

## Processore ARM: Strumenti e Applicazioni

Per avere una panoramica sul mondo ARM, faremo innanzitutto riferimento alle principali architetture ARM embedded e ai cosiddetti System-on-Chip (SoC), ovvero circuiti integrati che contengono un intero sistema, quindi un processore centrale, un chipset, i controller per le memorie, la circuiteria relativa all'I/O ed eventualmente sottosistemi di tipo audio o video. Successivamente, porremo l'attenzione ai principali sistemi operativi progettati per l'architettura ARM, ed infine proporremo alcuni esempi di strumenti di sviluppo, compilazione e debug (le cosiddette Tool Chain) specifici per questo tipo di sistemi.

## Processore ARM: Esempi di System-On-Chip

A titolo esemplificativo, presentiamo tre esempi di System-on-Chip basati su ARM realizzati da tre importanti aziende che operano nel settore dei sistemi a microprocessore. Si tenga presente che il mercato è estremamente ampio, e quella presentata è solo una piccola parte dei sistemi disponibili.

- *Samsung Electronics*

La Samsung Electronics ha un importante settore dedicato alla realizzazione di sistemi embedded, prevalentemente per applicazioni relative alla telefonia e ai dispositivi elettronici portatili quali lettori MP3 o dispositivi GPS. All'indirizzo web

[http://www.samsung.com/global/business/semiconductor/products/mobilesoc/Products\\_ApplicationProcessor.html](http://www.samsung.com/global/business/semiconductor/products/mobilesoc/Products_ApplicationProcessor.html)

è possibile osservare, oltre ad una panoramica sulle principali soluzioni offerte dall'azienda coreana, anche un grafico riguardante l'evoluzione dei sistemi ARM in base all'avanzamento della miniaturizzazione dei transistors.

Inoltre, all'indirizzo

<http://www.datasheetcatalog.org/datasheet2/e/0lrp9fdj0zyd6e2k2e8ej8lkzupy.pdf> è possibile reperire il *datasheet* del S3C2410A, esempio di SoC basato su ARM utilizzato in molte applicazioni, fra cui l'industria automobilistica. Alla pagina 35 del documento è possibile visualizzare il diagramma a blocchi del dispositivo, dove sono visibili tutte le componenti del sistema, fra cui possiamo notare una MMU per la gestione della memoria virtuale, due cache separate per dati e istruzioni, e i moduli periferici quali I2C e USB per le interconnessioni.

- *NXP Semiconductors*

La NXP Semiconductors è una divisione della ditta olandese Philips Electronics dedicata alle soluzioni basate su semiconduttore. La sua catena di sistemi a microprocessore fornisce numerose soluzioni SoC, molte delle quali basate su architetture ARM. Una panoramica è disponibile all'indirizzo web <http://www.standardics.nxp.com/microcontrollers/>. Ad esempio, all'indirizzo <http://www.standardics.nxp.com/products/lpc1000/datasheet/lpc1766.pdf> possiamo visionare il *datasheet* del modello LPC1766, ed in particolare alla pagina 5 del documento è visibile lo schema a blocchi, che illustra una struttura differente dall'esempio precedente, in quanto è realizzata in modo più integrato, con un'architettura basata su un bus AHB che gestisce le connessioni con uno schema a matrice multilayer. Fra i dispositivi periferici, possiamo notare i moduli CAN, interfacce seriali asincrone contraddistinte da un'elevata affidabilità (in quanto implementano un meccanismo di controllo degli errori). Inoltre, possiamo notare come i periferici siano connessi mediante due bus APB distinti, che in virtù del supporto del bus AHB alle transazioni di tipo split, consentono di incrementare il *throughput* del sistema.

