Programmazione dei microcontollori AVR con GCC, libc 1.0.4

Guido Socher (homepage)

L'autore:

A Guido piace linux perché è veramente un buon sistema per sviluppare il proprio hardware.

Tradotto in Italiano da:
Gianpaolo Demarchi (homepage)

Premessa:

I microcontrollori AVR 8−bit RISC sono un tipo di microcontrollori molto comune. Essi contengono in un solo chip EEPROM, Ram, convertitori da analogico a digitale, molte linee di input e output, timer, UART per comunicazione via RS232 e altro ancora.

La caratteristica peculiare di questi microcontrollori è l'esistenza di un completo sistema di sviluppo sotto Linux: è possibile programmarli infatti in C con il GCC.

Ho già scritto nel Marzo 2002 un articolo con lo stesso soggetto. Da allora però sono cambiate un sacco di cose sia nello sviluppo della libreria avr−libc, sia per il fatto che il microcontrollore AT90S4433 usato in quella serie di articoli non viene più prodotta da Atmel.

Questo articolo quindi è un aggiornamento dell'articolo del Marzo 2002. In questo articolo verranno quindi utilizzati le libc−1.0.4. e il microcontrollore ATmega8.

Introduzione

Molte persone si sono interessate alla programmazione dei microcontrollori dopo l'articolo che scrissi nel 2002. Lo scoglio maggiore è la realizzazone di un sistema di sviluppo funzionante. Se qualcosa non funziona infatti non si hanno in generale indizi sulle cause del malfunzionamento. Il cavo programmatore sbagliato? Malfunzionamenti nel circuito? Installazione errata? La porta parallela disabilitata nel BIOS? Il modulo del kernel ppdev compilato in maniera errata? Le ragioni per cui le cose non funzionano possono essere molteplici.

Per facilitare l'ingresso nel mondo eccitante dei microcontrollori shop.tuxgraphics.org offre ora un CD avviabile con un manuale e l'hardware necessario per la programmazione. Tutto quello che si deve fare è quindi avviare il computer dal CD e tutto sarà pronto e funzionante.. Non è necessario installare nessun software aggiuntivo e non viene modificato nulla sul computer.

Anchi'io ogni tanto uso questo CD in quanto spesso l'hardware che realizzo spesso sopravvive a molte generazioni di kernel e versioni di software installate sul PC. Se ad un certo punto voglio modificare il software del microcontrollore, usando il CD non devo preoccuparmi di se l'ambiente di sviluppo è ancora funzionante sul mio PC Linux. Faccio semplicemente partire la macchina da CD e il sistema è pronto e funzionante.

Indipendentemente dal CD, descriverò nei paragrafi seguenti l'installazione del sistema di sviluppo GCC per gli AVR. Se avete preso il CD su tuxgraphics potete saltare la parte seguente ed andare direttamente al paragrafo "Un piccolo progetto di prova".

Installazione del software: di cosa c'è bisogno

Per utilizzare l'ambiente di sviluppo GNU C c'è bisogno del software seguente:

binutils−2.15.tar.bz2	Disponibile su: ftp://ftp.gnu.org/gnu/binutils/ o su altri mirror. Es: ftp://gatekeeper.dec.com/pub/GNU/binutils/
gcc−core−3.4.2.tar.bz2	Disponibile su: ftp://ftp.gnu.org/gnu/gcc/ o su altri mirror. Es: ftp://gatekeeper.dec.com/pub/GNU/gcc/
avr−libc−1.0.4.tar.bz2	La libreria C per gli AVR: http://savannah.nongnu.org/projects/avr−libc/
uisp−20040311.tar.bz2	Il software per programmare gli AVR: http://savannah.nongnu.org/projects/uisp

Installeremo tutti i programmi in /usr/local/avr. Questa scelta viene fatta per mantenere separati questi programmi dal compilatore C standard di Linux. Creiamo la directory in questa maniera:

```
mkdir /usr/local/avr
```

e aggiungiamo la directory dei programmi al PATH:

```
mkdir /usr/local/avr/bin
export PATH=/usr/local/avr/bin:${PATH}
```

