornata e particolarmente specializzata nell'ingegnerizzazione e produzione di apparecchi elettronici.

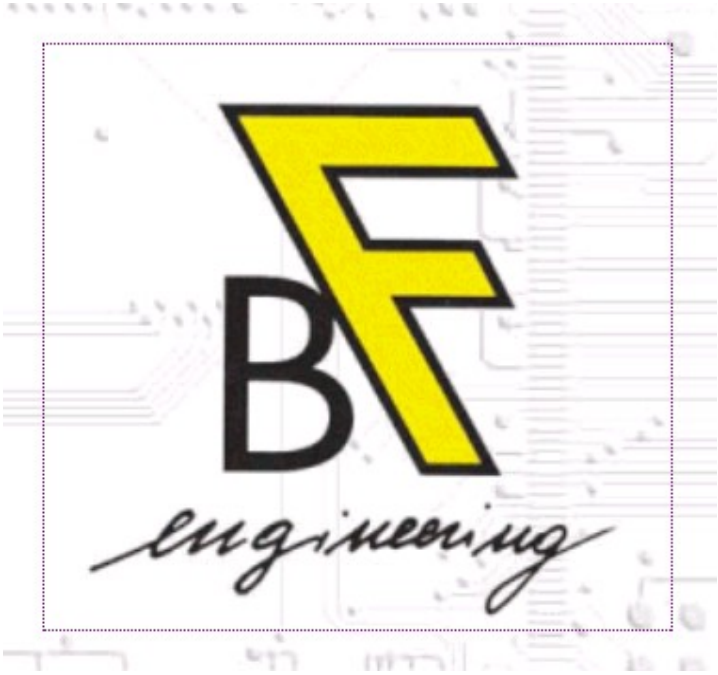


Figura 6.1.1

Essa si avvale di sistemi avanzati come workstation Apple e software Mentor Graphics che le permettono di interfacciarsi con i più diffusi CAD presenti sul mercato.
Il servizio offerto dalla BF ENGINNERING puo' venire cosi' riassunto:

- Masterizzazione di circuiti analogici,digitali,ecl.
- Documentazione Meccanica
- Liste componenti personalizzate
- Liste posizionamenti automatici personalizzate
- Photoplottatura laser
- Campionatura circuiti stampati

Per quanto riguarda la fase produttiva la BF ENGINNERING è specializzata in:

- Fornitura di circuiti stampati
- Fornitura di lamine per pasta saldante
- Montaggi di campionatura e serie sia in tecnologia radizionale che in SMT

L'azienda per la masterizzazione ha accettato il nostro ordine per una campionatura di circa 35 schede il 03/12/2010, confermandoci come data di consegna il 14-15 dello stesso mese.
Ci siamo però resi conto che a causa dell'aumento dei componenti e delle periferiche di

comunicazione esterne la densità troppo elevata delle piste avrebbe potuto creare disturbi.
La scelta, dopo varie valutazioni, è stata quella di passare al multistrato e quindi far progettare una scheda su 4 strati con la maggior parte dei componenti in SMD, questo ci garantisce una densità di piste minore e un'immunità ai disturbi maggiore.
Grazie alla disponibilità del progettista di BF engineering ho potuto assistere a parte del disegno della scheda e alle modifiche da noi richieste.
Il software di sviluppo è il Board Station della Mentor Graphics, un sistema completo di disegno di schede PCB che aiuta a posizionare i componenti, a collegare e ordinare il gran numero di connessioni tra di loro.
Cosa impensabile da farsi manualmente con un diverso programma di disegno tecnico a causa del numero elevatissimo di connessioni.

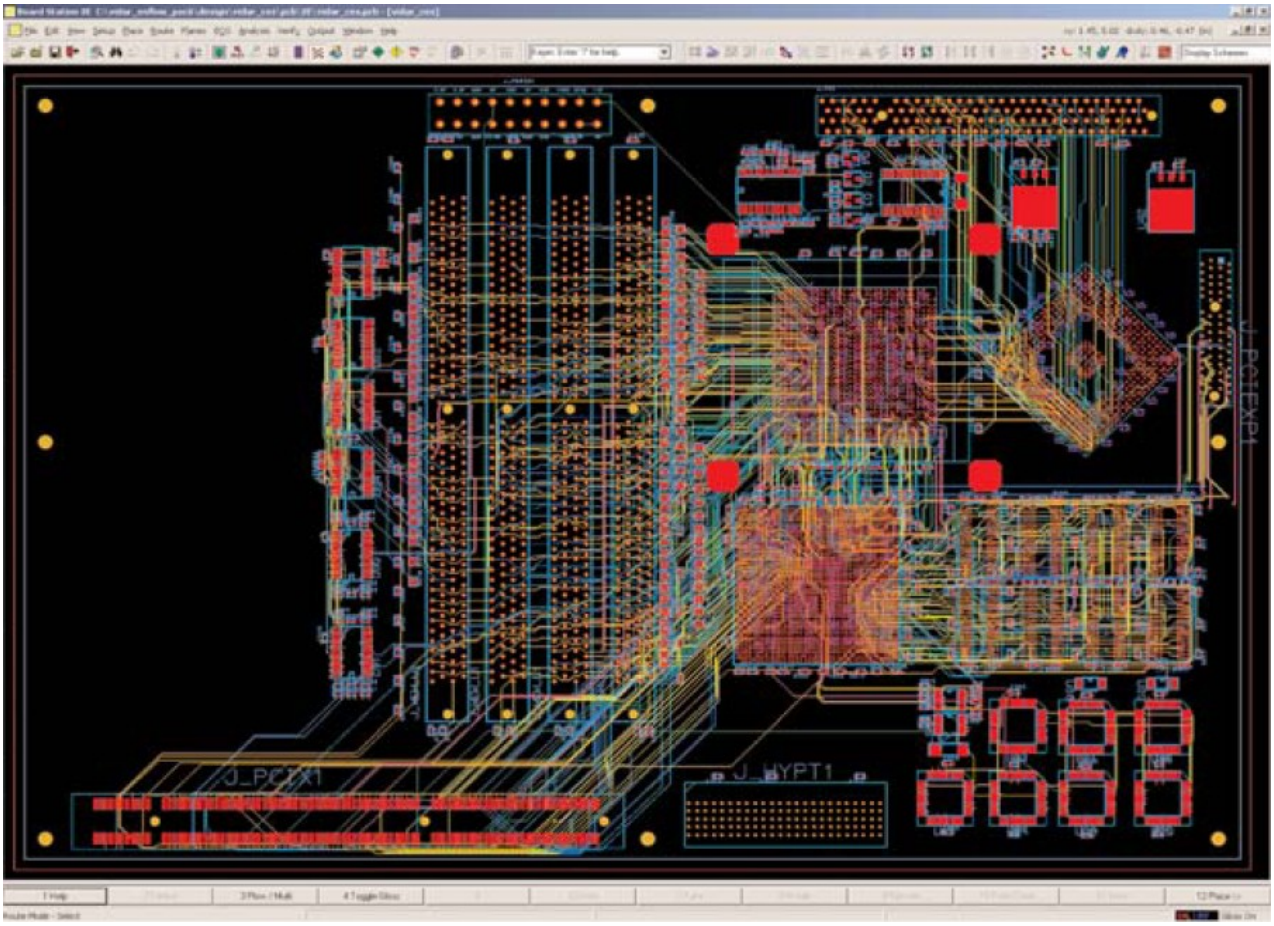


Figura 6.1.2

Il lavoro effettuato dal progettista si riduce a riordinare le piste e i componenti in modo da ottenere un insieme di collegamenti che si possono incrociare tra loro solo su 2 strati diversi, con una immunità ai disturbi il più grande possibile.
Si deve tener conto che le connessioni hanno una larghezza minima che dipende dalle potenze in gioco nel circuito, notiamo infatti piste più grosse nei collegamenti con tensione di 12 volt, mentre abbiamo piste più sottili nei punti raggiunti dalla tensione di 5 volt e nei generici collegamenti tra componenti.
A causa delle modifiche al progetto e delle festività la data di consegna è stata posticipata di qualche giorno.
Il 22/12/2010 abbiamo consegnato schede e componenti per la saldatura di 6 campioni di schede alla ditta EL.PI. elettronica di Sant'Angelo di Piove di Sacco

6.2 Saldatura componenti

La ditta EL.PI elettronica dal 1986 si occupa di montaggio di prodotti elettronici, assemblaggio e gestione totale o parziale dei componenti da distinta del cliente, con fornitura della scheda completa, collaudi e test funzionali su specifica del cliente.
A partire dal 1960 è stata sviluppata una tecnica chiamata Surface Mounting Technology (SMT), che prevede il montaggio di componenti appositamente progettati direttamente a contatto della superficie del circuito stampato. I componenti (SMD, Surface Mounting Device) sono progettati per avere il minimo ingombro e peso possibile, ed i contatti sono costituiti dalla metallizzazione delle estremità dell'oggetto, oppure da corte terminazioni metalliche sporgenti.

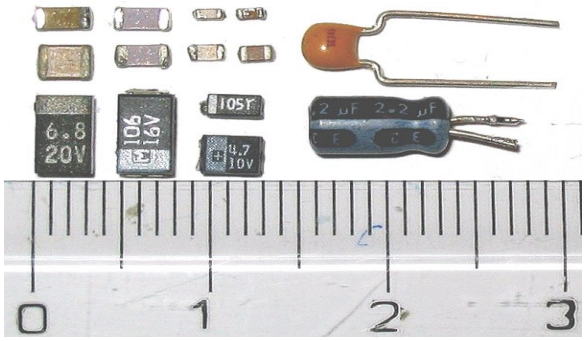


Figura 6.2.1

Un componente SMD può avere un ingombro pari ad un decimo di un componente tradizionale e costare, compreso il montaggio, fino ad un quarto (Figura 6.2.1).
Questa fondamentale tecnologia ha consentito una vera e propria rivoluzione industriale nel mondo dei circuiti elettronici, infatti rispetto alla tecnologia tradizionale PTH, ha i seguenti vantaggi:

- consente di collocare (tramite processo serigrafico) una quantità molto precisa di pasta saldante sulle piazzole di rame (dette "pads") che in seguito alloggeranno terminali di componenti;
- dà la possibilità di utilizzare potenti e velocissime automazioni per collocare i componenti sul circuito stampato, riducendo l'incidenza della manodopera e quindi consentendo una maggiore economia ed una maggiore qualità;
- consente di saldare i componenti elettronici alle pads tramite un processo termico molto più controllabile e meno stressante di quello usato per i componenti tradizionali;
- consente di usare componenti molto più miniaturizzati, riducendo drasticamente le dimensioni degli apparecchi elettronici;
- riduce il numero di fori da praticare sul circuito stampato in quanto non sono più necessari i fori per alloggiare le terminazioni dei componenti

Prima del posizionamento del componente occorre depositare una certa quantità di pasta saldante sulle piazzole destinate alla saldatura.
I componenti SMD vengono commercializzati in confezioni adatte al prelievo automatizzato; in particolare si usano bobine a nastro continuo per i componenti più grandi e/o dotati di polarizzazione, e da piccole scatole (bulk) nel caso dei componenti più minuscoli. Una macchina automatica preleva i componenti dalle confezioni mediante apposite testine ad aria aspirante, e li depone con precisione nella loro collocazione finale sul circuito stampato. I componenti sono in genere trattenuti nella loro posizione fino alla fase di saldatura dalla viscosità dei mattoncini di pasta saldante sottostanti ai terminali del componente; in alcuni casi particolari si può anche avere un punto di colla pre-depositata sul circuito stampato (normalmente mediante processo serigrafico).

Il circuito così completo di componenti viene preriscaldato e collocato in un apposito forno elettrico suddiviso in zone di presiscaldo (pre-heating), refusione (reflow) e raffreddamento (cooling) dove la temperatura e la ventilazione possono essere regolate con grande precisione. Per mezzo di un nastro trasportatore il circuito avanza lentamente nel forno attraversando aree con temperature via via crescenti (Figura 6.2.3).

Figura 6.2.3

Superato il punto di fusione della lega saldante, la pasta saldante rifonde e aderisce alle superficie metalliche scoperte, realizzando così il giunto saldante vero e proprio tra il circuito stampato i e componenti elettronici SMD. In seguito il circuito attraversa zone a temperature calanti per consentire un graduale raffreddamento dei materiali.
I componenti specifici per questa tecnica sono stati dotati di piccoli terminali metalliche per saldarli direttamente al circuito stampato.

