www.asap-firmware.com

Prefazione

Sistema Operativo uCLinux
Coldfire MCF5272
Queued Serial Peripheral Interface (QSPI)
uCLinux su M5272C3
Software applicativo e debug

Titolo: FWS0006A

uCLinux E SPI SU M5272C3

Sistema Operativo uCLinux

La storia

Molte cose sono state scritte e sono reperibili sulla storia di Linux, una stroria che è diventata quasi un classico nella storia dell'informatica, paragonabile e allo stesso tempo in contrapposizione all'altro evento per antonomasia e cioè alla nascita del DOS per opera di Bill Gates.

Il kernel di Linux è stato originariamente sviluppato da Linus Torvalds (foto a sinistra).

Il progetto iniziale era ispirato a Minix, un piccola versione di UNIX sulla quale Torvalds iniziò a lavorare nell'agosto del 1991, ottenendo come risultato la prima versione 0.0.1

Nell'ottobre del 1991 arrivò alla creazione della versione 0.0.2 la prima versione che si può definire "ufficiale", che Linus stesso annunciò nel gruppo di discussione *comp.os.minix* e comprendeva la shell dei comandi bash e GNU compiler; era il primo e importante passo e una serie di sviluppatori abbracciò il progetto.

Dopo la versione 0.0.3 ci fu un rapido susseguirsi di nuove versioni grazie anche all'appoggio degli sviluppatori divenuti ormai una vera e propria comunità telematica.
La versione 1.0, la prima e essere dichiarata stabile, venne rilasciata nel dicembre 1993 e poteva "funzionare" su processori di classe i386.

Nel 1997, tre anni dopo il rilascio della prima versione, si stima che Linux sia utilizzato da circa 3 milioni di persone, cosa difficile da stabilire con esattezza in quanto non esiste una licenza per gli utenti individuali.

Alla fine del 1998 la stima è raddoppiata a 7 milioni, ed il suo tasso di crescita è più veloce di qualsiasi altro sistema operativo.

Linux viene spesso rappresentato con un pinguino, divenuto la "mascotte" del sistema operativo e che ha preso il nome di Tux (interpretabile come l'abbreviazione di **T**orvalds**U**ni**X**).

Linux e GNU

GNU/Linux è una libera e distribuibile versione di UNIX sviluppata da Linus Torvalds presso l'Università di Helsinki in Finlandia. GNU/Linux è un "crogiolo" di conoscenze ed esperienze che hanno trovato in quest'ultimo una piazza virtuale dove crescere rapidamente.

All'interno del sistema operativo non viene utilizzato in nessun modo codice licenziato da enti commerciali e buona parte del software che ruota attorno segue questa "corrente" di pensiero abbracciata dal Progetto GNU della Free Software Foundation.

Linux infatti rappresenta unicamente il kernel mentre GNU/Linux sarebbe la giusta formulazione per indicare il kernel di Linux e tutto il software di gestione offerto dal progetto GNU che rende un sistema operativo utilizzabile.

uCLinux

uClinux è un derivato del kernel Linux 2.0 pensato per microcontrollori senza Memory Management Units (MMU).

Inoltre gran parte dei binari e del codice sorgente per il kernel sono stati riscritti per ridurre il codice di base; questo significa che il kernel uClinux è molto più ridotto del kernel 2.0 originale, pur mantenendo i principali vantaggi del sistema operativo come stabilità, funzionalità di networking avanzato, e ampio supporto di file systems.

Caratteristiche :

- Common Linux API
- uCkernel < 512 kb
- uCkernel + tools < 900 kb

Principali limitazioni rispetto a Linux :

- il supporto multitasking è garantito ma si utilizza *vfork()* invece di *fork()* per cui il processo *padre* è bloccato fino a che il *figlio* non esegue una shell con *exec()* o termina con *exit()*
- Lo stack non incrementa automaticamente ma viene dimensionato alla compilazione con *mmap*
- la protezione di memoria tra task diverse non è garantita a causa dell'assenza della MMU
- non è possibile eseguire lo *swap* della memoria, con conseguente aumento della memoria totale necessaria per gli eseguire gli applicativi

La distribuzione uCLinux comprende il kernel 2.0.x con funzionalità ridotte ma di dimensioni contenute, il kernel 2.4.x che comprende funzionalità aggiuntive come supporto USB e IPV6, e il più recente kernel 2.6.x.

Inoltre viene supportata una vasta serie di schede prodotte da altrettanti fornitori, tra le quali l'utente dovrà selezionare quella in dotazione prima della compilazione del kernel.

