LinuxFocus article number 231
http://linuxfocus.org

De AVR Microcontroller programmeren met GCC

door Guido Socher (homepage)

Over de auteur:

Guido houdt van Linux, niet alleen om dat leuk is om de fantastische mogelijkheden van dit systeem te ontdekken, maar ook vanwege de mensen die betrokken zijn bij het ontwerp ervan.

Vertaald naar het Nederlands door:
Guus Snijders
<ghs/at/linuxfocus.org>

Kort:

De AVR 8-Bit RISC Microcontroller van Atmel is een veel voorkomende Microcontroller. Het is een enkel geïntegreerd circuit met EEPROM, RAM, Analoog naar Digitaal convertor, een aantal digitale input en output lijnen, timers, UART voor RS232 commmunicatie en meer.

Het beste is echter dat een complete programmeer omgeving beschikbaar is onder Linux: De microcontroller is te programmeren in C met behulp van GCC. In dit artikel zal ik uitleggen hoe GCC te installeren en te gebruiken. Ook zal worden uitgelegd hoe de software in de Microcontroller geladen wordt. Alles wat je hiervoor nodig hebt zijn een AT90s4433 Microcontroller, een 4Mhz kristal, kabels en een paar andere goedkope onderdelen.

Dit artikel is slechts eeen introduktie. In een later artikel zullen we een LCD paneel met een paar drukknoppen, analoge en digitale inputs, een hardware Watchdog en een paar LED’s bouwen. Het idee is dat dit een algemeen bruikbaar contole paneel wordt voor een Linux Server, maar eerst zullen we kijken hoe de programmeer omgeving wordt opgezet en dat is waar dit artikel over gaat.

Software installatie: benodigdheden

Om de GNU C ontwikkel omgeving te gebruiken, heb je de volgende software nodig:

binutils-2.11.2.tar.bz2	Beschikbaar op: ftp://ftp.informatik.rwth-aachen.de/pub/gnu/binutils/ of ftp://gatekeeper.dec.com/pub/GNU/binutils/
gcc-core-3.0.3.tar.gz	Beschikbaar op: ftp://ftp.informatik.rwth-aachen.de/pub/gnu/gcc/ of ftp://gatekeeper.dec.com/pub/GNU/gcc/
avr-libc-20020106 .tar.gz	De AVR C-library is beschikbaar op: http://www.amelek.gda.pl/avr/libc/ Je kunt het ook downloaden vanaf deze server: download pagina
uisp-20011025.tar.gz	De AVR programmer is beschikbaar op: http://www.amelek.gda.pl/avr/libc/ Je kunt het ook hier downloaden: download pagina

We zullen alle programma's installeren in /usr/local/atmel. Dit is om het programma gescheiden te houden van je normale Linux C compiler. Het maken van deze directory gaat met:

mkdir /usr/local/atmel

Software installatie: GNU binutils

Het binutils pakket bevat alle low-level utilities voor het bouwen van object bestanden. Het bevat een AVR assembler (avr-as), linker (avr-ld), bibliotheek tool (avr-ranlib, avr-ar),programma's voor het generen van object bestanden die in de EEPROM van de microcontroller kunnen worden (avr-objcopy), disassembler (avr-objdump) en utilities zoals avr-strip en avr-size.

Om de binutils te compileren en te installeren, gebruik:

```
bunzip2 -c binutils-2.11.2.tar.bz2 | tar xvf -
cd binutils-2.11.2
./configure --target=avr --prefix=/usr/local/atmel
make
make install
```

Voeg de regel /usr/local/atmel/lib toe aan het bestand /etc/ld.so.conf en gebruik het commando /sbin/ldconfig om de cache opnieuw op te bouwen.

Software installatie: AVR gcc

avr-gcc zal onze C compiler zijn.

Gebruik de volgende commando's om te compileren en te installeren: Run the following command to build and install it:

```
tar zxvf gcc-core-3.0.3.tar.gz
cd gcc-core-3.0.3
./configure --target=avr --prefix=/usr/local/atmel --disable-nls --enable-language=c
```

```
make
make install
```

Software installatie: De AVR C-library

De c-library wordt nog altijd doorontwikkeld. De installatie kan per release veranderen. Als je de instructies stap voor stap wilt volgen, raad ik aan om de versie te gebruiken die in de tabel wordt vermeld. Deze versie heb ik getest, en deze werkt goed voor alle programma's die we in deze en volgende artikelen zullen schrijven.

De omgevings variabelen (syntax is voor bash):

```
export CC=avr-gcc
export AS=avr-as
export AR=avr-ar
export RANLIB=avr-ranlib
export PATH=/usr/local/atmel/bin:${PATH}

./configure --prefix=/usr/local/atmel --target=avr --enable-languages=c --host=avr
make
make install
```

Software installatie: De Programmeur

De programmeur software laadt de speciaal geprepareerde object code in de EEPROM van onze Microcontroller.

De uisp programmeur voor Linux is een erg goede programmeur. Het kan direct gebruikt worden vanuit een Makefile. Je voegt alleen een "make load" regel toe en je kunt de software compileren en laden in een keer.