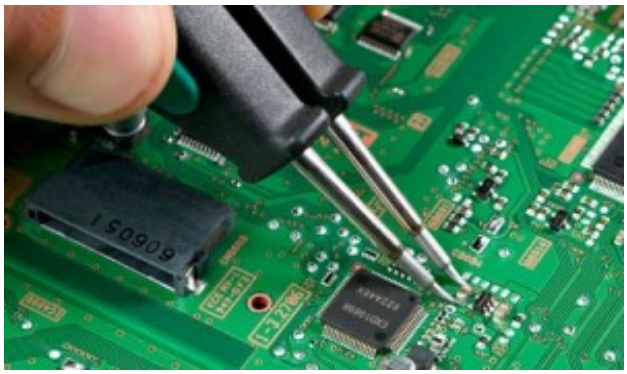


Figura 6.2.2

Sono stati consegnati i componenti e le schede vuote per realizzare 6 schede complete. L'azienda ci ha restituito le schede montate il 10/01/2011 e sono da subito iniziati i test.

Test Finale

Per gli ultimi test ci siamo concentrati sulla nuova scheda montata e collegata al display OLED con il software visto nel capitolo 5 già precaricato.
Abbiamo ricavato un piccolo spazio vicino ad una postazione pc per il collaudo dell'attrezzatura, dove sono disponibili e a portata di mano il programmatore per il micro, un alimentatore a 12 volt e vari strumenti di misura, tra i quali un tester e un oscilloscopio digitale.
Al momento della connessione con l'alimentazione dobbiamo utilizzare in modo adeguato il Jumper per la selezione della linea di reset.
Per la fase di programmazione il jumper va collegato in modo tale da connettere il “reset” al pin apposito del programmatore USB.
Per la fase di esecuzione il jumper deve connettere il pin di riavvio del micro con l'uscita del circuito watchdog.
Se la scelta fatta rispecchia la seconda configurazione, il micro riceverà ad intervalli di tempo regolari un segnale di riavvio dal circuito di watchdog esterno.
In questa fase infatti manca ancora un programma che gestisca il segnale di reset della scheda.
E' necessario, come già detto, che il programma modifichi ad intervalli minimi di tempo il segnale di WatchDog in modo da non far commutare l'uscita del circuito astabile.
Per questo è stata sviluppata una routine che deve essere chiamata sia all'interno del main che in altre funzioni, al fine di “rinfrescare” il segnale di controllo.
Il programma che si è sviluppato in seguito, come spiegato nel capitolo 4, utilizza principalmente le linee seriali, le linee I²C, gli ingressi analogici e alcune uscite di comando.
Una volta caricato il software e ripristinato il collegamento del jumper, la scheda deve lavorare in modo autonomo e seguire i passi decisi in fase di programmazione.
Il primo controllo effettuato al momento della connessione ha riguardato le tensioni, è infatti essenziale che le tensioni soprattutto per le connessioni con periferiche esterne siano il più possibile stabili e precise.

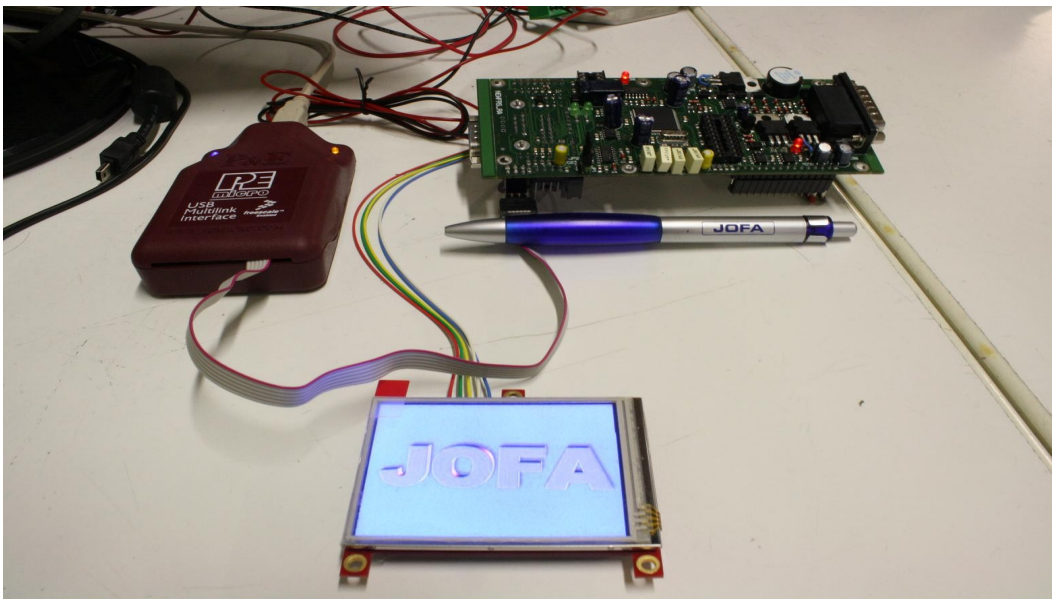


Figura 7.1

Dopo alcuni tentativi, varie modifiche e upgrade al software possiamo affermare che la scheda funziona perfettamente e si interfaccia egregiamente con tutta la vecchia strumentazione esterna, quali memorie, display e ingressi analogici.

Esegue la conversione analogico digitale in modo corretto e non comporta esagerate temperature in fase di lavoro.
Il codice creato serve anche per testare tutto il protocollo di comunicazione sviluppato e realizzato nel programma del display OLED; per questo il software per il micro contiene un gran numero di righe dedicate alla comunicazione seriale.
Nel sorgente scritto è stata inserita la creazione di un grafico tensione/tempo con 2 curve di diverso colore.
Nella figura 7.1 sono state fotografate la scheda connessa al programmatore e al display mentre visualizza la prima schermata con il logo Jofa.

Problematiche emerse nel corso dello sviluppo

Nel progetto della scheda e in generale per percorso di produzione ci sono stati degli “incidenti di percorso” o per cosi dire dei piccoli problemi che hanno rallentato e posticipato la fine del progetto.

- Errore nel datasheet micro nuovo
  Quando siamo passati allo studio del micro nuovo ci siamo imbattuti in un'incoerenza riguardante la piedinatura del microcontrollore Freescale.
  Una tabella con l'elenco dettagliato dei pin del componente infatti riporta una informazione discordante con il disegno che segue, mettendoci qualche serio dubbio sull'esatta configurazione di piedini.
  Contattato il servizio clienti freescale via mail il problema si è risolto entro pochi giorni con un chiarimento, le scuse e dei ringraziamenti per la segnalazione effettuata.

- Tempi di consegna componenti
  I tempi di consegna dei componenti han creato abbastanza disagio in quanto alcuni integrati con consegna prevista per la prima settimana di dicembre, ad oggi devono ancora essere inviati.
  Il componente in questione è il BTS3408G, è stato ordinato ancora nei primi mesi del 2010 un blocco di 2500 componenti con un'attesa minima di 6 mesi.
  Purtroppo i mercati asiatici e dei paesi ad oggi in piena fase di crescita assorbono la maggior parte della produzione di componenti elettronici mondiale.
  Questo fatto crea un problema di reperibilità per paesi come in Italia che non hanno una grossa produzione richiesta di componenti elettronici.
  Siamo riusciti a farci inviare una campionatura di circa 15 integrati da testare perché altrimenti non sarebbe bastato un anno per riuscire a costruire il primo prototipo.

- Errore in fase di masterizzazione
  Dopo l'ordine inviato a BF engineering per lo sbroglio de progetto ci siamo resi conto di una possibile semplificazione al circuito e di un piccolo errore di progettazione.
  Abbiamo fermato il progetto e rivisto assieme all'azienda esterna gli errori e le possibili semplificazioni e in un paio di ore abbiamo completato il disegno del PCB.

- Lamina per la pasta saldante danneggiata
  Per la saldatura di componenti a tecnologia SMD è necessaria una lamina speciale creata dalla stessa azienda che stampa il circuito PCB con delle apposite finestre per il deposito della pasta saldante.
  Al momento della consegna ci siamo accorti di alcuni difetti di lavorazione notevoli.
  Come si vede dalla foto, in due punti la lamina ha dei fori che non son definiti e rischiano di connettere due pin del microcontrollore altrimenti isolati tra di loro.
  La lamina difettata è stata sostituita in un paio di giorni dall'azienda produttrice.

- Elevata densità vs potenza massima componenti SMD
  Nella tecnologia SMD come nella tecnologia tradizionale THT(through holder technology) ci sono componenti per determinati livelli di potenza dissipabile (e quindi dimensioni), per cui avendo delle limitazioni per le dimensioni della scheda, siamo costretti a scegliere di inserire componenti di 0,5-0,8 mm di altezza.
  Questo limite in materia di dimensioni comporta un limite nella potenza massima dissipabile dai componenti.

Per questo è stato necessario rivedere tutti i componenti resistivi e controllare la potenza in essi dissipata.
Sono state sostituite alcune resistenze che non avevano un margine elevato di potenza rispetto a quella mediamente dissipata con il circuito acceso.

- Errata libreria ORCAD con conseguente errore stampa scheda
  In azienda sono presenti delle librerie personali per la creazione di componenti personalizzati o componenti non presenti nelle librerie standard del programma.
  Con il praticissimo copia-incolla però non abbiamo fatto altro che alimentare questo problema e creare errori che non si sono mai manifestati grazie ai programmi vecchi delle aziende collaboratrici quali BF engineering ma ora, hanno creato un problema di collegamenti nelle nuove schede.
  Nelle vecchie librerie infatti sono invertiti 2 piedini dell'integrato stabilizzatore 7805.
  Sono state rinnovate le librerie dei componenti in modo da eliminare possibili problemi futuri.

Conclusioni

L'esperienza in Jofa è stata sicuramente positiva, ho appreso un metodo di lavoro preciso e attento grazie alla grande competenza e precisione degli ingegneri e dei tecnici che lavorano al suo interno.
In più occasioni ho dovuto mettere alla prova le mie conoscenze e soprattutto la mia capacità di apprendimento per gestire problemi di collaudo o di programmazione software o di dimensionamento componenti elettronici.
Ritengo di aver fatto un buon lavoro visto che son stati svolti obbiettivi stabiliti come opzionali all'inizio dell'esperienza di tirocinio.
Immagino che la scheda creata verrà utilizzata sugli strumenti di nuova generazione con una semplificazione sostanziale della programmazione dei microcontrollori e una base solida per l'utilizzo di display simili a quello studiato durante lo stage.
Inoltre basterebbe poco per implementare qualche altra routine per poter inserire dei comandi utilizzando il touchscreen del display stesso, magari in strumenti che non sono sottoposti a sollecitazioni meccaniche elevate.

Bibliografia e Sitografia

I. Instruction Set – Architecture & Adressing Modes – RASM05 Macro Assembler Syntax and Directives
II. Sito web della Freescale e Customers Support www.freescale.com
III. Sito web P&E Microcomputers Systems, Inc. www.pemicro.com
IV. Sito web 4D Systems www.4D-labs.com
V. Sito web esperienza personale di utilizzo del display www.lucasproject.it
VI. Sito dell'azienda www.jofa.it
VII. Sito acquisto componenti
VIII. Quaderno relazione di fine corso e appunti di Programmazione di Sistemi Digitali
IX. Datasheet 74HC_HCT244.pdf
X. Datasheet 74HC_HCT374_CNV_2.pdf
XI. Datasheet 78lxx (stabilizzatore 5V).pdf
XII. Datasheet 5279(uln2003).pdf
XIII. Datasheet 21703J(I²C EEprom).pdf
XIV. Datasheet 187365_DS(regolatore 3.3v).pdf
XV. Datasheet 0900766(Trigger di schmitt).pdf
XVI. Datasheet 0900766b80d5a363(Quarzo).pdf
XVII. Datasheet 0900766b80d25e46(MM74HCT04MX).pdf
XVIII. Datasheet bts3408g(switch protezione).pdf
XIX. Datasheet fn3020(icl232).pdf
XX. Datasheet max3232.pdf
XXI. Datasheet MC9S08DZ128 (nuovo uC).pdf
XXII. Datasheet MC68HC05B4 (vecchio uC).pdf
XXIII. Datasheet MK48T08_394825(SRAM).pdf
XXIV. Datasheet PCA9306 TexasInstrument(Traslatore di Livello 3.3-5).pdf
XXV. Datasheet sn74avch1t45.pdf
XXVI. Datasheet tlp521(Photocoupler).pdf
XXVII. Datasheet uUSB-MB5(connettore Display uUSB).pdf
XXVIII. Datasheet uOLED-3202X-P1T_Users_Manual_Rev1_0.pdf
XXIX. Datasheet PmmC Loader Users Guide Rev1.1.pdf
XXX. Datasheet PICASO-GFX2-4DGL-Internal-Functions-rev2.pdf
XXXI. Datasheet 4D_AMOLED_Presentation.pdf
XXXII. Datasheet FS-MDLS16265SS-07(display vecchio).pdf