Nel caso il proprio Sistema non sia ancora supportato da uCLinux, bisogna considerare tre livelli principali di porting :

1) ARCHITETTURA

La condizione più complicata si verifica quando il processore non è ancora supportato o differisce troppo dalle architetture supportate

In questo caso è necessario creare una nuovo ramo *linux/arch* e creare o adattare da un'altra architettura circa una ventina di files relativi alle istruzioni specifiche del core CPU

2) PIATTAFORMA

Questo in realtà è rilevante solo per l'architettura m68k no-MMU, in cui esistono molte varianti del core aventi differenze nella modalità supervisor/user e Floating Point Unit (FPU).

In questa situazione abbiamo un processore già supportato in *linux/arch* ma ha differenze che impattano nel kernel come varianti del core o periferiche on-chip.

In questo caso è necessario creare una nuovo ramo *linux/arch/platform* e creare circa sei files per i traps (ingressi al kernel da programmi utente), gestione degli interrupt e relativi vettori.

3) SCHEDA

In questo caso l'architettura assomiglia di più a una piattaforma esistente.

Tuttavia esistono delle lievi differenze che interessano il kernel come la quantità o il tipo di memoria e le periferiche esterne.

In questo caso è necessario creare una nuovo ramo *linux/arch/platform/cpu/* e creare alcuni files come *rom.ld* e *crt0_rom.S*

Coldfire MCF5272

Di seguito sono elencate le principali caratteristiche di questa CPU :

- Processore Versione 2 Codlfire RISC
  1. bus indirizzi e dati a 32bit
  2. frequenza core e bus 66MHz
  3. 16 registri a 32bit
  4. MAC unit
- Memoria on-chip
  1. 4Kb SRAM
  2. 16Kb ROM
  3. 1Kb cache istruzioni
- Due UART asincrone/sincrone
  1. Operatività full-duplex
  2. Possibilità segnali controllo modem
  3. FIFO Tx e Rx di 24bytes ognuna
- Modulo ethernet
  1. 10 base T, half o full-duplex
  2. 100 baseT half duplex e limitata in full
  3. MII
- USB
  1. 12 MBps
  2. compatibile con specifiche USB 1.1
- Interfaccia memoria esterna
  1. 8,16,32 bit bus per SRAM e SROM
  2. supporto SDRAM
  3. wait-state programmabile
- Queued Serial Peripheral Interface (QSPI)
  1. full-duplex, three wire, sincrona
  2. fino a 4 chisp select diretti e 15 con decoder esterno
  3. master operation
- Timer
  1. 4x16bit timer con capture e compare per timer1 e timer2
  2. 15ns di risoluzione a 66MHz
  3. watchdog timer
- PWM Unit con 3 canali

Queued Serial Peripheral Interface (QSPI)

La periferica QSPI come si evince dal nome si basa sullo standard SPI ma permette anche di eseguire dei trasferimenti in coda.

Infatti è possibile accodare fino a 16 trasferimenti alla volta eliminando l'intervento dell'utente tra l'uno e l'altro.

Le principali caratteristiche sono le seguenti :

- lunghezza dato configurabile da 8 a 16 bits con incremento di 1bit
- quattro chip select diretti con cui è possibile controllare 15 dispositivi con un decoder esterno
- velocità del clock da 129 kbps fino a 33Mbps a 66MHz
- ritardo programmabile tra un trasferimento e l'altro
- polarità e pase del clock programmabile

- supporto modalità wraparound per trasferimenti continui

uCLinux su M5272C3

In dotazione abbiamo una evaluation board Motorola M5273C3 Rev 1.2 (ora Freescale) equipaggiata con Coldfire MCF5272.

Lo scopo dell'applicativo è dimostrare come sia disponibile utilizzare la periferica QSPI con OS uCLinux, per poter interfacciarsi con altre periferiche che utilizzano lo standard SPI; ad esempio sulla base della nostra esperienza su CAN Bus possiamo pensare di comunicare con un'altra CPU (es. DSP56F803) con periferica CAN in cui viene gestito in real-time un protocollo standard come il CANOpen, oppure gestire una controller CAN Bus esterno come Infinenon 82C900 e Intel 82527 (od altri che dispongono di SPI).

Nel primo caso i vincoli imposti dalle esigenze di real-time sono assolte dalla CPU esterna; utilizzando il canale di comunicazione è possibile creare un protocollo con il quale il master (MCF5272) interroga lo slave (DSP56F803) per gestire gli oggetti CAN Bus attraverso una sorta di mappa condivisa.