Installazione del software: GNU binutils

Il pacchetto binutils fornisce tutte le utilities a basso livello necessarie per la creazione dei file oggetto. Il pacchetto comprende un assembler AVR (avr−as), un linker (avr−ld), strumenti per l'uso di librerie (avr−ranlib, avr−ar), un programma per generare file oggetto da caricare nella EEPROM del microcontrollore (avr−objcopy), un disassembler (avr−objdump) e altre utility quali avr−strip e avr−size.

Per compilare e installare le binutils è sufficiente eseguire:

```
tar jxvf binutils-2.15.tar.bz2
cd binutils-2.15/
mkdir obj-avr
cd obj-avr
../configure --target=avr --prefix=/usr/local/avr --disable-nls
make

# come root:
make install
```

Per aggiornare la cache del linker è necessario aggiungere la riga /usr/local/avr/lib al file /etc/ld.so.conf e lanciare il comando /sbin/ldconfig

Installazione del software: AVR GCC

avr−gcc sarà il nostro compilatore C.

Per compilarlo e installarlo è sufficiente eseguire:

```
tar jxvf gcc-core-3.4.2.tar.bz2
cd gcc-3.4.2

mkdir obj-avr
cd obj-avr
../configure --target=avr --prefix=/usr/local/avr --disable-nls --enable-language=c

make

# come root:
make install
```

Installazione del software: la libreria avr−libc

La libreria avr−libc è molto più stabile ora rispetto a quanto non lo fosse nel Marzo 2002.
Per compilarla ed installarla è sufficiente eseguire:

```
tar jxvf avr-libc-1.0.4.tar.bz2
cd avr-libc-1.0.4
PREFIX=/usr/local/avr
export PREFIX
sh -x ./doconf
./domake
```

```
cd build
#come root:
make install
```

Installazione del software: il Programmatore

Il software programmatore carica il codice oggetto specificatamente preparato per il microcontrollore nella sua memoria EEPROM.

Il programma uisp è un programmatore molto potente. Può essere utilizzato direttamente all'interno di un Makefile. E' sufficiente infatti aggiungere la regola "make load" per compilare e caricare il codice in un unico passaggio.

Per installare uisp basta eseguire:

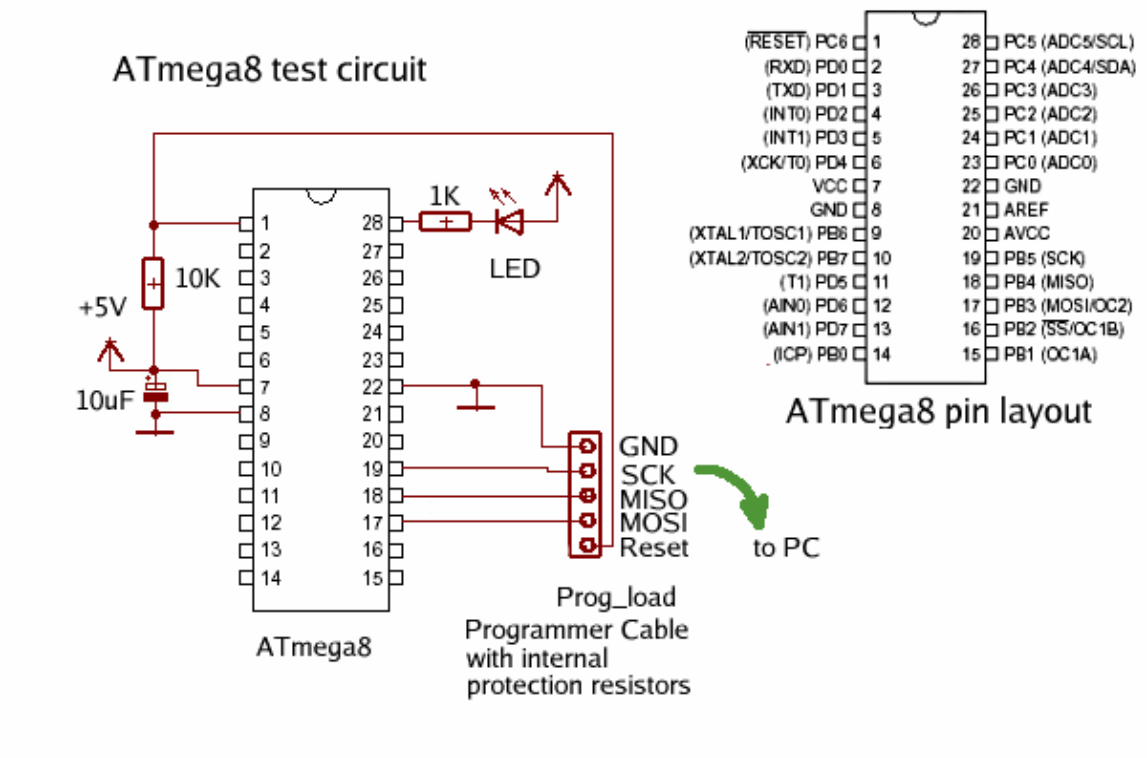
```
tar jxvf uisp-20040311.tar.bz2.tar
cd uisp-20040311
./configure --prefix=/usr/local/avr
make

# come root:
make install
```