- *ST Microelectronics*

La compagnia italo-francese ST Microelectronics è uno dei principali produttori al mondo di sistemi a microprocessore. Realizza soluzioni di tipo SoC per una numerosa varietà di settori, quali l'industria automobilistica, la telefonia, l'elettronica di consumo. La sua catena di microcontrollori, consultabile all'indirizzo <http://www.st.com/mcu/>, offre molte soluzioni basate su ARM. Un esempio è la famiglia di modelli STR73x, il cui *datasheet* è visionabile all'indirizzo <http://www.st.com/stonline/products/literature/ds/11651.pdf>. Anche in questo caso è disponibile il diagramma a blocchi, alla pagina 8, che evidenzia fra le altre caratteristiche un modulo di memoria di tipo Flash dedicata ai programmi, i moduli CAN già analizzati in precedenza, e un dispositivo JTAG, che come possiamo vedere non risulta collegato al bus APB. Tale modulo dispone infatti di una connessione diretta con la CPU, e viene utilizzato per effettuare debug e test di tipo hardware. Infine, ST fornisce anche dei circuiti integrati di valutazione (le cosiddette Evaluation Board) utili ai progettisti che decidono di utilizzare un sistema a microprocessore nel dispositivo che intendono realizzare. Le Evaluation Board sono circuiti semplificati, la cui produzione comporta costi di realizzazione molto inferiori a quelli di un SoC dedicato, che comunque forniscono un esempio delle funzionalità e delle caratteristiche del dispositivo. Un esempio è disponibile all'indirizzo [http://www.st.com/mcu/contentid-34-86-STR710\\_EVAL.html](http://www.st.com/mcu/contentid-34-86-STR710_EVAL.html).

## Processore ARM: Sistemi Operativi

Al momento attuale, il mercato dei sistemi operativi offre tre soluzioni principali più o meno specifiche per l'architettura ARM.

- Windows CE

La Microsoft Corp., già leader nel settore dei sistemi operativi per sistemi desktop, ha sviluppato il sistema operativo Windows CE (Compact Edition) per la sua applicazione nei sistemi embedded e dispositivi portatili. Nel 2002 un consorzio di aziende a semiconduttori ha ufficializzato la produzione di SoC ottimizzati per questo sistema operativo, come è possibile leggere nel comunicato stampa della stessa Microsoft all'indirizzo <http://www.microsoft.com/presspass/press/2002/sep02/09-18armsummitpr.mspx>. Windows CE, pur essendo un derivato della famiglia di sistemi Windows, è basato su un kernel totalmente differente. Attualmente sul mercato esistono numerose sue varianti (Windows Mobile, Microsoft PocketPC, etc.) che fanno comunque riferimento ad evoluzioni della stessa piattaforma.

- Linux

Sono state sviluppate numerose *release* di sistemi operativi open source basate su kernel Linux, specifiche per l'architettura ARM e per i sistemi embedded in generale. Un elenco di queste distribuzioni, prevalentemente basate sulla versione 2.6 del kernel, è reperibile all'indirizzo [http://www.arm.com/products/os/linux\\_download.html](http://www.arm.com/products/os/linux_download.html). E' richiesto l'utilizzo di un *boot loader* software per effettuare il caricamento del sistema operativo all'avvio. Oltre alle distribuzioni specificatamente sviluppate, esistono anche degli esempi di *porting* di distribuzioni già esistenti. All'indirizzo <http://www.debian.org/ports/arm/> sono presenti informazioni riguardanti la *porting* della distribuzione Debian su architetture ARM.

Il principale vantaggio di adottare un sistema operativo di tipo open source consiste nella possibilità di poter adattare e ottimizzare il kernel in modo estremamente personalizzato e specifico per il proprio progetto, così da avere le massime prestazioni possibili dal punto di vista del software. Risulta tuttavia evidente che non sempre questo processo di ottimizzazione è immediato da realizzare, e spesso può comportare numerose difficoltà e richiedere molto tempo,

diversamente da una versione commerciale che fornisce un'assistenza dedicata alle proprie esigenze. Comunque, su internet sono reperibili numerose guide, *HowTos*, e risposte alle domande più frequenti per la risoluzione dei problemi, il più delle volte in formato Wiki, come per esempio all'indirizzo [http://linux.onarm.com/index.php/Main\\_Page](http://linux.onarm.com/index.php/Main_Page).