Appendice A
Codice Sorgente Display

```
#platform "uOLED-32028-P1T"

/*************************************************

* Filename: OLED_4.4dg
* Created: 2010/12/13
* Author: Andrea Bisenzi
*Description: Software Display OLED

*************************************************/

/*************************************************
*************************************************

DOWNLOAD IMMAGINI IN uSD
Caricare le immagini ed eventuali video tramite il Graphic
Composer. Selezionare Device-->Load Options e scegliere uSD Raw, selezionare
il drive esatto su cui caricare l'insieme delle immagini.
Nuovo progetto, in grandezza scrivere 240x320 e ricordarsi di inserire le immagini ruotate
in senso orario di 90°.
Richiamare sempre la funzione uSD_Init.
Usare il file creato dal GC con estensione *.gc per passare
gli indirizzi corretti dell'immagine/video alle funzioni di visualizzazione.

*************************************************
*************************************************/


/*************************************************
*************************************************
****      PROTOCOLLO TRASMISSIONE          ****
*************************************************
*************************************************

[STX] [3BYTE (numero byte pacchetto)] [1BYTE (comando principale)] [1BYTE (comando secondario)]
                                (livello 0)                 (livello 1)
[4BYTE (opzionali) (comandi specifici)] [nBYTE (informazioni)] [ETX]
   L'invio di dati tramite seriale si attiva con
   la ricezione da parte del display del carattere
   STX seguito dalla grandezza in byte del pacchetto
   che desidero inviare.
   Si accede quindi al livello 0.

            ***************
            ***livello 0***
            ***************

   I comandi di livello 0 che riceve il
   display sono:
   '/'         comando scrittura stringhe
   '*'         comando stampa immagini                **non implementato**
   '@'          comando di visualizzazione video        **non implementato**
   '~'         comando visualizzazione grafico

            ***************
```

```
        ***livello 1***
        **************

    SCRITTURA STRINGHE:
    1)con la ricezione di un carattere '+'
      inizializza una nuova riga accetta in ingresso        P.E. STX + "021" + "/+02P" + " PROVA SCRITTURA" + ETX
      in sequenza il numero di riga(2Byte)                  scrive un buffer di 20 caratteri che aggiunge al display nella
      (caratteri da 0 a 30),il colore della riga stessa     riga numero 2 con il colore rosa la stringa " PROVA
                                                            SCRITTURA"
      (B=blu,R=rosso,r=rosa,L=lime,A=arancione,G=giallo
      V=verde,M=marrone,P=rosa,W=bianco)e di seguito
      i caratteri della stringa.
      Quando sono stati trasferiti i caratteri desiderati
      e la trasmissione è andata a buon fine il DISPLAY invia
      come comando di controllo il carattere ACK.
      Se la ricezione non è andata come desiderato il
      DISPLAY invia il carattere NAK.
    2)con la ricezione del caratere 'ESC' (0x1B)         P.E. STX + "002" + "/" + ESC + ETX  Ripulisce lo schermo
      Viene ripulito il display.
    3)con la ricezione del carattere   '-'               P.E. STX + "004" + "/-06" + ETX     cancella la riga numero 06
      entro nella routine di cancellazione riga.
      accetta come carattere il numero(2 Byte) da 1 a 30 corrispondente
      al numero di riga da cancellare.
    4)con la ricezione del carattere '%' inizializza il vettore     P.E. STX + "018" + "/%" + "0255034405771511" + ETX
      delle grandezze delle singole righe del display.              modifica il vettore che contiene le grandezze di riga
      Accetta in ingresso il numero di riga(2Byte),                 riga2(grandezza x5) riga3(grandezza x4) .....
      la larghezza dei caratteri della riga e l'altezza degli stessi.
      se desidero inizializzare più righe metto in sequenza (nel buffer che invio)
      altre scritture del tipo[numero riga(2Byte)][lunghezza caratteri][altezza caratteri]

    VISUALIZZAZIONE GRAFICO
    1)Con la ricezione del carattere '+' il display aspetta 10 caratteri     P.E. STX + "022" + "~+" + "TENSIONE
                                                                             TEMPO     " + ETX
      per la label da inserire nell'asse Y e altri 10 caratteri per la label   scrive un buffer di 22 caratteri inizializza un
      dell'asse X. A fine ricezione il display mi visualizza il grafico e      nuovo grafico con Tensione nelle ordinate e
       inserisce il nome delle variabili X e Y del grafico.                    tempo nelle ascisse

    2)Con la ricezine del carattere 'p' accetta le coordinate del
      nuovo punto per il disegno del grafico. Attende                  P.E. STX + "005" + "~p" + "050" + ETX
      3 byte  per il valore di X e                                     Aggiunge un nuovo punto al grafico e lo unisce
      3 byte per il valore di Y.                                       con una retta al valore dell'ultimo punto sul grafico.

    3)Con la ricezione del carattere 'c' inizializza una nuova curva.
       riporta il cursore nell'origine e attente il colore desiderato        P.E. STX + "005" + "~cG" + ETX
       per la curva.                                                           nuova curva di colore giallo.

**********************************************
**********************************************/

#inherit "4DGL_16bitColours.fnc"
//definizione costanti
#CONST
   ACK       0x06
   NAK       0x15
   ETX       0x03
   STX       0x02
   ESC       0x1B
#END
//definizione variabili
var riga,tast,colour,count,buffer[256],COM_BUF,y_value[10],x_value[10],Xpos,Ypos,width[30],height[30];
```

```
var etx_flag,nak_flag,timer_flag;

//Serve per settare il font. Moltiplica lunghezza e larghezza del font1 per gli indici presenti nel //vettore width e nel vettore height
func fontbig(var width,var height)
 //   var moltiplier;
 //   moltiplier:=5;
    txt_Set(TEXT_OPACITY,OPAQUE);
    txt_Set(TEXT_WIDTH, width);
    txt_Set(TEXT_HEIGHT, height);
endfunc

 //Riceve un carattere corrispondente ad un colore e salva nella varibile colour il colore desiderato.
 //viene chiamata dalla routine di scrittura riga e per il colore della curva del grafico.
func setColour()
    var temp;
    temp:=buffer[count++];
    colour:=WHITE;
  //  if(isupper(temp))
        if(temp=='G') colour:=YELLOW;
        if(temp=='V') colour:=GREEN;
        if(temp=='L') colour:=LIME;
        if(temp=='B') colour:=BLUE;
        if(temp=='A') colour:=ORANGE;
        if(temp=='M') colour:=BROWN;
        if(temp=='R') colour:=RED;
 //   if(temp=='P') colour:=PURPLE;
        if(temp=='P') colour:=PINK;
        if(temp=='W') colour:=WHITE;
 //   endif
    txt_Set(TEXT_COLOUR, colour);
endfunc

 //Riceve un carattere numerico
 //e restituisce il valore numerico in binario del carattere ricevuto.
func numChar(var temp)
    var num_char;
        if(temp=='0') num_char:= 0;
        if(temp=='1') num_char:= 1;
        if(temp=='2') num_char:= 2;
        if(temp=='3') num_char:= 3;
        if(temp=='4') num_char:= 4;
        if(temp=='5') num_char:= 5;
        if(temp=='6') num_char:= 6;
        if(temp=='7') num_char:= 7;
        if(temp=='8') num_char:= 8;
        if(temp=='9') num_char:= 9;
    return num_char;
endfunc

  //Riceve 2 caratteri in sequenza e salva il numero corrispondente nella
  //variabile riga
func setLine()
    riga:= numChar(buffer[count++])*10 + numChar(buffer[count++]);
endfunc


  //Modifica i vettori delle grandezze e permette di amplificare larghezza
  //e altezza dei caratteri
  //il buffer contiene il numero della linea in cui desidero modificare la
```

```
  //grandezza dei caratteri e modifica il vettore delle grandezze alla posizione corrispondente
func inizGrandezza()
    while(count<COM_BUF)
        setLine();
        width[riga]:=numChar(buffer[count++]);
        height[riga]:=numChar(buffer[count++]);
    wend
endfunc

  //Elimina la riga scelta sostituendola con una stringa vuota
  //la funzione controlla i vettori di larghezza e altezza
  //per inserire il carattere vuoto con grandezza corretta
func delLine()
    var stringa;
    setLine();
    txt_MoveCursor(riga,0);
    fontbig(width[riga],height[riga]);
    print("                                          ");
endfunc

  //Stampa a video la stringa contenuta nel buffer ricevuto
  //con riga e colore prescelti
func printLine()
    var stringa;
    txt_MoveCursor(riga,0);                 //muove il cursore nella riga impostata
    while(count<COM_BUF)                    //imposta la grandezza dei caratteri come preimpostato
        fontbig(width[riga],height[riga]);  //nel vettore grandezze
        putch(buffer[count++]);             //inserisce il carattere
    wend
endfunc

  //Realizza gli assi per un nuovo grafico cartesiano
  //e inserisce i nomi delle variabili ricevute nel buffer
  //rispettivamente nelle ordinate e poi nelle ascisse
func newGraph()
    var contatore;
    gfx_Cls();
    Xpos:=9;
    Ypos:=231;
    //Y LINE  visualizza l'asse y spesso 4 pixel
    gfx_Vline(7,240,10,WHITE);
    gfx_Vline(7,240,9,WHITE);
    gfx_Vline(7,240,8,WHITE);
    gfx_Vline(7,240,7,WHITE);
    gfx_Vline(7,240,11,WHITE);
    //X LINE  visualizza l'asse x spesso 4 pixel
    gfx_Hline(0,313,233,WHITE);
    gfx_Hline(0,313,230,WHITE);
    gfx_Hline(0,313,231,WHITE);
    gfx_Hline(0,313,232,WHITE);
    gfx_Hline(0,313,232,WHITE);



    //realizza i triangoli che fungono da frecce direzionali degli assi
    gfx_Triangle(9,0,2,7,17,7,WHITE);
    gfx_Triangle(313,223,313,238,320,231,WHITE);
    txt_Set(TEXT_COLOUR,WHITE);
    txt_MoveCursor(1,4);
```

```
    contatore:=count;
    //stampa vicino ai vertici degli assi la grandezza visualizzata
    while(contatore - count < 10) putch(buffer[contatore++]);
    contatore:=count + 10;
    txt_MoveCursor(27,42);
    while(contatore - count < 20) putch(buffer[contatore++]);
    colour:=RED;
endfunc

  //Inizializza una nuova spezzata con un nuovo colore
  //che ha il suo zero nell'origine degli assi
func newCurve()
    Xpos:=9;
    Ypos:=231;
    setColour();
endfunc

  //inserisce nel grafico un nuovo punto
  //riceve le coordinate
//inserisce un nuovo punto nel grafico e unisce il punto con il precedente tramite una retta
func addPoint()
    var XposOld,YposOld,XposNew,YposNew;
    XposOld:=Xpos;
    YposOld:=Ypos;
    XposNew:=numChar(buffer[count++])*100 + numChar(buffer[count++])*10 + numChar(buffer[count++]);
    YposNew:=numChar(buffer[count++])*100 + numChar(buffer[count++])*10 + numChar(buffer[count++]);
    gfx_Line(XposOld,YposOld,XposNew,YposNew,colour);
    Xpos:=XposNew;
    Ypos:=YposNew;
endfunc

//stessa funzione di addpoint() ma inserisce i punti ad intervalli di x predefiniti
//è sufficiente inviare il solo nuovo valore in y
func addPoint2()
    var XposOld,YposOld,XposNew,YposNew;
    XposOld:=Xpos;
    YposOld:=Ypos;
    //XposNew:=numChar(buffer[count++])*100 + numChar(buffer[count++])*10 + numChar(buffer[count++]);
    YposNew:=numChar(buffer[count++])*100 + numChar(buffer[count++])*10 + numChar(buffer[count++]);
    XposNew:=Xpos + 10;    //si sposta nell'asse x del valore preimpostato "10"
    gfx_Line(XposOld,YposOld,XposNew,YposNew,colour);
    Xpos:=XposNew;
    Ypos:=YposNew;
endfunc









//funzione di decodifica del buffer
//se risulta corretta la ricezione la funzione buffread() legge ogni singolo byte
//del buffer ed esegue il comando corrispondente
func buffRead()
    if(buffer[count]=='/')                  //entra se sono nella fase di scrittura e modifica testo
        count++;
```

```
        if(buffer[count]=='+')                  //aggiunta nuova riga
          count++;
          setLine();
          setColour();
          printLine();
        endif
        if(buffer[count]== ESC)                 //pulizia schermo
          gfx_Cls();
        endif
        if(buffer[count]=='-')                  //cancellazione riga
          count++;
          delLine();
        endif
        if(buffer[count]=='%')                   //modifica il vettore delle grandezze
          count++;
          inizGrandezza();
        endif
      endif
      if(buffer[count]=='~')                     //entra se sono in fase di disegno grafico
        count++;
        if(buffer[count]=='+')                  //crea gli assi per un nuovo grafico
          count++;                              //e inserisce il nome delle variabili
          newGraph();
        endif
        if(buffer[count]=='p')                  //inserisce un nuovo punto nel grafico
          count++;
          //addPoint2();
          addPoint();
        endif
        if(buffer[count]=='c')                  //inserisce una nuova curva che parte dall'origine
          count++;                              //nel grafico che sto visualizzando
          newCurve();
        endif
        if(buffer[count]==ESC)                  //ripulisco lo schermo
          gfx_Cls();
        endif
      endif
    endfunc
    //inizializza il display con un'immagine personalizzata Jofa e setta le impostazioni principali per
    //velocità di trasmissione e modalità schermo
    func inizializza()
      var i;
      setbaud(BAUD_9600);
      if(!(uSD_Init()))                 //controlla la presenza della memory card
        while(!(uSD_Init()))
          gfx_Set(SCREEN_MODE, LANDSCAPE);
          fontbig(3,3);
          print("Please Insert uSD Card!!");
          while(1);
        wend
      gfx_Set(SCREEN_MODE, PORTRAIT);
      endif
      i:=0;
      while(i < 30)
        height[i]:=2;
        width[i]:=2;
        i++;
      wend
    //    uSD_SetSector(0x0000, 0x0000);
    //    uSD_Video(15, 0);
```

```
    uSD_SetSector(0x0000, 0xD445);          //preleva l'immagine dall'indirizzo
    uSD_Image(0, 0);                        //di memoria della memory card e la visualizza
    gfx_Set(SCREEN_MODE, LANDSCAPE);
    txt_Set(FONT_SIZE, FONT1);
    txt_Set(TEXT_COLOUR, LIME);
endfunc
//riceve il buffer e controlla la coerenza dello stesso
//se la ricezione impiega troppo tempo il programma torna in stand-by uscendo dalla funzione.
func buffLoad()
  var contatore;
  txt_MoveCursor(0,0);
  sys_SetTimer(TIMER0,500);
  tast:=-1;
  contatore:=0;
  COM_BUF:=0;
  //memorizzo la grandezza totale del buffer
  while(contatore<3 && !timer_flag && !etx_flag)
    if(sys_GetTimer(TIMER0)<(0.001)) timer_flag:=1;
    tast:=serin();
    if(tast==ETX) etx_flag:=1;
    if(tast>0 && !etx_flag)
      if(contatore==0)COM_BUF := COM_BUF + numChar(tast)*100;
      if(contatore==1)COM_BUF := COM_BUF + numChar(tast)*10;
      if(contatore==2)COM_BUF := COM_BUF + numChar(tast)*1;
      //COM_BUF contiene la grandezza del buffer contenente l'informazione
      contatore++;
    endif
  wend
  contatore:=0;
  sys_SetTimer(TIMER0,1000);
  tast:=-1;
  while(/*count<(COM_BUF) && */!etx_flag && !timer_flag)
    if(sys_GetTimer(TIMER0)<(0.001)) timer_flag:=1;
    tast:=serin();
    if(tast==ETX) etx_flag:=1;
    if(tast>0) buffer[count++]:=tast;
  wend
  if(!timer_flag)
    if((count - 1)==COM_BUF)
      serout(ACK);
      etx_flag:=0;
    else
      serout(NAK);
      nak_flag:=1;
    endif
  else
    serout(NAK);
    timer_flag:=1;
  endif
  count:=0;
  tast:=-1;
endfunc
//funzione principale
func main()
  inizializza();
  txt_Set(TEXT_COLOUR, BLUE);
  repeat
    count:=0;
    while((tast:=serin())!=STX);
    nak_flag:=0;                            //resetta le flag di errore
```

```
        timer_flag:=0;
        etx_flag:=0;
        buffLoad();
        if(!nak_flag && !timer_flag && !etx_flag)       //controlla le flag prima di passare
          buffRead();                                   //alla decodifica del buffer
        endif
    forever
while(1);
endfunc
```