Nel secondo caso la gestione è affidata alla CPU con OS uCLinux, il quale non essendo preemptive consente nelle ultime versioni un "soft" real-time con tempi di latenza sugli interrupt dell'ordine di qualche millisecondo.

Questo potrebbe risultare un collo di bottiglia nella gestione in ricezione degli oggetti CAN Bus, specialmente a velocità di bus superiori a 125kbps.

Esistono comunque delle patch per ottenere un "hard" real-time come quello denominato RTAI che è stato sviluppato dal Politecnico di Milano (www.aero.polimi.it); affronteremo e analizzeremo prossimamente lo sviluppo di questa soluzione.

Installazione

Il kernel uCLinux viene fornito con l'aggiunta di una serie di applicativi, per formare quella che comunemente viene definita una *distribuzione*; essendo Open Source questa è liberamente scaricabile via internet da www.uclinux.org/pub/uCLinux

In questo caso è stata ultilizzata la *uClinux-dist-20040408.tar.gz* e la tools chain *m68k-elf-tools-20030314.sh.*

L'installazione della tools chain avviene automaticamente digitando "*sh m68k-elf-tools-20030314*".

A questo punto occorre creare un directory dove installare uCLinux, copiare il file .tar.gz ed estarre il suo contenuto con "*tar xzvf uClinux-dift-20040408.tar.gz*"; viene creata un directory *uClinux-dist* dalla quale eseguiremo "*make menuconfig*" per configurare il kernel.

All'apertura del menu selezionare *Vendor/Product Selection*, quindi Motorola su *Vendor* e M52C72C3 su *Product* (vediamo che sono supportate altre board come M5282C3, M5206eC3, M5407C3, M5249C3 ecc.).

Uscire e passare quindi a *Kernel/Library/Defaults Selections* e selezionare linux-2.4.x kernel version e uC-libc Libc version e Customize kernel Settings; selezionare Exit fino ad uscire confermando il salvataggio.

Si riaprirà il menu di configurazione del kernel; su *Character devices* selezionare Coldfire QSPI Support e uscire salvando.

A questo punto eseguire "*make dep*" per le dipendenze e poi "*make*" per la compilazione; se tutto va a buon fine otterremo il file "*uClinux-dist/images/image.bin*" che verrà copiato anche in */tftpboot*, la directory di default del server TFTP che dovrebbe essere già installato nel nostro PC Linux.

Passiamo ora al collegamento del PC Linux alla seriale TERMINAL sul connettore DB9 P3, a Ethernet sul connettore RJ45 JS1, ed dell'alimentatore su P1 o P2 (es. con D.C. 5V - 1A).

Nel PC Linux lanciare un emulatore di terminale come "minicom" impostando 19200bps-N-8-1, e all'accensione della scheda apparirà il prompt "*dBug>*" dell'omonimo rom-monitor che fungerà da terminale interfaccia utente via seriale.

Impostare "*set filename image*" e "*set filetype image*" per informare il monitor che il file da trasferire è *image.bin*; digitando semplicemente "*dBug>dn*" inizierà il trasferimento via TFTP; alla fine digitare "*go 0x20000*" e apparirà il logout del nostro uCLinux.

Software applicativo e debug

Per inserire nel progetto il software applicativo creare la directory *uClinux-dist/user/qspi* ed estrarre al suo interno il contenuto del file *qspi.tar.gz* associato a questa soluzione.

Aprire il file *uClinux-dist/user/Makefile* e inserire la seguente linea :
dir_$(CONFIG_USER_QSPI_QSPI) += qspi
per aggiungere la directory "qspi" alla lista delle directory del progetto

Aprire il file *uClinux-dist/config/Configure.help* e inserire le seguenti linee :
CONFIG_USER_QSPI_QSPI
<spazio><spazio>Use QSPI to peripheral interfacing
per aggiungere una stringa di max 70 caratteri indentata di due spazi per l'help in linea relativo

Aprire il file *uClinux-dist/config/config.in* e inserire la seguente linea ad esempio nel menu Miscellaneous Applications:
comment 'QSPI interface'
bool 'qspi'<TAB><TAB><TAB>CONFIG_USER_QSPI_QSPI
per inserire nel relativo menu di configurazione (menuconfig) l'inserimento del progetto utente QSPI.

Per comprendere il nuovo progetto software qspi con "*make menuconfig*" passare al menu Miscellaneous Applications e abilitare la selezione *qspi*.