- Symbian OS

Symbian Ltd., azienda in comproprietà fra i maggiori produttori di dispositivi di telefonia mobile (Nokia, Sony Ericsson, Panasonic, Siemens, Samsung) produce un sistema operativo, chiamato appunto Symbian OS, dedicato prevalentemente a dispositivi di telefonia mobile. A partire dalla versione 9.4, è stata introdotto il supporto per i processori della famiglia ARMv7. Le specifiche di questa versione del sistema operativo Symbian sono disponibili all'indirizzo <http://www.symbian.com/files/rx/file9468.pdf>.

La procedura di caricamento di un sistema operativo all'avvio di un sistema ARM è la seguente. In un banco di memoria di tipo Flash ROM viene memorizzato il boot loader, l'immagine compressa del kernel del sistema operativo e le informazioni relative al file system. Il boot loader è un programma software che ha il compito di attivare le funzioni "vitali" del dispositivo, e di decomprimere nella memoria principale l'immagine del kernel. Successivamente, sarà compito del kernel attivare il resto dei dispositivi che compongono il sistema. Il processo è illustrato in Figura 1.

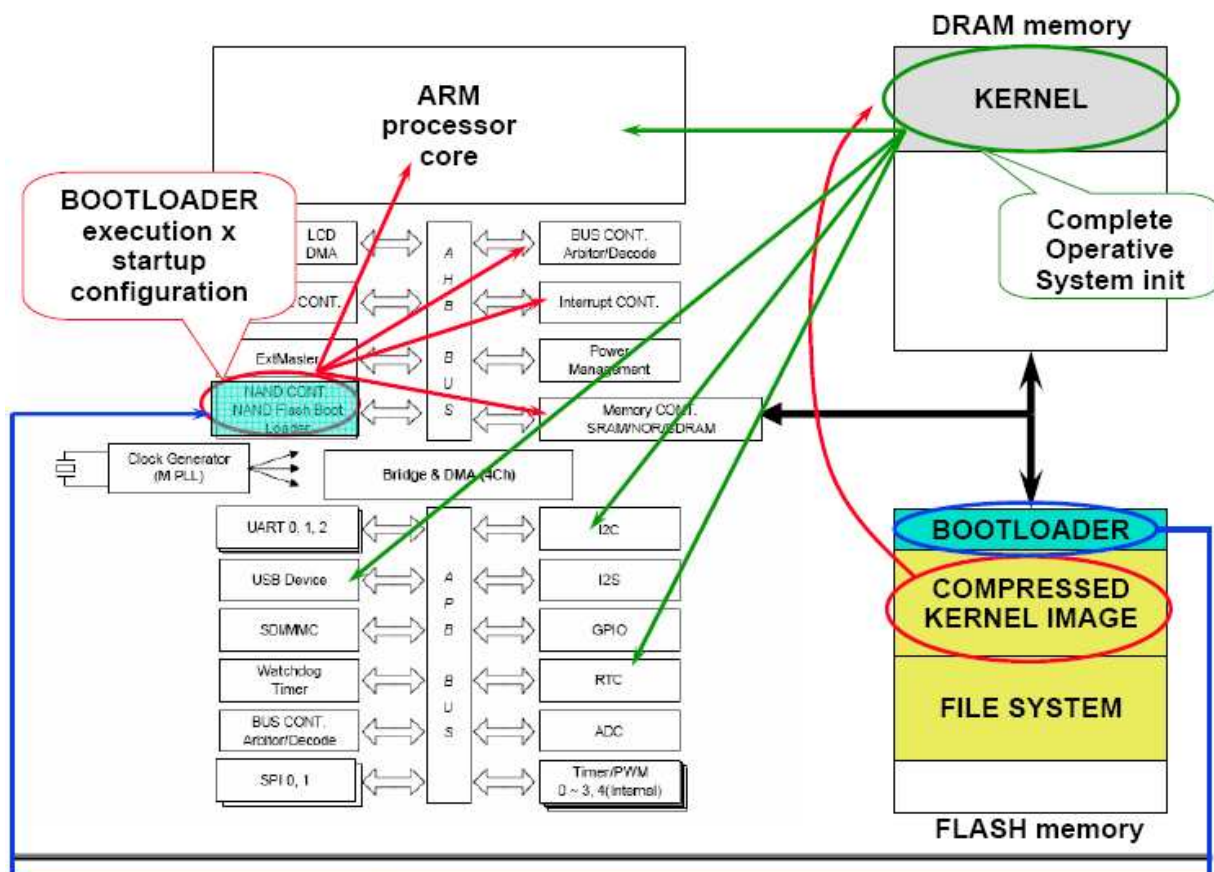


Figura 1

La procedura di caricamento dei dati sulla Flash ROM avviene tramite un'interfaccia dedicata. In base alla tipologia di sistema in considerazione, questa interfaccia può essere:

- IEEE 1149.1 Boundary Scan (JTAG) utilizzata nei System-on-Board;
- IEEE 1500 SECT utilizzata nei System-on-Chip.

Questo tipo di interfacce nasce principalmente per risolvere l'esigenza di effettuare un testing dei circuiti integrati. Tramite una catena di circuiti logici (Flip Flop), si interconnettono fra loro i dispositivi che giacciono su un circuito integrato (o su un singolo chip), come illustrato nella figura 2.