Un piccolo progetto di prova

Cominceremo con un piccolo circuito di prova, che potrà essere migliorato con l'aggiunta di altre parti in futuro.

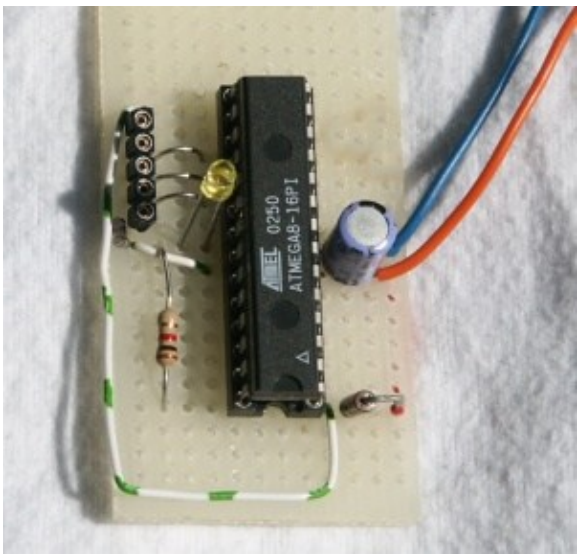
Tale circuito può venire anche usato come banco di prova per progetti hardware più complessi. E' possibile infatti provare il software caricato e aggiungere sensori e sistemi di misura.

Il programma di prova presentato qui di seguito serve per far lampeggiare un LED.



Hardware necessario

Per costruire il circuito sono necessari i componenti elettronici elencati nella tabella successiva. Benchè gli AVR siano microcontrollori molto comuni, può essere che non siano disponibili nel vostro negozio di elettronica di fiducia, ma i grossi distributori (www.conrad.de (Germania), digikey.com (US), RS o Distrelec (Italia)) sicuramente ne hanno in magazzino. E' possibile ottenere il kit completo o solo il microcontrollore anche sul sito shop.tuxgraphics.org