Infine da *uClinux-dist/ eseguire "make dep"* e poi *"make"* e alla fine dovremo trovare il file *uClinux-dist/romfs/bin/qspi*; dalla scheda eseguire il download via TFTP con "*dBug>dn*".

Verificare che l'interfaccia eth0 sia configurata con un indirizzo IP digitando "*dBug>ifconfig*" e leggendo il risultato; se non lo fosse impostare un indirizzo a piacimento ad esempio "*dBug>ifconfig 192.168.1.8*" e lanciare uClinux con "*go 0x20000*".

Digitando "*ls /bin*" dovremo trovare il nostro applicativo sotto forma del file "*qspi*".

Debug

Per eseguire il debug si utilizzano due programmi uno per il PC e l'altro per la scheda.

Dalla scheda digitare "*dBug>gdbserver :8000 /bin/qspi*" che attiva l'applicativo gdb come server utilizzando ethernet sulla porta 8000 (o un'altra sicuramente non utilizzata). Dal PC da *uClinux-dist/* digitare "*m68k-elf-gdb ./user/qspi/qspi.gdb*" che attiva l'applicativo gdb come client con il file *qspi.gdb*.

Dal prompt di (gdb) collegarsi alla scheda con "*(gdb) target remote 192.168.1.8:8000*"

A connessione avvenuta dovremo essere nel main: verifichiamolo digitando "*list*" e leggendo a video le righe di codice relative.

Da questo punto in poi sono validi tutti i comandi in linea relativi a gdb e visibili digitando "*help*"; possiamo ad esempio mettere un breakpoint nel main con "*(gdb)b main*", iniziare l'esecuzione con "*(gdb) cont*", andare avanti step-by-step con "*(gdb)n*" oppure visualizzare il contenuto delle variabili con "*(gdb)p <nome variabile>*".

I segnali relativi sono disponibili sul connettore J6 come segue:
connettore J6 – pin B35 – segnale QSPI_DOUT / WSEL (Data Output)
connettore J6 – pin B37 – segnale QSPI_CLK / BUSW1 (Clock)
jumper JP22-3 / connettore J6 – pin A38 – segnale QSPI_CS0 / BUSW0 (Chip Select)

La soluzione ASAP

Il semplice applicativo realizzato è a puro scopo dimostrativo e può essere riassunto con i seguenti punti:

1. viene eseguita l'apertura del dispositivo visto come un file
2. per mezzo della chiamata di sistema *ioctl* si impostano i parametri di funzionamento
3. vengono immessi i valori dei dati da trasmettere
4. vengono trasmessi i dati

Nel main-loop troviamo il seguente codice :

```
if ((SpiPort=open("/dev/qspi1", O_RDWR)) < 0)
```

apertura del dispositivo *qspi1* in modalità lettura-scrittura

```
InpValue= GetLongValue("Input DOUT type[0=PushPull,1=HiZ]:\n");
if (ioctl(SpiPort, QSPIIOCS_DOUT_HIZ,InpValue))
```

impostazione del pin DataOut in push-pull o alta impedenza

```
InpValue= GetLongValue("Input DOUT bits N.[8<=N<=16]:\n");
if (ioctl(SpiPort, QSPIIOCS_BITS,InpValue))
```

impostazione del numero di bits del dato

```
InpValue= GetLongValue("Input Clock Polarity[0-1]:\n");
if (ioctl(SpiPort, QSPIIOCS_CPOL,InpValue))
```

impostazione livello inattivo del clock con 0 per basso e 1 per alto

```
InpValue= GetLongValue("Input Clock Phase[0-1]:\n");
if (ioctl(SpiPort, QSPIIOCS_CPHA,InpValue))
```

0 per validare il dato sul livello attivo del clock e cambiare sul fronte successivo
1 per cambiare il dato sul livello attivo del clock e validare sul fronte successivo

```
InpValue= GetLongValue("Input Clock Value[130<kbps<16500]:\n");
if (ioctl(SpiPort,QSPIIOCS_BAUD,(uint32_t)((66000.0/(2.0*InpValue))+0.5)) )
```

impostazione del baudrate; il valore immesso viene usato per calcolare con la seguente formula il valore da passare al registro QMR[BAUD] della CPU :

QMR[BAUD] = Clock CPU / (2 * baudrate desiderato)

Per ogni ulteriore informazione

è raggiungibile al seguente indirizzo di posta elettronica : info@asap-firmware.com