Appendice B
Codice Sorgente Micro MC9S08DZ128

```
#include <hidef.h>      /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */
#include <stdio.h>
#include <stdlib.h>
#include "MCUinit.h"
#include <math.h>


#ifdef __cplusplus
 extern "C"
#endif


//definizione costanti
const char ACK=0x06;
const char NACK=0x15;
const char ETX=0x03;
const char STX=0x02;
const char ESC=0x1B;
//fine costanti


//definizione funzioni
void MCU_init(void);                    /* Device initialization function declaration */
void imre(void);                        //rinfresca il watchdog
void tx_sc1(char character);            //trasmette seriale 1 selezionata in tx
void tx_sc2(char character);            //trasmette seriale 2 selezionata in tx
void tx(int selett,char vett[],int vett_dim);    //trasmette nella seriale passata come parametro
char rx_sc1(void);                      //riceve seriale 1 selezionata in rx
char rx_sc2(void);                      //riceve seriale 2 selezionata in rx
char rx(char selett);                   //riceve nella seriale passata come parametro
void delay(int modulo);                 //ritardo variabile (modulo * 6.4us)
void display16x2(char vett[]);
float adcconversion(int adcline);       //conversione analogico digitale del canale passato come
int temperature(void);                  //parametro esplicito//fine def funzioni
void tx_i2c (void);
void rx_i2c (void);
//fine def funzioni

//funzione principale
void main(void) {



  //variabili locali main
  char sadcval[9],buffer[100],i,ricezione,flag=0,gradi[3];
  int ret_val,temperatura;
  float c=0.0,adcval;
  //fine variabili locali main

  MCU_init();                           //chiamata per l'inizializzazione
  delay(31250);                         //permette il reset del display
  PTJD_PTJD6=1;                         //accendo il display OLED
  PTBD_PTBD2=1;                         //abilito le uscite degli switch
```

```
PTBD_PTBD3=1;                          //abilito un led verde
PTCD_PTCD7=1;                          //abilito il secondo led verde
for(i=0;i<35;i++){
  delay(31250);                        //ogni ciclo for aggiuntivo sono +200ms
}

/* loop forever */
for(;;) {
   ret_val=sprintf(buffer,"%c%s%c",STX,"014/%022309772033",ETX);
   tx(1,buffer,19);
   ret_val=sprintf(buffer,"%c%s%c%c",STX,"002/",ESC,ETX);
   tx(1,buffer,7);
   ret_val=sprintf(buffer,"%c%s%c",STX,"029/+02A  questo e'il messaggio ",ETX);
   tx(1,buffer,34);
   ret_val=sprintf(buffer,"%c%s%c",STX,"012/+09L JOFA  ",ETX);
   tx(1,buffer,17);
   ret_val=sprintf(buffer,"%c%s%c",STX,"004/-02",ETX);
   tx(1,buffer,9);
   ret_val=sprintf(buffer,"%c%s%c",STX,"013/+20B TEST OK",ETX);
   tx(1,buffer,18);

   //grafico
   ret_val=sprintf(buffer,"%c%s%c%c",STX,"002/",ESC,ETX);
   tx(1,buffer,7);
   ret_val=sprintf(buffer,"%c%s%c",STX,"022~+TENSIONE  TEMPO     ",ETX);
   tx(1,buffer,27);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p018050",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p030060",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p040020",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p070080",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p075200",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p120200",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p130100",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p200180",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"003~cA",ETX);
   tx(1,buffer,8);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p018090",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p030100",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p040060",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p070120",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p080230",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p120230",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p140140",ETX);
   tx(1,buffer,13);
   ret_val=sprintf(buffer,"%c%s%c",STX,"008~p200220",ETX);
   tx(1,buffer,13);
```

```
      for(i=0;i<2;i++){
        delay(15625);
      }
      PTBD_PTBD3=0;
      PTCD_PTCD7=1;
      for(i=0;i<2;i++){
        delay(15625);
      }
      PTBD_PTBD3=1;
      PTCD_PTCD7=0;
      adcval=((adcconversion(4)*5.0)/4095.0);
      ret_val=sprintf(sadcval,"%f",adcval);
      SCI1C2_RE=1;
      ret_val=sprintf(buffer,"%c%s%s%c",STX,"020/+27PValore=",sadcval,ETX);
      tx(1,buffer,25);
      ricezione=rx(1);
    /* temperatura=temperature();
      ret_val=sprintf(gradi,"%3d",temperatura);
      ret_val=sprintf(buffer,"%c%s%s%c",STX,"019/+02Ytemperatura=",gradi,ETX);
      tx(1,buffer,24);
      delay(50000);    */
      //PTLD_PTLD5=1;                    //buzzer
      //delay(5000);
      //PTLD_PTLD5=0;
   //    for(i=0; i<10; i++) ricezione=rx(1);
    }
    /* loop forever */
}
//fine funzione main


void imre(void){
  if(PTLD_PTLD3==1) PTLD_PTLD3=0;    //rinfresca il bit di controllo del circuito watchdog
                                     //esterno
  else PTLD_PTLD3=1;                 //se è alto lo pone basso e viceversa
}


void tx_sc1(char character){
  while(!SCI1S1_TDRE);               //attende che il buffer di invio sia vuoto
  imre();                            //chiama l'aggiornamento del circuito di watchdog
  SCI1D=character;                   //salva il carattere nel buffer di invio
  while(!SCI1S1_TC);                 //attente il completamento della trasmissione
}


void tx_sc2(char character){
  while(!SCI2S1_TDRE);               //attende che il buffer di invio sia vuoto
  imre();                            //chiama l'aggiornamento del circuito di watchdog
  SCI2D=character;                   //salva il carattere nel buffer di invio
  while(!SCI2S1_TC);                 //attente il completamento della trasmissione
}


void tx(int selett,char vett[],int vett_dim){
  int i=0;
  if(selett==1){                     //seleziona la periferica seriale SC1
    SCI1C2_TE=1;                     //attiva la periferica
    for(i=0;i<vett_dim;i++){         //scorre il buffer da trasettere
      tx_sc1(vett[i]);               //carica nel buffer il carattere da inviare
```

```
    }
    SCI1C2_TE=0;                    //disabilita la periferica
    delay(10000);                   //ritardo per comunicazione con il display
  }                                 //(valori inferiori a 7000 possono causare perdita di
                                    //informazione)
  if(selett==2){
    SCI2C2_TE=1;
    for(i=0;i<vett_dim;i++){
      tx_sc2(vett[i]);
    }
    SCI2C2_TE=0;
    //delay(31250);
  }
}



char rx_sc1(){
  char flag,ricezione;
  flag=0;
  while(!flag){                 //rimane nel ciclo finchè non trova un dato nel buffer di ricezione
    imre();
    if(SCI1S1_RDRF){            //se c'è un dato nel buffer lo memorizza
      ricezione=SCI1D;
      flag=1;                   //setta la flag per uscire dal ciclo
    }
  }
  flag=0;
  return ricezione;             //restituisce il dato ricevuto
}


char rx_sc2(){
  char flag,ricezione;
  flag=0;
  while(!flag){                 //rimane nel ciclo finchè non trova un dato nel buffer di ricezione
    imre();
    if(SCI2S1_RDRF){            //se c'è un dato nel buffer lo memorizza
      ricezione=SCI2D;
      flag=1;                   //setta la flag per uscire dal ciclo
    }
  }
  flag=0;
  return ricezione;             //restituisce il dato ricevuto
}


char rx(char selett){           //seleziona la linea di ricezione desiderata
  if(selett==1) return rx_sc1();
  if(selett==2) return SCI2D;
}


void display16x2(char vett[]){
  char i;
  PTLD_PTLD1=1;
  PTLD_PTLD2=0;
  for(i=0;i<16;i++){
    PTKD=vett[i];
    PTLD_PTLD0=1;
```

```
  }
}



void delay(int modulo){
  TPM1SC_TOF=0;                  //ripulisce la Timer overflow flag
  TPM1MOD=modulo;                //salva nel registro il numero di conteggi da effettuare
  while(!TPM1SC_TOF);            //attende la fine del conteggio
  imre();                        //chiamata alla funzione di aggiornamento watchdog
  TPM1SC_TOF=0;                  //ripulisce la Timer overflow flag
}


float adcconversion(int adcline){
  int mask;
  mask=224;                      //Maschera i bit di controllo-stato
  imre();                        //chiamata alla funzione di aggiornamento watchdog
  ADCSC1=(ADCSC1 & mask)+adcline;      //avvia la conversione selezionando una delle linee analogiche
  while(!ADCSC1_COCO);           //attende la fine conversione
  return ((ADCRL*5.0)/255.0);    //restituisce il valore di tensione come numero di conteggi
}


int temperature(void){
  int temperatura;
  float vdd,vtemp;
  vdd=adcconversion(27)/**(5.0/1.20)*/;
  vtemp=adcconversion(26)/*/5.0*vdd*/;
  if(vtemp<1.396)
    temperatura=(25-((vtemp-1.396)/0.003266));
  else
    temperatura=(25-((vtemp-1.396)/0.003638));
  return temperatura;
}


void tx_i2c (void){
if(!IIC2S_BUSY){
  PTGD_PTGD7=1;
  IIC2C1_IICEN=1;                //abilita il modulo i2c
  IIC2C1_TX=1;                   //Trasmissione selezionata
  IIC2C1_RSTA=0;
  IIC2C1_MST=1;                  //Master mode abilitato
  delay(1);
  IIC2D=0xA0;                    //Indirizzo memoria, selezione pacchetto, bit di scrittura/lettura(pari/dispari)
  while((!IIC2S_TCF) || IIC2S_RXAK);
  IIC2D=0x16;
  while((!IIC2S_TCF) || IIC2S_RXAK);
  IIC2D='B';
  while((!IIC2S_TCF) || IIC2S_RXAK);
  IIC2D='E';
  while((!IIC2S_TCF) || IIC2S_RXAK);
  IIC2D='N';
  while((!IIC2S_TCF) || IIC2S_RXAK);
  IIC2D='O';
  while((!IIC2S_TCF) || IIC2S_RXAK);
  IIC2D='N';
  while((!IIC2S_TCF) || IIC2S_RXAK);
```

```
  IIC2D='E';
  while((!IIC2S_TCF) || IIC2S_RXAK);
     PTLD_PTLD5=1;
     delay(5000);
     PTLD_PTLD5=0;
  IIC2C1_MST=0;
}
while(IIC2S_BUSY);
}

void rx_i2c (void){
  char temp[16],i;
  if(!IIC2S_BUSY){
    PTGD_PTGD7=1;
    IIC2C1_IICEN=1;                 //abilita il modulo i2c
    IIC2C1_TX=1;                    //Trasmissione selezionata
    IIC2C1_MST=1;                   //Master mode abilitato START BIT
    delay(1);
    IIC2D=0xA2;                     //Indirizzo memoria, selezione pacchetto, bit di scrittura o lettura
    while(!IIC2S_TCF || IIC2S_RXAK);
    PTGD_PTGD7=0;
    delay(1);
    IIC2D=0xA3;
    IIC2C1_TX=0;                    //ricezione selezionata
    while(!IIC2S_TCF || IIC2S_RXAK);
    for(i=0;i<16;i++){
      temp[i]=IIC2D;
    }
  }
}
```