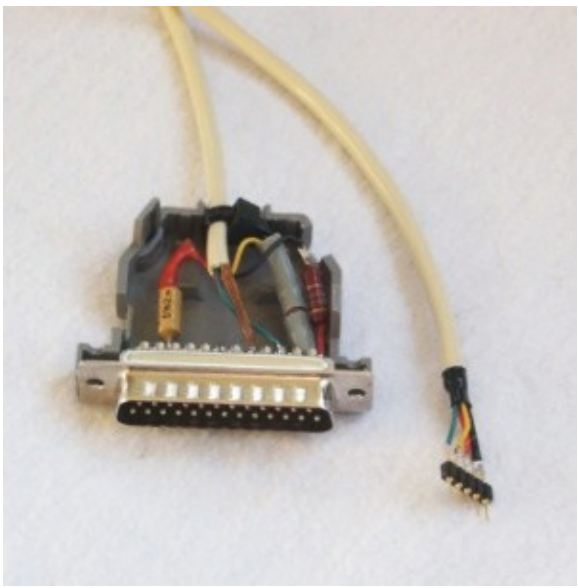
1 x ATmega8 in versione DIP, Atmel 8 bit Avr processore risc.
1 x 28 pin 7.5mm zoccolo per IC Lo zoccolo da 28 pi è spesso difficile da trovare. Tipicamente gli zoccoli da 28 sono larghi 14mm ma noi ne abbiamo bisogno di uno da 7.5 ..
1 x 10K resistenze (codice colore: marrone, nero, arancione) 1 x 1K resistenza (codice colore: marrone, nero, rosso) 1 x 10uF condensatore elettrolitico Alcuni fili 1 x LED

Basetta millefori
Il materiale seguente è necessario per il programmatore (non serve se si acquista il "Linux AVR programming kit" da tuxgraphics): 1 x DB25 connettore da inserire nella porta parallela. Un qualsiasi connettore/zoccolo da 5 pin per il programmatore. Raccomando di usare "precision strip connectors" (simili agli zoccoli per IC) e staccare 5 piedini. 1 x 220 Ohm resistenza(codice colore: rosso, rosso, marrone) 2 x 470 Ohm resistenza (codice colore: giallo, viola, marrone)

In aggiunta ai componenti elencati sopra bisogna procurarsi un alimentatore DC stabilizzato a 5 V oppure è possibile utilizzare per l'alimentazione del circuito una batteria da 4,5 V

Avrete probabilmente notato che non vi è alcuna necessità di un cristallo esterno. Questo perchè l'ATmega8 ha un oscillatore incorporato. Tale oscillatore può tranquillamente venir usato nel caso in cui non sia necessaria una tempistica accurata. Nel caso però in cui si voglia realizzare un'apparecchiatura di misura abbastanza accurata oppure si voglia utilizzare l'interfaccia UART/RS232 allora è necessario montare un oscillatore a cristallo. Il tipo di oscillatore utilizzato viene definito attraverso dei "fuse bits" che possono essere modificati dal programmatore. Di default è attivato l'oscillatore interno a 1 MHz.

Costruire l'hardware del programmatore



I microcontrollori AVR possono essere programmati via ISP (In System Programing).
Ciò vuol dire che non è necessario rimuovere in microcontrollore dal circuito per programmarlo. E' possibile procurarsi vari programmatori hardware per 50−150 Euro. Utilizzando Linux è possibile realizzare un programmatore molto semplice che fa' al caso nostro. Basta avere sul computer una porta parallela libera e cavo descritto nelle righe seguenti.

Da notare che il cavo programmatore qui presentato rappresenta una versione migliore rispetto a quella presentata nel Marzo 2002. In questo nuovo cavo le resistenze di protezione vengono montate direttamente nel connettore, permettendo anche di risparmiare un po' di spazio nel scheda del circuito. I collegamenti per il cavo programmatore devono essere eseguiti nella maniera seguente:

pin sul pcb	pin sul AVR	resistenza di protezione	Pin sulla porta parallela
5	Reset (1)	−−	Init (16)
4	MOSI (17)	470 Ohm	D0 (2)
3	MISO (18)	220 Ohm	Busy (11)
2	SCK (19)	470 Ohm	Strobe (1)
1	GND	−−	GND (18)

Il cavo inoltre non deve essere più lungo di 70 cm.