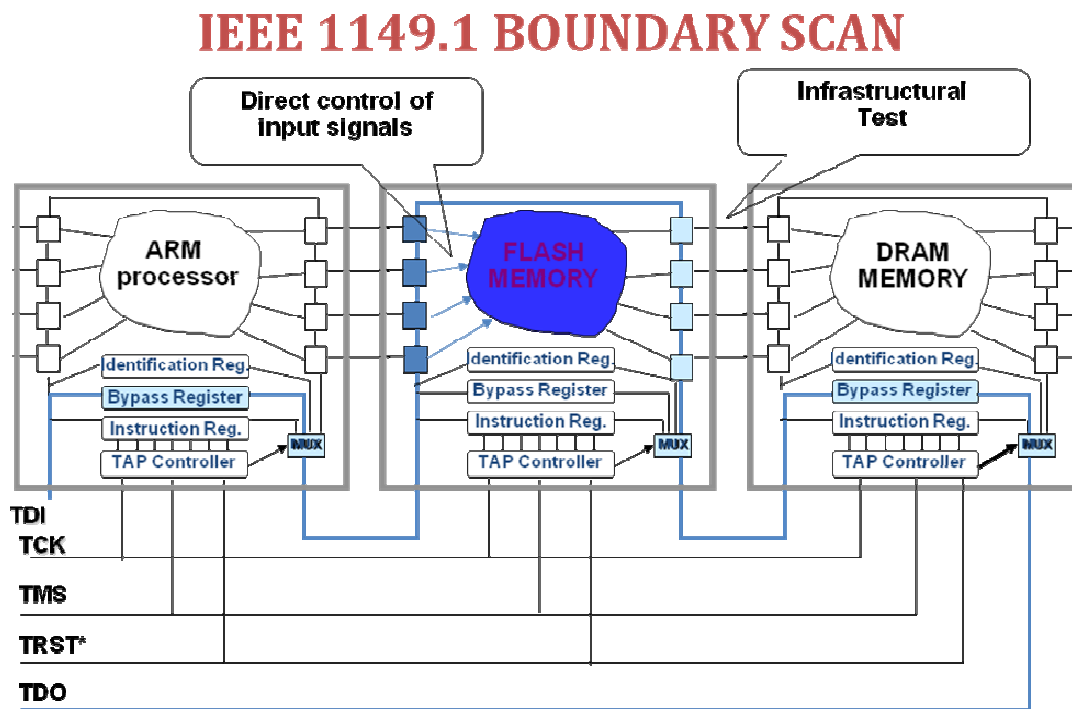


Figura 2

Attraverso i singoli pin dei dispositivi interessati, vengono trasmessi in modo seriale sincrono dei flussi di dati, che vengono successivamente analizzati in uscita per verificare il corretto funzionamento del dispositivo. I segnali che compongono l'interfaccia JTAG sono 5:

- TCK: segnale di clock del trasferimento
- TMS: segnale di selezione della modalità di test
- TRST: segnale di reset
- TDI – TDO: Test Data In/Out, connessi in modo seriale fra i dispositivi (TDO di un dispositivo connesso al TDI del successivo)

Si noti che il percorso dei dati di test attraversa l'intero "perimetro" del dispositivo, da cui il nome di Boundary Scan.

# Processore ARM: Tool Chain

Con il nome di Tool Chain facciamo riferimento ad un insieme di programmi che consentono lo sviluppo di applicazioni, a partire dal codice ad alto livello sino al linguaggio macchina. Nei sistemi ARM, la Tool Chain è costituita prevalentemente da tre tipologie di programmi:

**Cross-Compiler:** esegue la compilazione di file sorgenti scritti in linguaggi generalmente differenti (tipicamente C/C++ e ASM). In uscita genera un codice eseguibile, che può essere testato e sottoposto a debug prima di essere caricato nel sistema (generalmente operando in ambienti emulati).

**Loader:** esegue il caricamento in memoria Flash del codice eseguibile del programma, sfruttando i meccanismi visti in precedenza (JTAG, SECT).

**Hardware Debugger:** consente un interfacciamento diretto alla CPU del sistema ARM tramite dispositivi dedicati (quali l'Embedded ICE) per eseguire il debug Hardware.

Alcuni esempi di questi applicativi sono disponibili in rete. Ad esempio, all'indirizzo <http://www.keil.com> sono reperibili numerosi software a pagamento, sviluppati dalla Keil (software house facente parte del consorzio ARM) fra i quali citiamo il  $\mu$  Vision, ambiente completo di sviluppo e debug per applicativi ARM. In figura 3 è mostrato uno screenshot del programma.