Appendice C
Codice Assembly RASM05

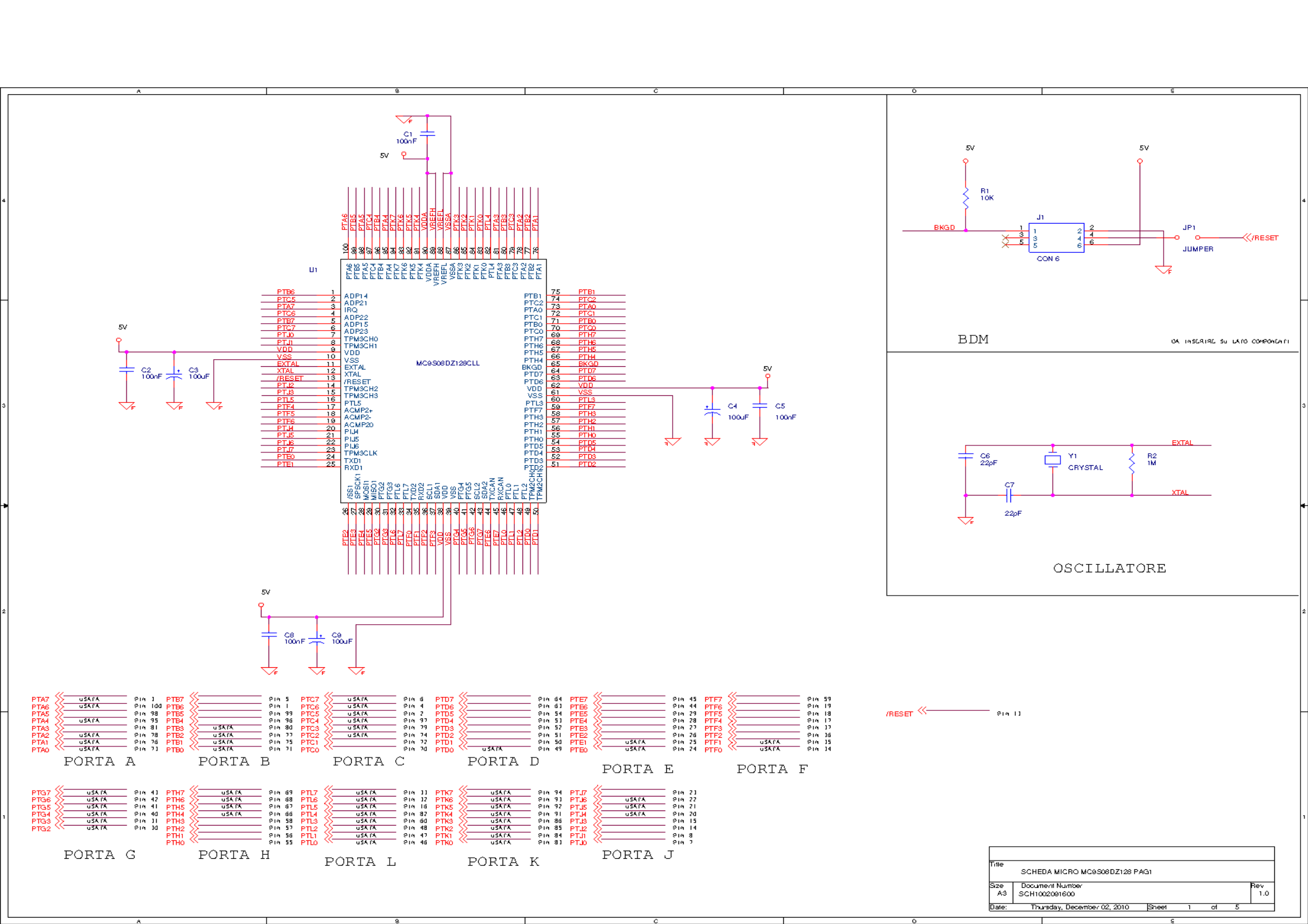
```
*LDST1
*CARICA LA STRINGA IN SERIALE PER SCRIVERLA SUL DISPLAY OLED(LEGGE LA *STRINGA JOF1)
LDST1
   CLR   PROVV          ;azzera il puntatore
   LDA   #10H           ;Inizializza la variabile SCRA2 a 16
   STA   SCRA2
LDSTA1
   LDX   PROVV          ;carica il puntatore
   LDA   JOF1,X         ;leggo la stringa da visualizzare e salvo il carattere in datx
   STA   DATX
   JSR   TX8BB          ;salto all'invio del carattere
LDSTB1
   INC   PROVV          ;aggiorna le variabili del loop
   DEC   SCRA2
   BNE   LDSTA1         ;se nn è finita la scrittura torna a LDSTA1
   RTS
*LDST2
*CARICA LA STRINGA IN SERIALE PER SCRIVERLA SUL DISPLAY OLED(LEGGE LA *STRINGA JOF3)
LDST2
   CLR   PROVV          ;azzera il puntatore
   LDA   #10H           ;Inizializza la variabile SCRA2 a 16
   STA   SCRA2
LDSTA2
   LDX   PROVV          ;carica il puntatore
   LDA   JOF3,X         ;leggo la stringa da visualizzare e salvo il carattere in datx
   STA   DATX
   JSR   TX8BB          ;salto all'invio del carattere
LDSTB2

   INC   PROVV          ;aggiorna le variabili del loop
   DEC   SCRA2
   BNE   LDSTA2         ;se nn è finita la scrittura torna al LDSTA2
   RTS


*ROUTINE DI TRASMISSIONE PER OLED ATTRAVERSO IL CONNETTORE DISPLAY *VECCHIO
TX8BB
   LDA   #0FFH
   STA   REGB
   JSR    DEL005        ;ritardo di 5ms
   LDX   #8             ;inizializzo x per contare 8 bit     TEMPI (usec.)
   LDA   PORTB          ;invio il bit di start
   AND   ANDTX          ;linea a 1
   STA   PORTB          ;PORTA PER TX                                    __>
   LDA   DATX           ;                                                3
   LDA   DATX           ;6 usecondi per i tempi                    3
   JSR    DEL1TR              ;Tempo/bit -27 cicli -8                          *
*27 MICROS.FINO A PROSSIMA OUT
TX8BB0
   LSR    DATX          ;IN CY I BITS DA 0 A7                          [5]
   LDA   PORTB          ;PREPARA PER SETTARE BIT DI OUT          [3]
   BCS    TX8BB1        ;CY=BIT A 1                               [3]
   AND   ANDTX          ;LINEA A 1                               [3]
   BRA   TX8BB2          ;                                        [3]
```

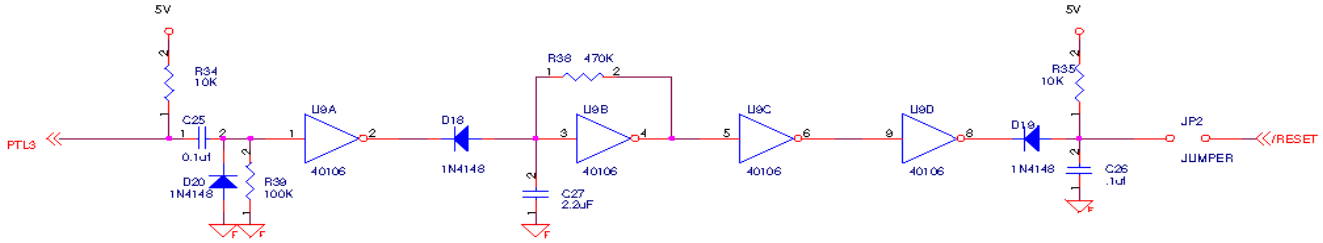
```
TX8BB1
   ORA    ORATX          ;LINEA A 0                                    3
   BRA    TX8BB2               ;                                       3
TX8BB2
   STA    PORTB          ;OUT                                            [4]__>
   JSR    DEL1TR               ;TEMPO/BIT -27 cicli -8                  [*]
   DECX                  ;FINITI 8 BITS   ?                 [3]
   BNE    TX8BB0         ;                           [3]
   LDA    DATX       ;PER TEMPI                          3
   STA      DATX               ;                        4
   STA      DATX               ;                        4
   LDA     PORTB         ;                           3
   ORA    ORATX          ;STOP, LINEA A 0                 3
   STA    PORTB          ;                           4 __>
   JSR      IMRE               ;INCLUSA JSR 21 CICLI                    21
   JSR     DEL1TR              ;TEMPO-BIT - 17.5 MICROS.          *
   JSR      IMRE               ;INCLUSA JSR 21 CICLI                    21
   LDA    DATX
   CLR    REGB           ;                           3
   RTS                   ;                           6
*****************************************
*ROUTINE DI RICEZIONE PER OLED ATTRAVERSO IL CONNETTORE DISPLAY *VECCHIO
RX8BBA
   JSR    SETTO               ;SETTA TIME OUT
   JSR     IMRE
RX8BBX
   LDA   PORTB          ;
   AND   ANDRX          ;                           [3]:
   BEQ   RX8BB2         ;0=START                    [3]:
   DEC   TIML           ;                           [5]
   BNE   RX8BBX         ;                           [3]
   LDA   PORTC          ;IMP.RESET                   3
   EOR   #8H            ;COMPLEM.BIT 3,RESET         2
   STA   PORTC          ;                           4
   DEC   TIMH           ;TIME - OUT ?                6
   BNE   RX8BBX                                      3
   LDA   #1             ;FLAG                        2
   STA   TIMOUT                                      5
   RTS                  ;ESCE PER TIME-OUT           6
RX8BB2
   JSR    DEL1RX
   LDA    PORTB
   AND    ANDRX                                      3
  BNE     RX8BBX          ;RIPROVA                   3
  LDA     #8                                         2
  STA     SCRA            ;CONTATORE 8 BITS          4
  LDA     #8                   ;PER TEMPI                    2
  STA     SCRA          ;PER TEMPI                     4
  LDA     #8                   ;PER TEMPI                    2
  STA     SCRA          ;PER TEMPI                     4
  JSR      DEL1TR
RX8BB1
  LDA     PORTB
  AND     ANDRX          ;LEGGE DATO E FA ANI COL SUO BIT    [3]
  BEQ     RX8BB3          ;BIT VALE 0                        [3]
  SEC                     ;BIT VALE 1                        [2]
  BRA     RX8BB4                                             [3]
RX8BB3
  CLC                                                        2
  BRA     RX8BB4                                             3
```

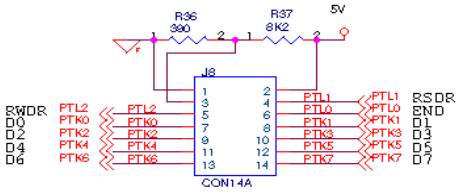
```
RX8BB4
  ROR    SCRA1          ;DATO ENTRA DA SX. IN SCRA1          [5]
  JSR    DEL1TR             ;TEMPO/BIT -27 cicli -8                  [*]
  DEC    SCRA           ;DECREMENTATO CONTATORE          [5]
  BNE    RX8BB1         ;(ESCE AL CENTRO DEL BIT DI STOP)  [3]
  CLR    TIMOUT         ;FLAG
  RTS

****************************************

OLED
  LDA    STX                ;carica STX per iniziare la comunicazione
  STA    DATX
  JSR    TX8BB
```