Da notare nella figura come le resistenze di protezione possano essere montate direttamente nel connettore.

Scrivere il software

L'ATmega8 può essere programmato in C puro con l'aiuto del gcc. La conoscenza dell'assembler AVR può risultare utile ma non è necessaria.

La libreria AVR libc ha il suo avr−libc−user−manual−1.0.4.pdf (1139921 bytes) che descrive tutte le funzioni disponibili in C. Dal sito di Atmel, (www.atmel.com, vai a : avr products −> 8 bit risc−> Datasheets),è possibile inoltre scaricare il datasheet completo, che descrive tutti i registri e spiega come usare la CPU.

Una cosa da ricordare quando si usa un microcontrollore è che esso è dotato di pochi Byte di RAM. Ciò implica che non si possono dichiarare strutture di dati complesse o stringhe molto lunghe. Il programma inoltre non deve contenere chiamate a funzioni "annidate" (NdT: in inglese deeply nested) o funzioni ricorsive.

Ma la pratica conta più della grammatica. Scriveremo ora un programma che farà lampeggiare un LED a intervalli di 0.5 secondi. Non è un programma utilissimo, ma è un buon punto da cui partire.

La libreria avr−libc è cambiata molto. In origine si settava un bit su una porta chiamando sbi() e si resettava con cbi(). Ora l'uso di tali funzioni è deprecato. Nella "vecchia maniera" si sarebbe scritto:

```
/* defines for future compatibility */
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

void main(void)
{
      /* INITIALIZE */
      /* enable PC5 as output */
      sbi(DDRC,PC5);


      /* BLINK, BLINK ... */
      while (1) {
            /* led on, pin=0 */
            cbi(PORTC,PC5);
            delay_ms(500);
            /* set output to 5V, LED off */
            sbi(PORTC,PC5);
            delay_ms(500);
      }
}
```

Mentre l'esempio seguente fa lo stesso usando la nuova sintassi:

```
void main(void)
{
      /* INITIALIZE */
      /* enable PC5 as output */
      DDRC|= _BV(PC5);


      /* BLINK, BLINK ... */

      /* PC5 is 5 (see file include/avr/iom8.h) and _BV(PC5) is 00100000 */
```

```
        while (1) {
              /* led on, pin=0 */
              PORTC&= ~_BV(PC5);
              delay_ms(500);
              /* set output to 5V, LED off */
              PORTC|= _BV(PC5);
              delay_ms(500);
        }
}
```

Il pezzo di codice presentato sopra mostra quanto sia semplice scrivere un programma. Qui per brevità viene mostrato solo la parte "main" del programma, e la funzione delay_ms è inclusa nel listato completo (avrm8ledtest.c). Per usare il pin PC5 come output bisogna settare il bit PC5 nel registro della direzione dei dati per la porta C (DDRC). Fatto ciò si può porre il pin PC5 a 0V chiamando la funzione cbi(PORTC,PC5) (pulisci il bit PC5) oppure si può mettere a 5V con sbi(PORTC,PC5) (set bit PC5). Il valore della stringa "PC5" è definito in iom8.h che è inclusa via io.h. Non è necessario preoccuparsi. Se avete già scritto programmi per sistemi multiutente/multitasking come linux saprete sicuramente che non bisogna utilizzare loop infiniti. Essi infatti sono uno spreco di CPU e rallentano il sistema. Nel caso degli AVR però non abbiamo altri programmi o altri task che girano, in effetti non c'è neppure un sistema operativo che gira. Dunque è normale usare loop infiniti

Compilare e caricare

Prima di cominciare è necessario assicurarsi di avere /usr/local/avr/bin nel PATH. Se necessario editate .bash_profile o .tcshrc e aggiungete:

export PATH=/usr/local/avr/bin:${PATH} (for bash)
setenv PATH /usr/local/atmel/bin:${PATH} (for tcsh)