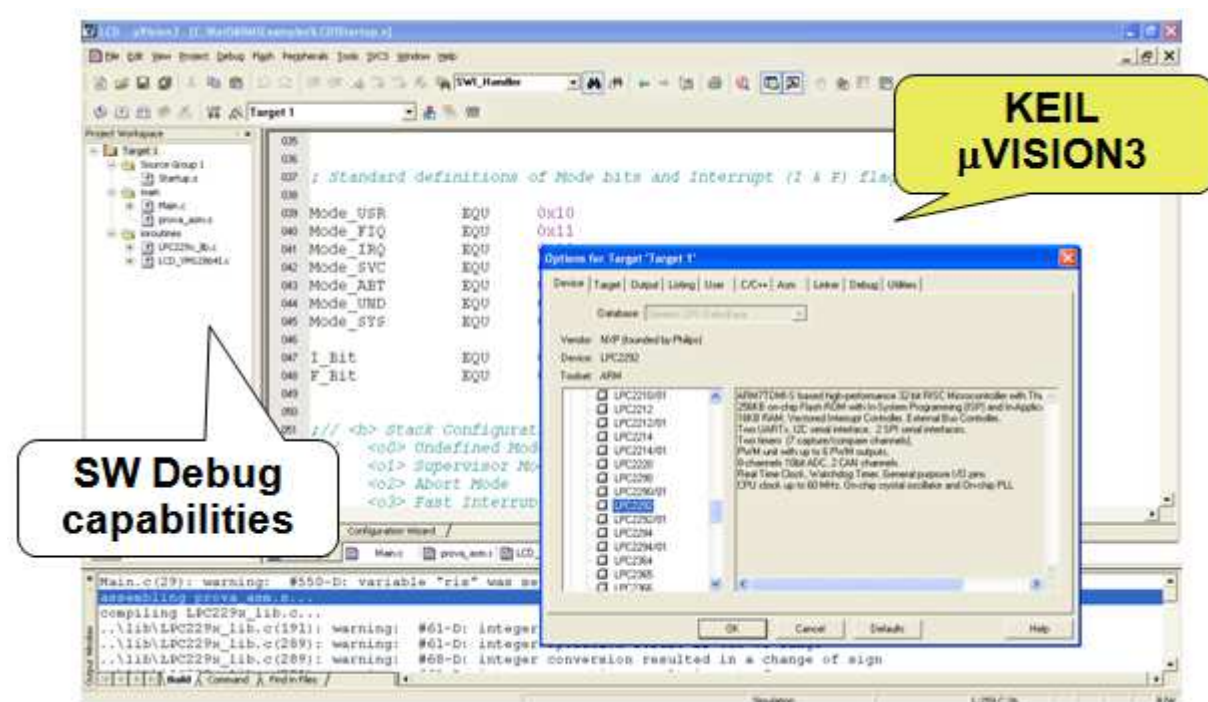


Figura 3

Per ambienti Linux, una Tool Chain open source completa è disponibile all'indirizzo [http://www.codesourcery.com/gnu\\_toolchains/arm](http://www.codesourcery.com/gnu_toolchains/arm).

Per quanto riguarda gli applicativi di Loading, esistono delle soluzioni ad-hoc realizzate per specifici prodotti, come per esempio l'utilità reperibile all'indirizzo [http://www.cad.polito.it/~sanchez/01ktm/2008-09/developmenttools/LPC2000\\_Flash\\_ISP\\_Utility\\_v2.2.3.zip](http://www.cad.polito.it/~sanchez/01ktm/2008-09/developmenttools/LPC2000_Flash_ISP_Utility_v2.2.3.zip) sviluppata per il caricamento delle memorie Flash nei dispositivi di tipo Philips LPC2000, come illustrato nella figura 4.





Figura 4

Una soluzione più versatile e personalizzabile è offerta dal software OpenWince, disponibile all'indirizzo <http://openwince.sourceforge.net/> .

Infine, per quanto riguarda i debugger hardware, la ARM mette a disposizione un Tool progettato per l'utilizzo con dispositivi di tipo Embedded ICE. Maggiori informazioni sono reperibili all'indirizzo <http://www.arm.com/products/DevTools/eXDI2RVI.html> .

## Caso di Studio: NXP LCP2292

A titolo esemplificativo, presentiamo un dispositivo di tipo System-on-Board basato su processore ARM avente le seguenti caratteristiche:

LPC 2292 Development Board prodotta da NXP

- CPU: ARM7TDMI-S a 16/32 bit
- 256 kb di memoria Flash riservata ai programmi, 16 kb di RAM interna
- 512 kb di RAM on-board
- 2 Mb di Flash On-board (External Flash, SST39VF1601)
- Frequenza di funzionamento massima: 60 Mhz
- Moduli per le interconnessioni e periferici: UARTs, CAN, I2C-bus, SPI, 32-bit Timers, connettore JTAG 20-pin standard per programmazione Flash e debugging, porta USB esterna
- Alimentazione tramite USB o alimentatore esterno 5V DC.

La scheda si presenta come in figura 5.