Appendice D - Circuiti

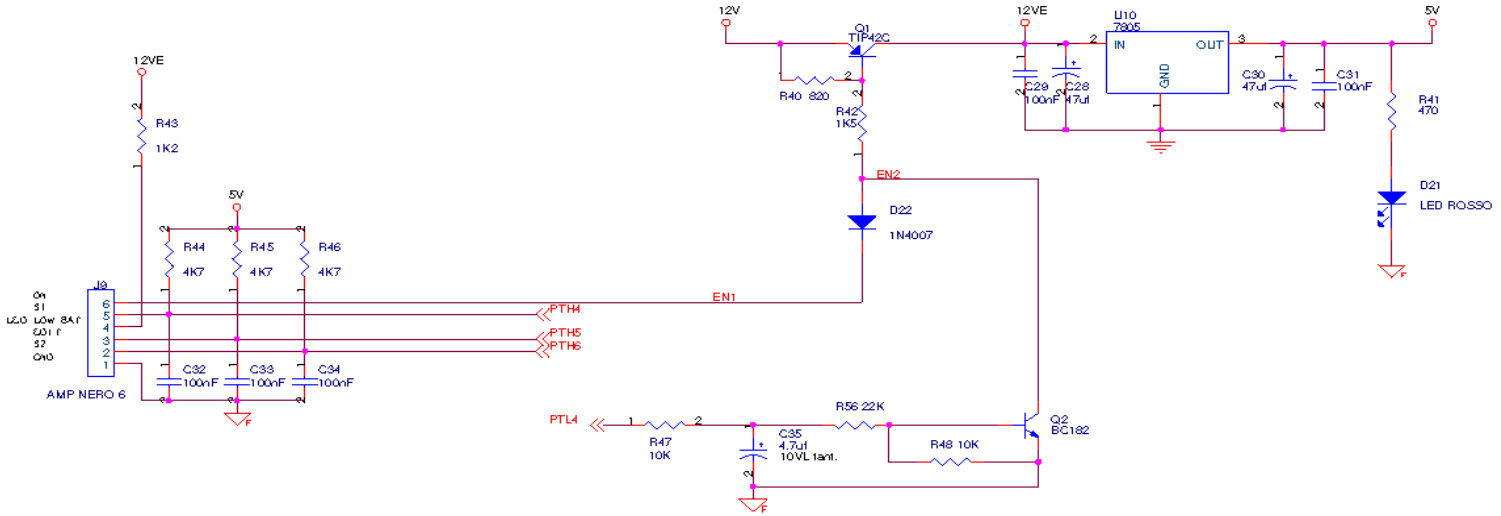
Indice delle illustrazioni:


U1
MC9S08DZ128CLL
C1
100nF
C2
100nF
C3
100uF
C4
100uF
C5
100nF
C8
100nF
C9
100uF
5V
R1
10K
BKGD
J1
CON 6
JP1
JUMPER
/RESET
BDM
DA INSERIRE SU LATO COMPONENTI
C6
22pF
Y1
CRYSTAL
R2
1M
C7
22pF
EXTAL
XTAL
OSCILLATORE
PORTA A
PORTA B
PORTA C
PORTA D
PORTA E
PORTA F
PORTA G
PORTA H
PORTA L
PORTA K
PORTA J
Title
SCHEDA MICRO MC9S08DZ128 PAG1
Size
A3
Document Number
SCH1002091600
Rev
1.0
Date:
Thursday, December 02, 2010
Sheet
1
of
5



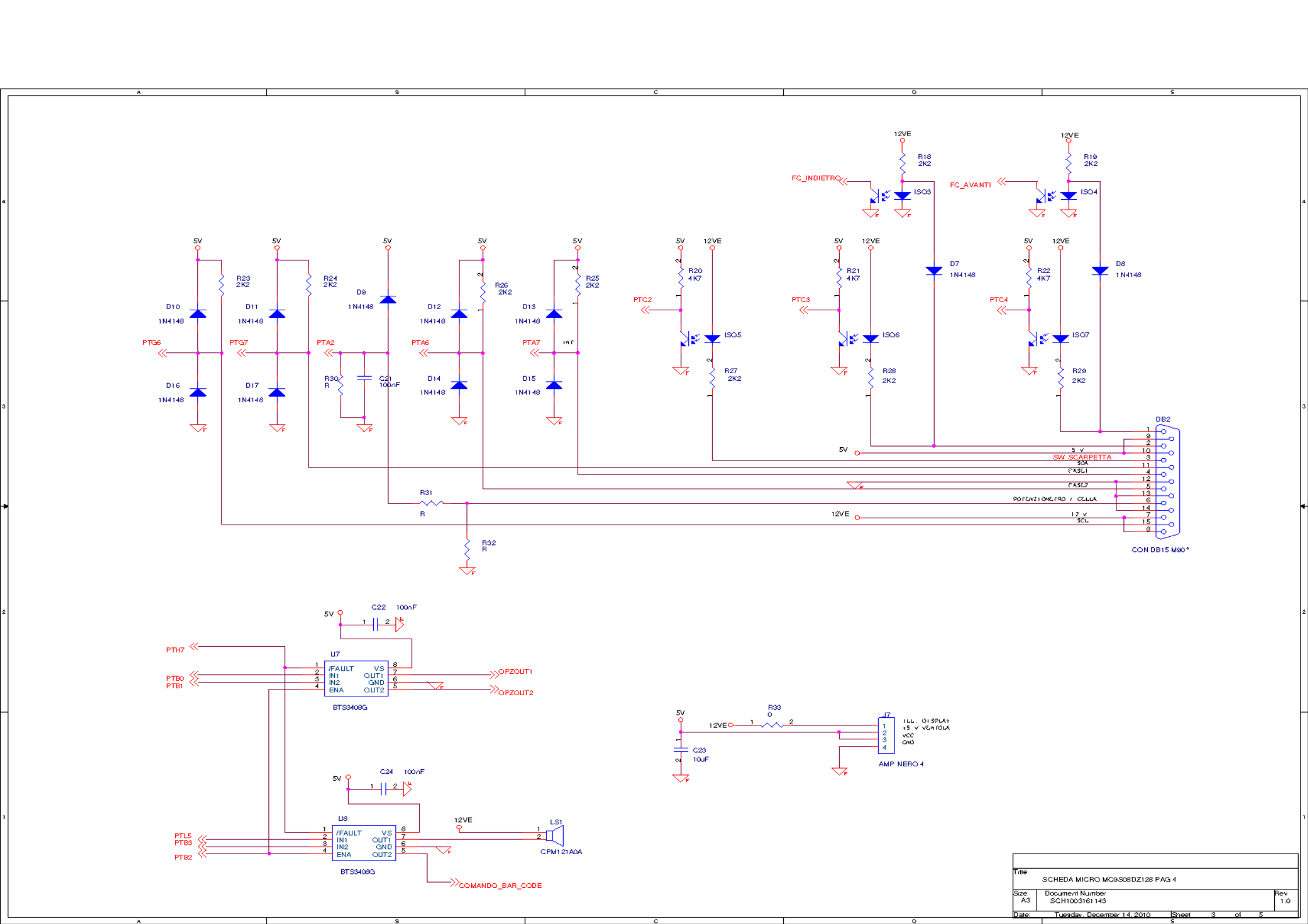


DISPLAY

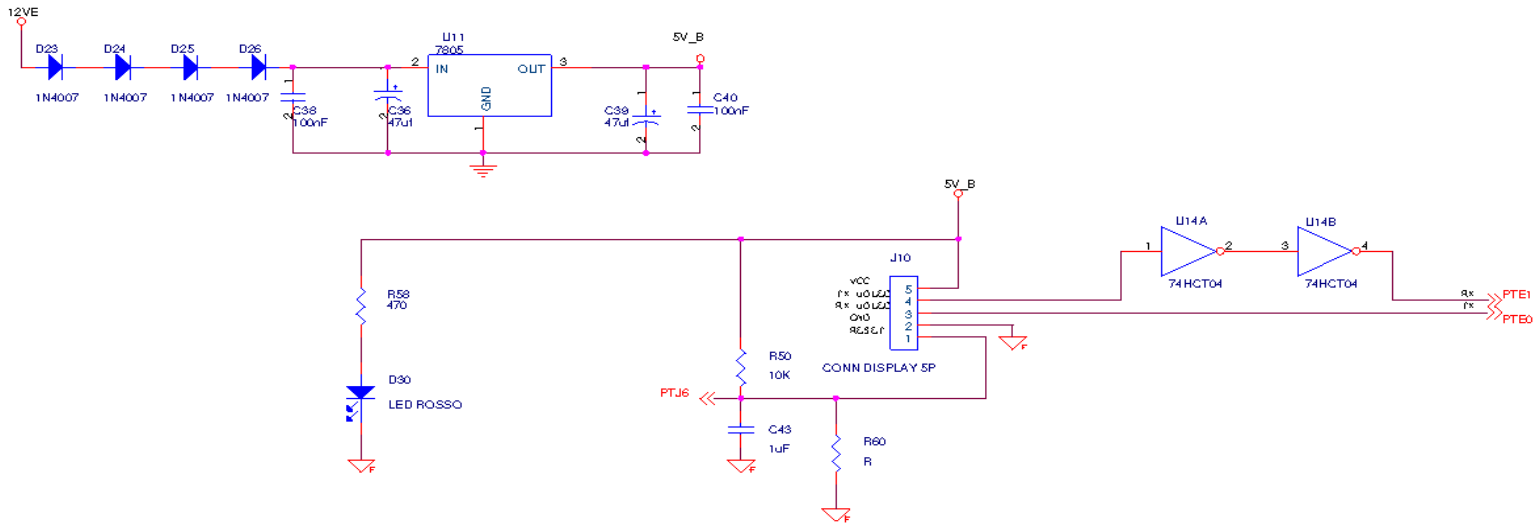


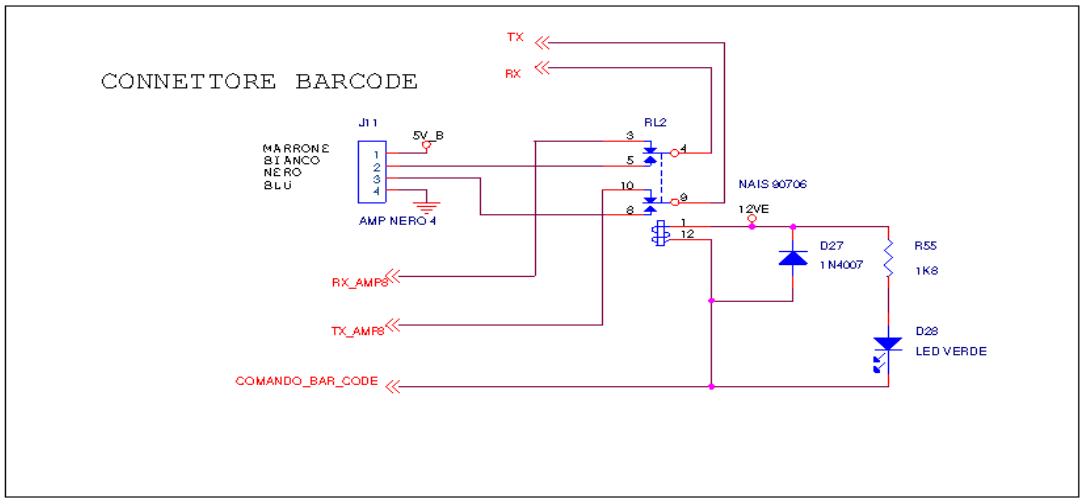
ALIMENTAZIONE E PULSANTI

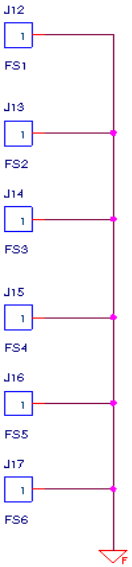
Title	SCHEDA MICRO MC9S08DZ128 PAG 2	
Size A3	Document Number SCH1003031619	Rev 1.0
Date: Thursday, December 23, 2010	Sheet 4 of 5	