Noi utilizzeremo la porta parallela e uisp per programmare l'AVR. uisp fa uso dell'interfaccia ppdev del kernel. Perciò bisogna caricare nel kernel i seguenti moduli necessari. Controllate con il comando /sbin/lsmod se sono caricati:

```
# /sbin/lsmod
parport_pc
ppdev
parport
```

altrimenti caricateli (come root):

```
modprobe parport
modprobe parport_pc
modprobe ppdev
```

E' in generale una buona idea far eseguire tali comandi automaticamente durante lo startup E' possibile aggiungerli ad un rc script (es. per Redhat in /etc/rc.d/rc.local)
Per utilizzare l'interfaccia pppdev come utente normale è necessario che root dia i permessi in scrittura eseguendo una volta il comando:

chmod 666 /dev/parport0

E' necessario inoltre accertarsi che non ci sia nessun demone di stampa in ascolto sulla porta parallela. Se così non fosse fermatelo prima di connettere il programmatore alla porta parallela. Ora tutto è pronto per compilare

il programma e caricarlo sul microcontrollore.

Il pacchetto contenente il progetto di prova (avrm8ledtest−0.1.tar.gz) include anche un make file. Basta quindi scrivere:

make
make load

Tale comando compilerà il software e lo caricherà sul microcontrollore. Non ho intenzione di scendere nei dettagli di tutti i comandi. E' possibile vederli nel Makefile e sono sempre i soliti. Neppure io me li ricordo tutti. Mi ricordo solo che mi basta fare "make load". Se volete scrivere un programma diverso, basta sostituire il nome del vostro nuovo programma al posto di avrm8ledtest.

Alcune interessanti binutils

Molto più interessanti del semplice processo di compilazione sono alcune delle binutils.
Tali utilities non sono però cambiate molto rispetto a quanto scritto nel Marzo 2002. Basta quindi dare un'occhiata al capitolo "Alcune interessanti binutils" nell' articolo 231 del Marzo 2002.

Idee e suggerimenti

L'ATmega8 è in generale compatibile con AT90S4433. E' necessario programmare i "fuse bits" per utilizzare l'oscillatore esterno, ma in generale l'hardware presentato in precedenza dovrebbe funzionare senza modifiche sostanziali. Purtroppo non ho avuto il tempo di ritestare tutti i circuiti per l'ATmega8 Per stare sicuri, è meglio utilizzare l'AT90S4433 per i vecchi articoli. Se le sfide non vi spaventano potete provare l'ATmega8 con i vecchi articoli/circuiti.

Ecco una lista degli articoli pubblicati:

- Pannello di controllo LCD per un server Linux
- Alimentatore DC basato su microcontrollori
- Frequenzimetro/contatore 1Hz−100MHz con display LCD e interfaccia RS232.
- Un display LCD per Linux su USB con watchdog e pulsanti.
- Costruire un robot "inseguitore di luce" autonomo.

Nota: il programmatore presentato in questo articolo include già le resistenze di protezione che nel caso dei vecchi articoli, erano montate direttamente sui circuiti. Per usare il nuovo programmatore con i circuiti vecchi basta sostituire le resistenze di protezione presenti sui circuiti stessi con dei fili.

Atmel fornisce una application note "AVR081: Replacing AT90S4433 by ATmega8" che descrive le incompatibilità tra i due microcontrollori: at90s4433_to_atmega8.pdf (101343 bytes)

Riferimenti

- Pascal Stang's AVRlib: http://www.procyonengineering.com/avr/avrlib/index.html or http://hubbard.engr.scu.edu/embedded/avr/avrlib/
- L'assembler tavrasm : www.tavrasm.org
- **Tutto il software e la documentazione menzionata nell'articolo**
- Sito web di Atmel: www.atmel.com

- Pagine dedicate all'elettronica dello shop tuxgraphics: shop.tuxgraphics.org
  (Qui è possibile acquistare il "Linux AVR programming CD", i kit e i microcontrollori)

2005−02−04, generated by lfparser_pdf version 2.51