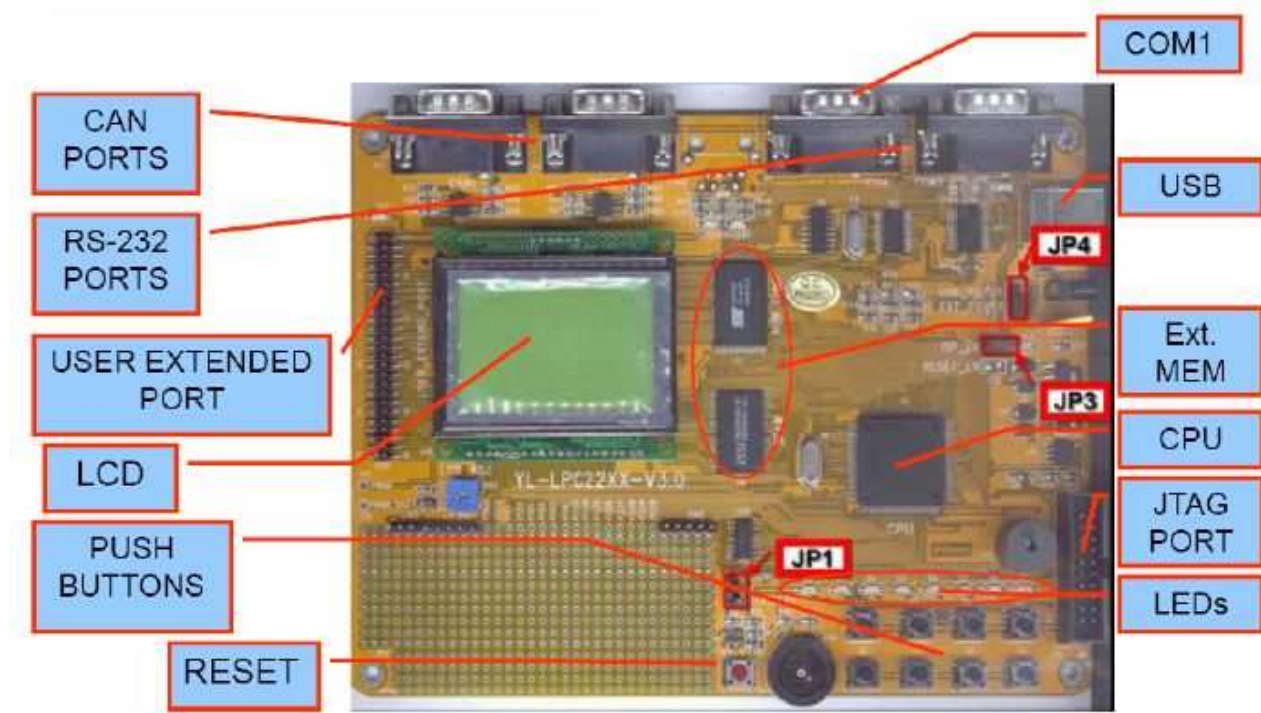


Figura 5

Sulla memoria Flash della scheda è stata caricata, mediante l'utilità LPC2000 Flash presentata nella sezione precedente, la seguente procedura in codice C:

```
1.  /*****
2.  *File :Main.c
3.  *Autor:GongJun
4.  *Date :
5.  *Modifier:cgf
6.  *Date:2005-8-5
7.  *Description:Main function
8.  *****/
9.  #include "../lib/config.h"
10. #include "../lib/board.h"
11. #define N 100
12. extern void Lcd_YM12864_Test( void );
13. extern void Lcd_YM12864_Test_clear(U32);
14. extern void Lcd_YM12864_black(U32);
15. extern void sum_int(int par1, int par2, int* ris);
16. extern void exchange_vett(char *vett_char, int n);
17. extern void avg_vett(U32 *vett_int, int n, U32* avg);
18. extern void copy_vett(char* vett_src, char* vett_dest, int n);
19. /*****
20. *Function      :void main
21. *Object        :main function
22. *Description:No
23. *Parameter     :No
24. *Return        :No
25. *****/
26. int main( void )
27. {
28.     U32 led=0xfe;
29.     int i;
30.     int par1, par2, ris;
31.     char vett[N],vett2[N];
32.     U32 vett_int32[N], avg;
33.     par1 = 10;
34.     par2 = 20;
35.     for (i=0 ; i<N ; i++)
36.     {
37.         vett[i]=N-i;
```

```

38.vett_int32[i]=i*i;
39.}
40.sum_int(par1, par2, &ris);
41.exchange_vett(vett, N);
42.avg_vett(vett_int32, N, &avg);
43.copy_vett(vett, vett2, N);
44.LedSet(led);
45.PortInit();
46.Lcd_YM12864_Test();
47.i=0;
48.while(1)
49.{
50.Lcd_YM12864_black(i%1024);
51.i++;
52.};
53.}