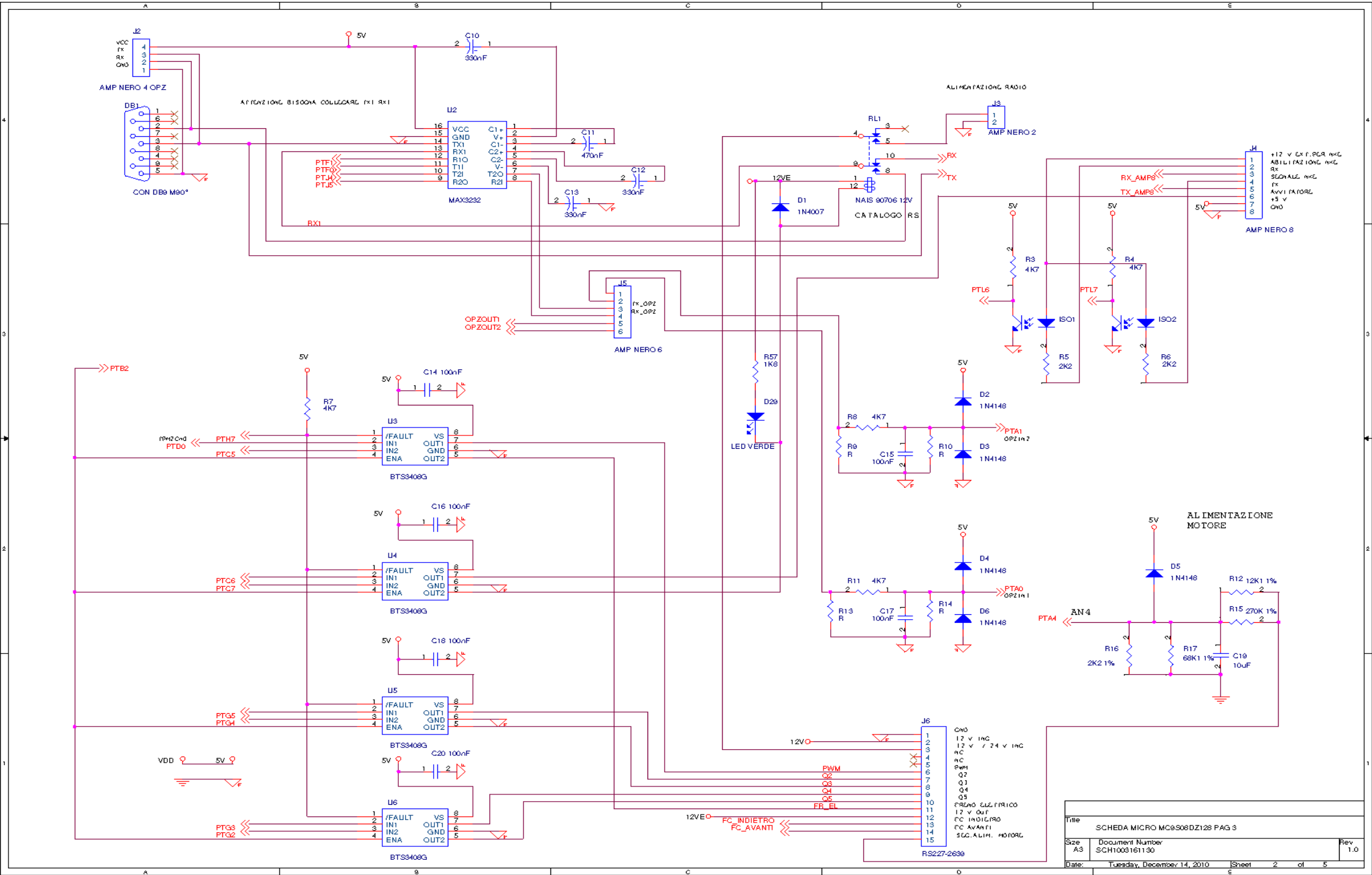
Title	SCHEDA MICRO MC9S08DZ128 PAG 4	
Size A3	Document Number SCH1003161143	Rev 1.0
Date: Tuesday, December 14, 2010	Sheet 3 of 5	

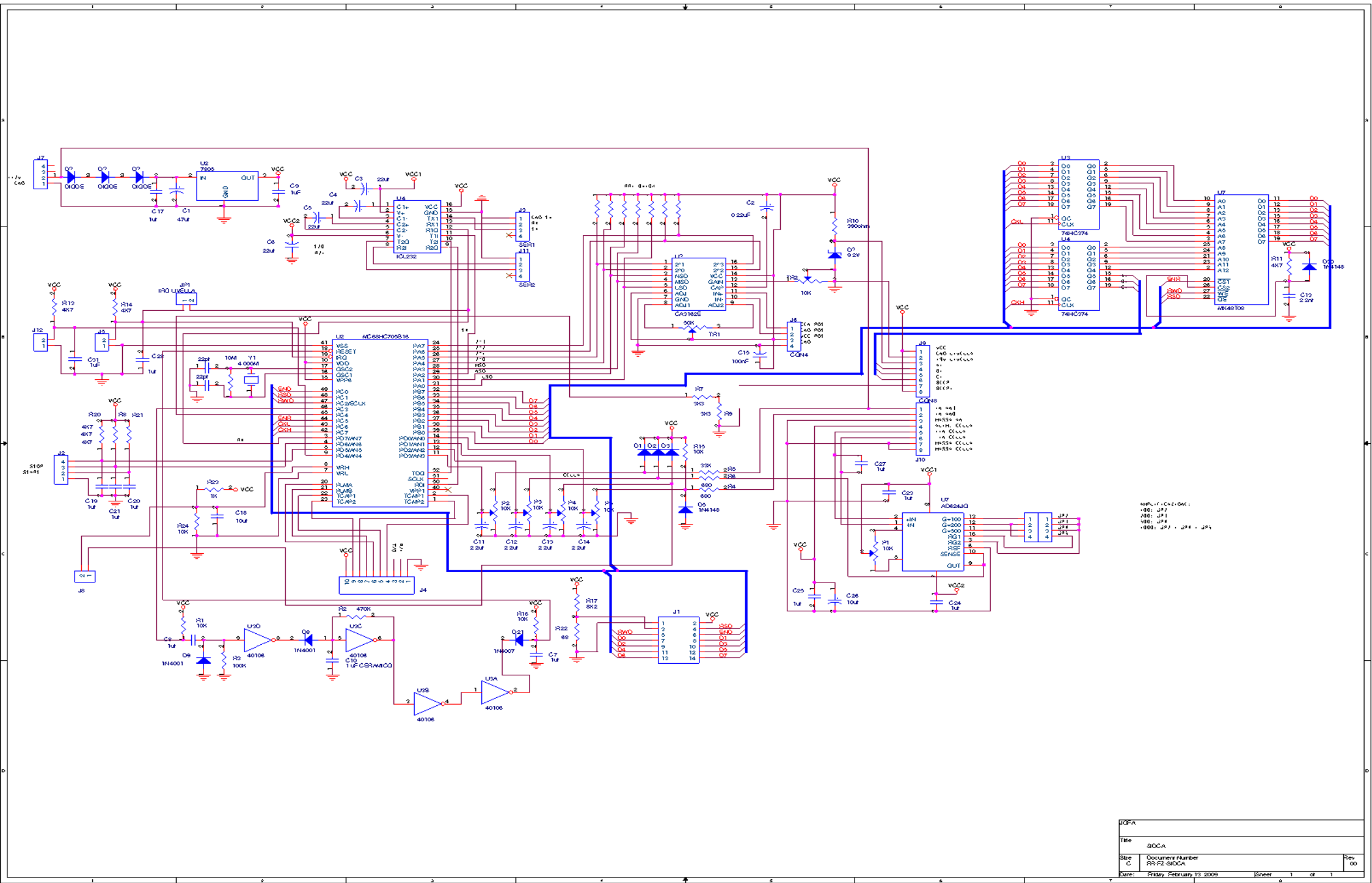




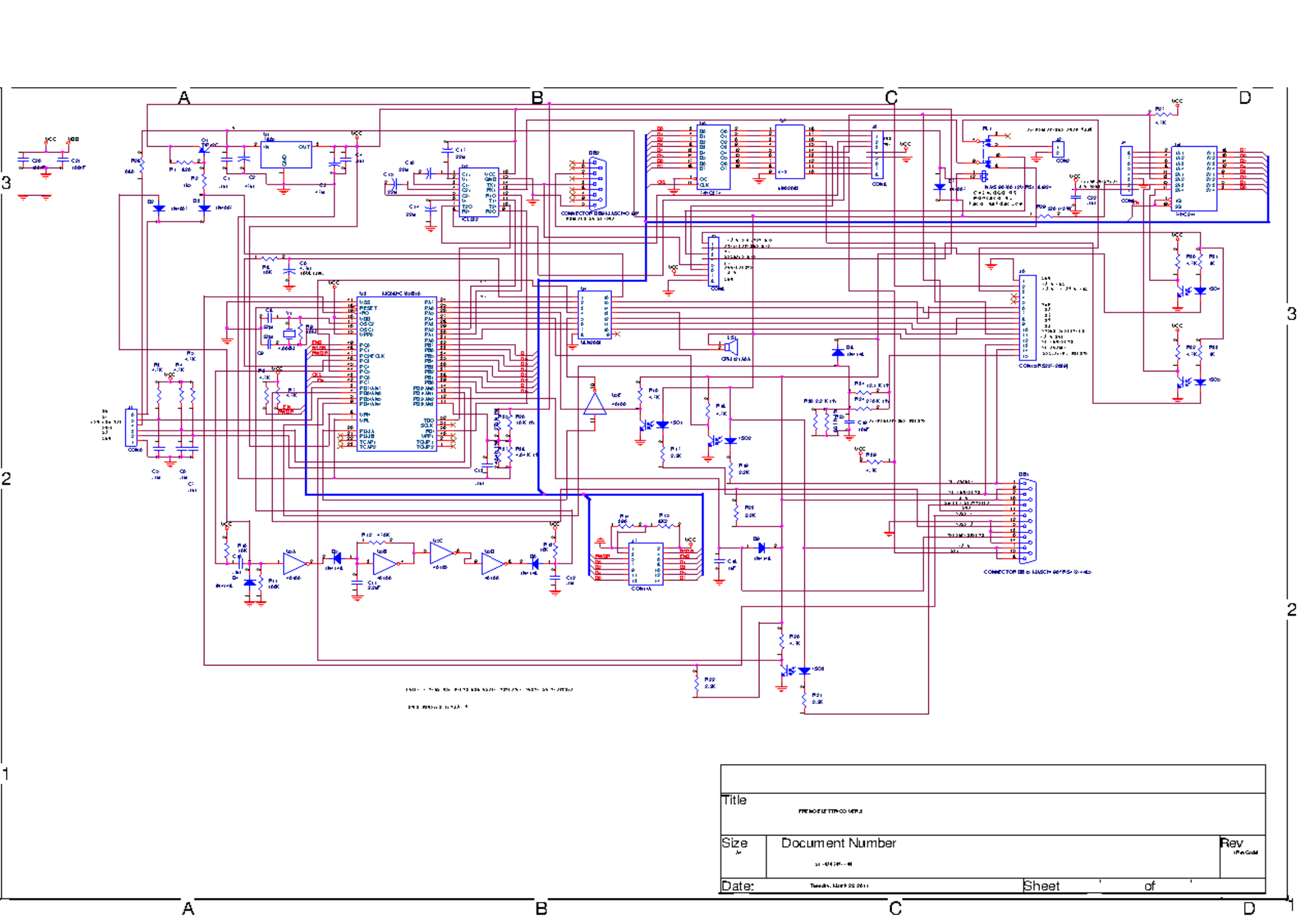


Title: SCHEDA MICRO MC9S08DZ128 PAG 5		
Size: A3	Document Number: SCH1003161143	Rev: 1.0
Date: Friday, January 28, 2011	Sheet 5 of 5	