```

Questa procedura inizializza il display LCD della scheda richiamando il driver grafico del dispositivo. Successivamente, imposta i pixel del LCD al colore nero in modo progressivo (tramite il ciclo WHILE definito nelle linee 49-52). Inoltre, come visibile nelle linee 15-18, fa uso di quattro procedure esterne scritte in linguaggio ASM per eseguire semplici operazioni di somma e caricamento di un dato in memoria, creazione e scambio di elementi di un vettore, calcolo della media e copia del vettore stesso. Di seguito viene riportato il codice ASM delle quattro procedure.

Procedura SUM\_INT per la somma di un dato:

```

1. AREA sum_int, CODE, READONLY ; name this block of code
2. EXPORT sum_int                ; to execute
3. start
4. ADDS      r3, r0, r1           ; r2 = r0 + r1
5. MOVEQ     r3, r0, LSL #2       ;if r3=0
6. STR       r3,[r2]             ;store r3 value
7. stop
8. MOV       PC, LR
9. END                            ; Mark end of file

```

Procedura EXCHANGE\_VETT per lo scambio di elementi di un vettore:

```

1. AREA exchange_vett, CODE, READONLY ; name this block of code
2. EXPORT exchange_vett              ; to execute
3. start
4. MOV       r5, #0x0
5. new       LDRB  r3,[r0,r5]         ;load in R3 [r0] - post indexed
6. LDRB      r4,[r0,r1]               ;load in R4 [r0+n]
7. STRB      r4,[r0,r5]
8. STRB      r3,[r0,r1]
9. SUB       r1,r1,#1
10. ADD      r5,r5,#1
11. CMP      r5,r1
12. BNE      new
13. stop
14. MOV      PC, LR                  ; Terminate
15. END                            ; Mark end of file

```



### Procedura AVG\_VETT per il calcolo della media degli elementi di un vettore:

```
1. AREA avg_vett, CODE, READONLY      ; name this block of code
2. EXPORT avg_vett                    ; to execute
3. start
4. MOV      r4, #0x0
5. MOV      r5, #0x0
6. new      LDR      r3, [r0,r5,ls1 #2] ;
7. ADD      r4, r4, r3                ; r4 = r4 + r3
8. ADD      r5, #1
9. CMP      r5, r1
10. BNE     new
11. str      r4,[r2]
12. stop
13. MOV      PC, LR                  ; Terminate
14. END                                           ; Mark end of file
```

### Procedura COPY\_VETT per la copia di un vettore:

```
1. AREA copy_vett, CODE, READONLY      ; name this block of code
2. EXPORT copy_vett                    ; to execute
3. start
4. ADD      r3, r0, #100
5. new
6. LDMIA    r0!, {r4,r13}              ; 10*4 bytes at time
7. LDMIA    r1!, {r4,r13}              ; 10*4 bytes at time
8. CMP      r0, r3
9. BNE     new
10. SWI      #0x10
11. stop
12. MOV PC, LR                        ; Terminate
13. END                                           ; Mark end of file
```

Si noti che la procedura COPY\_VETT fa uso del Software Interrupt (linea 10).