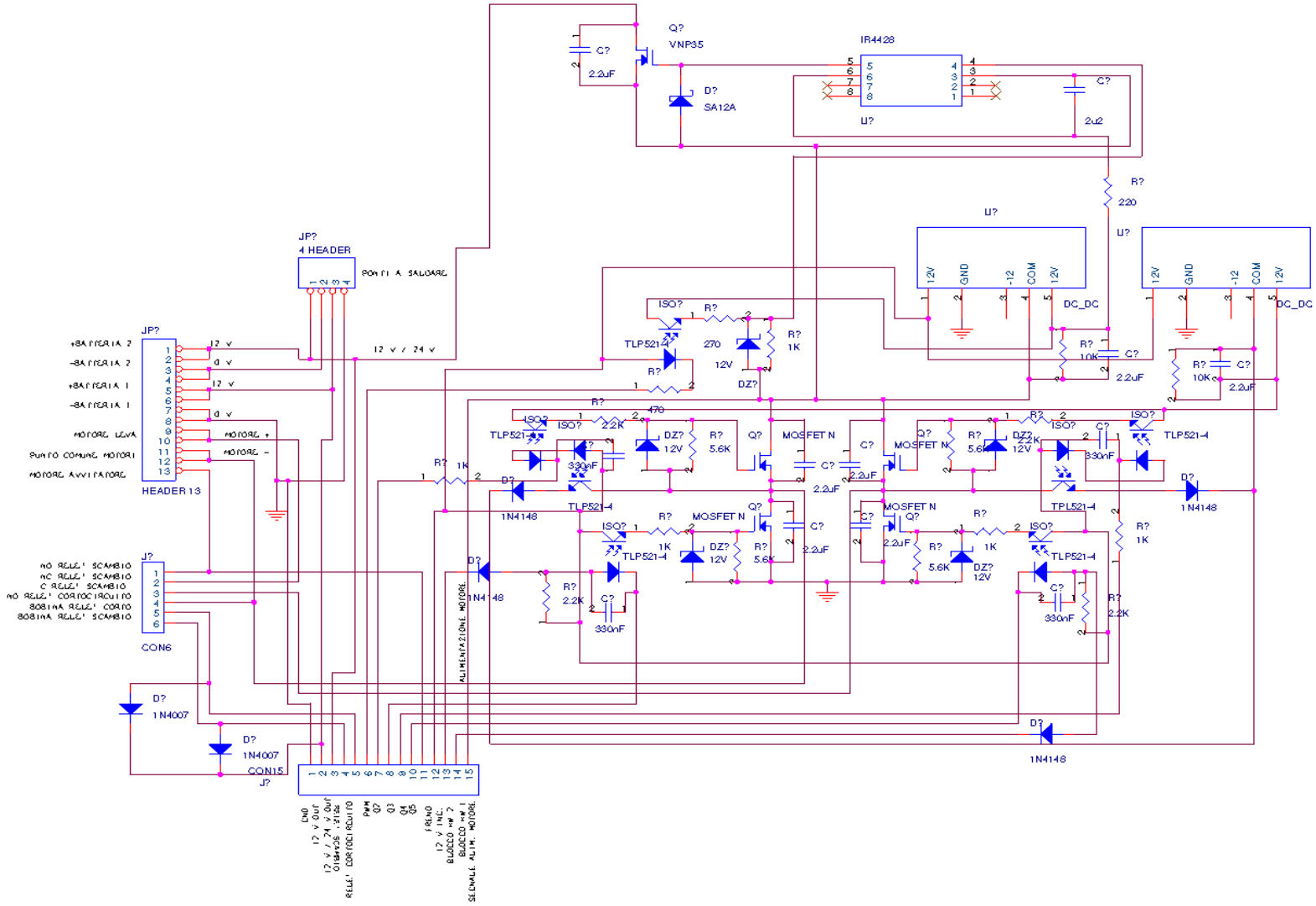

J2
AMP NERO 4 OPZ
DB1
CON DB9 M90°
ATTENZIONE BISOGNA COLLEGARE TX1 RX1
U2
MAX3232
C10
330nF
C11
470nF
C12
330nF
C13
330nF
5V
PTF1
PTF0
PTJ4
PTJ5
RX1
ALIMENTAZIONE RADIO
J3
AMP NERO 2
RL1
NAIS 90706 12V
CATALOGO RS
RX
TX
12VE
D1
1N4007
J4
AMP NERO 8
RX_AMP8
TX_AMP8
R3
4K7
R4
4K7
PTL6
PTL7
ISO1
ISO2
R5
2K2
R6
2K2
J5
AMP NERO 6
OPZOUT1
OPZOUT2
R57
1K8
D29
LED VERDE
PTB2
R7
4K7
C14 100nF
U3
BTS3408G
PTH7
PTD0
PTC5
R8
4K7
R9
R
C15
100nF
R10
R
D2
1N4148
D3
1N4148
PTA1
C16 100nF
U4
BTS3408G
PTC6
PTC7
R11
4K7
R13
R
C17
100nF
R14
R
D4
1N4148
D6
1N4148
PTA0
C18 100nF
U5
BTS3408G
PTG5
PTG4
C20 100nF
U6
BTS3408G
PTG3
PTG2
VDD
ALIMENTAZIONE
MOTORE
D5
1N4148
R12 12K1 1%
R15 270K 1%
R16
2K2 1%
R17
68K1 1%
C19
10uF
AN4
PTA4
J6
RS227-2639
12V
PWM
Q2
Q3
Q4
Q5
FB_EL
12VE
FC_INDIETRO
FC_AVANTI
Title
SCHEDA MICRO MC9S08DZ128 PAG 3
Size
A3
Document Number
SCH1003161130
Rev
1.0
Date:
Tuesday, December 14, 2010
Sheet
2
of
5



UOFA		
Title	SIOCA	
Size C	Document Number RR-PZ-SIOCA	Rev 00
Date: Friday, February 13, 2009	Sheet 1 of 1	



Title		
Size	Document Number	Rev
Date:		Sheet of



Tuesday, September 05, 2000
C:\LAVORO\ANDREA\DISEGNI ORCAD\FRENO_ELETTRICO_VER_4_POT\NEW_FR_POT.DSN

Title	FRENO ELETTRICO SCHEDA POTENZA VER_4	
Size A3	Document Number SCHO101301700	Rev
Date: Tuesday, September 05, 2000	Sheet 1 of 1	

Appendice E
Tabelle codici mnemonici istruzioni Assembly

Table 10-2 Register/memory instructions

		Addressing modes																	
		Immediate			Direct			Extended			Indexed (no offset)			Indexed (8-bit offset)			Indexed (16-bit offset)		
Function	Mnemonic	Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles	Opcode	# Bytes	# Cycles
Load A from memory	LDA	A6	2	2	B6	2	3	C6	3	4	F6	1	3	E6	2	4	D6	3	5
Load X from memory	LDX	AE	2	2	BE	2	3	CE	3	4	FE	1	3	EE	2	4	DE	3	5
Store A in memory	STA				B7	2	4	C7	3	5	F7	1	4	E7	2	5	D7	3	6
Store X in memory	STX				BF	2	4	CF	3	5	FF	1	4	EF	2	5	DF	3	6
Add memory to A	ADD	AB	2	2	BB	2	3	CB	3	4	FB	1	3	EB	2	4	DB	3	5
Add memory and carry to A	ADC	A9	2	2	B9	2	3	C9	3	4	F9	1	3	E9	2	4	D9	3	5
Subtract memory	SUB	A0	2	2	B0	2	3	C0	3	4	F0	1	3	E0	2	4	D0	3	5
Subtract memory from A with borrow	SBC	A2	2	2	B2	2	3	C2	3	4	F2	1	3	E2	2	4	D2	3	5
AND memory with A	AND	A4	2	2	B4	2	3	C4	3	4	F4	1	3	E4	2	4	D4	3	5
OR memory with A	ORA	AA	2	2	BA	2	3	CA	3	4	FA	1	3	EA	2	4	DA	3	5
Exclusive OR memory with A	EOR	A8	2	2	B8	2	3	C8	3	4	F8	1	3	E8	2	4	D8	3	5
Arithmetic compare A with memory	CMP	A1	2	2	B1	2	3	C1	3	4	F1	1	3	E1	2	4	D1	3	5
Arithmetic compare X with memory	CPX	A3	2	2	B3	2	3	C3	3	4	F3	1	3	E3	2	4	D3	3	5
Bit test memory with A (logical compare)	BIT	A5	2	2	B5	2	3	C5	3	4	F5	1	3	E5	2	4	D5	3	5
Jump unconditional	JMP				BC	2	2	CC	3	3	FC	1	2	EC	2	3	DC	3	4
Jump to subroutine	JSR				BD	2	5	CD	3	6	FD	1	5	ED	2	6	DD	3	7

Table 10-3 Branch instructions

		Relative addressing mode		
Function	Mnemonic	Opcode	# Bytes	# Cycles
Branch always	BRA	20	2	3
Branch never	BRN	21	2	3
Branch if higher	BHI	22	2	3
Branch if lower or same	BLS	23	2	3
Branch if carry clear	BCC	24	2	3
(Branch if higher or same)	(BHS)	24	2	3
Branch if carry set	BCS	25	2	3
(Branch if lower)	(BLO)	25	2	3
Branch if not equal	BNE	26	2	3
Branch if equal	BEQ	27	2	3
Branch if half carry clear	BHCC	28	2	3
Branch if half carry set	BHCS	29	2	3
Branch if plus	BPL	2A	2	3
Branch if minus	BMI	2B	2	3
Branch if interrupt mask bit is clear	BMC	2C	2	3
Branch if interrupt mask bit is set	BMS	2D	2	3
Branch if interrupt line is low	BIL	2E	2	3
Branch if interrupt line is high	BIH	2F	2	3
Branch to subroutine	BSR	AD	2	6

Table 10-6 Control instructions

		Inherent addressing mode		
Function	**Mnemonic**	**Opcode**	**# Bytes**	**# Cycles**
Transfer A to X	TAX	97	1	2
Transfer X to A	TXA	9F	1	2
Set carry bit	SEC	99	1	2
Clear carry bit	CLC	98	1	2
Set interrupt mask bit	SEI	9B	1	2
Clear interrupt mask bit	CLI	9A	1	2
Software interrupt	SWI	83	1	10
Return from subroutine	RTS	81	1	6
Return from interrupt	RTI	80	1	9
Reset stack pointer	RSP	9C	1	2
No-operation	NOP	9D	1	2
Stop	STOP	8E	1	2
Wait	WAIT	8F	1	2

Table 10-4 Bit manipulation instructions

		Addressing Modes					
		Bit set/clear			Bit test and branch		
Function	**Mnemonic**	**Opcode**	**# Bytes**	**# Cycles**	**Opcode**	**# Bytes**	**# Cycles**
Branch if bit n is set	BRSET n (n=0–7)				2•n	3	5
Branch if bit n is clear	BRCLR n (n=0–7)				01+2•n	3	5
Set bit n	BSET n (n=0–7)	10+2•n	2	5			
Clear bit n	BCLR n (n=0–7)	11+2•n	2	5			

Table 10-5 Read/modify/write instructions

		Addressing modes														
		Inherent (A)			Inherent (X)			Direct			Indexed (no offset)			Indexed (8-bit offset)		
Function	**Mnemonic**	**Opcode**	**# Bytes**	**# Cycles**	**Opcode**	**# Bytes**	**# Cycles**	**Opcode**	**# Bytes**	**# Cycles**	**Opcode**	**# Bytes**	**# Cycles**	**Opcode**	**# Bytes**	**# Cycles**
Increment	INC	4C	1	3	5C	1	3	3C	2	5	7C	1	5	6C	2	6
Decrement	DEC	4A	1	3	5A	1	3	3A	2	5	7A	1	5	6A	2	6
Clear	CLR	4F	1	3	5F	1	3	3F	2	5	7F	1	5	6F	2	6
Complement	COM	43	1	3	53	1	3	33	2	5	73	1	5	63	2	6
Negate (two's complement)	NEG	40	1	3	50	1	3	30	2	5	70	1	5	60	2	6
Rotate left through carry	ROL	49	1	3	59	1	3	39	2	5	79	1	5	69	2	6
Rotate right through carry	ROR	46	1	3	56	1	3	36	2	5	76	1	5	66	2	6
Logical shift left	LSL	48	1	3	58	1	3	38	2	5	78	1	5	68	2	6
Logical shift right	LSR	44	1	3	54	1	3	34	2	5	74	1	5	64	2	6
Arithmetic shift right	ASR	47	1	3	57	1	3	37	2	5	77	1	5	67	2	6
Test for negative or zero	TST	4D	1	3	5D	1	3	3D	2	4	7D	1	4	6D	2	5
Multiply	MUL	42	1	11												

Ringraziamenti

Sento il dovere e ho il piacere di ricordare e di ringraziare di cuore alcune persone che mi hanno accompagnato in quest'avventura durata forse un po più del dovuto ma ricca di emozioni e di eventi che hanno segnato la mia vita.
In primis la mia famiglia che mi ha visto crescere e cambiare molte volte in questo periodo universitario, perché mi ha sempre dato tutto, mi ha ascoltato come figlio/fratello/nipote ma anche e soprattutto come amico e ha saputo capirmi come forse nessun altro.
I miei nonni Gildo e Raffaello che mi hanno salutato entrambi durante il primo anno di università ma che hanno strutturato in modo importante la mia persona, perché li porterò per sempre come esempio di correttezza e rispetto.
Alla mia bellissima Serena che ho avuto al mio fianco dall'inizio alla fine, mi ha conosciuto, sopportato e compreso malgrado le innumerevoli vicissitudini che ci hanno colpito, mi auguro che tutto quello che abbiamo costruito possa continuare ancora per molto tempo.
I coscritti, gli amici di sempre e i compagni di scuola che ritrovo spesso nei miei ricordi con estrema gioia e affetto, spero sappiano che ho sempre voluto loro un sacco di bene.
Tutti gli amici e conoscenti di Padova, sono stati per me una dolce compagnia e un appoggio importante, foto video e storie sono a testimonianza del fatto che mi hanno reso felice.
Per ultimi ma non per importanza(!!!) i Furbetti, vecchi e nuovi, perché sono stati la compagnia ideale quasi insperata, perché sono stati amici, coinquilini, consulenti e collaboratori in molteplici occasioni di festa, di studio e di lavoro, saranno il più bel ricordo che avrò di Padova.

Senza tutti voi oggi non sarei come sono (e sarebbe un gran peccato)!!!

Grazie di cuore
Andrea