De installatie van uisp gaat als volgt in z'n werk:

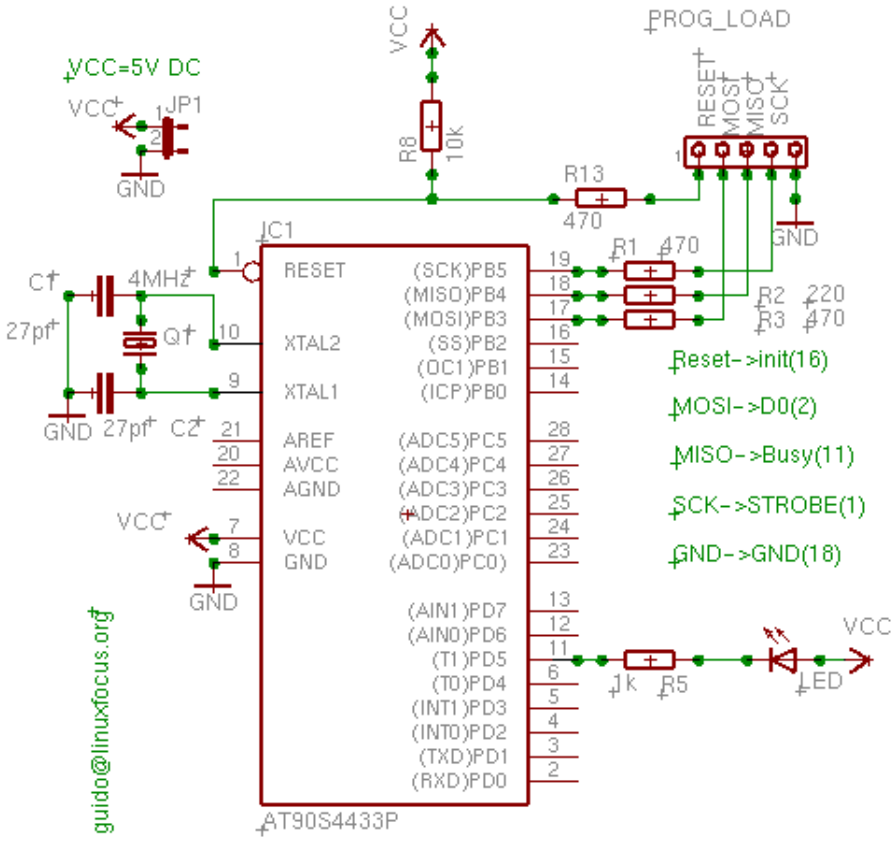
```
tar zxvf uisp-20011025.tar.gz
cd uisp-20011025/src
make
cp uisp /usr/local/atmel/bin
```

Een klein test project

We zullen beginnen met een klein test circuit. Het doel van dit circuit is alleen om onze ontwikkel omgeving te testen. We kunnen het gebruiken voor het compileren, downloaden en testen van een

eenvoudig programma. Het programma zal een LED laten knipperen.

Ik stel voor om een kleine printplaat te maken voor de microcontroller. Deze printplaat is later uit te breiden voor je eigen experimenten. Het is handig om hiervoor een breadboard te gebruiken. Je zou echter niet moeten proberen om de AVR met z'n 4Mhz kristal direct op het breadboard te plaatsen. Het is beter om een paar korte draden te gebruiken om input en output lijnen te verbinden met het breadboard daar zulke breadboard niet zijn gemaakt voor snelle digitale circuits. Het 4Mhz kristal en de condensators zouden fysiek dicht bij de microcontroller worden geplaatst.

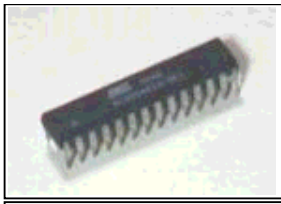
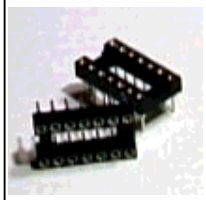
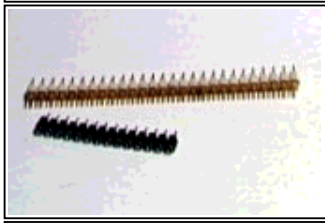
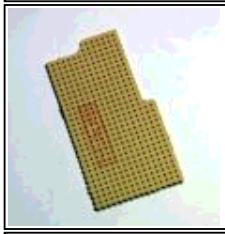


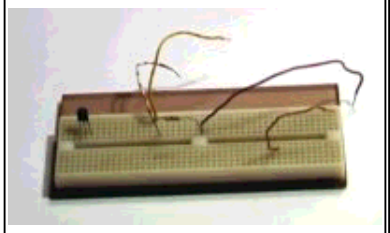
De weerstanden op de connector voor de programmeur zijn eigenlijk niet nodig in ons geval. Je hebt ze alleen nodig als je van plan bent de poort-B input/output lijnen voor andere doeleinden te gebruiken.

Udo Puetz heeft een, zo mogelijk, nog beginners-vriendelijk schema gemaakt, welke hier te vinden is: avr_layout_newbiefriendly.gif.

Benodigde Hardware

Wat je nodig hebt zijn de onderdelen die genoemd die genoemd worden in de tabel hieronder. Alle zijn erg veel voorkomend en goedkoop. Alleen de microcontroller is iets duurder, zo'n 7.50 Euro. Hoewel het een heel veel voorkomende microcontroller is, is hij niet in iedere lokale radio shop verkrijgbaar, maar bij de grotere distributeurs zoals (www.reichelt.de (germany), www.conrad.de (germany), www.selectronic.fr (france), etc..., waarschijnlijk zijn er in elk geval vergelijkbare in jouw land) is hij waarschijnlijk uit voorraad leverbaar.

	1 x AT90S4433, Atmel 8 bit Avr risc processor.
	2 x 14 pin IC socket or 1 x 28 pen 7.5mm IC voet De 28 pen voet is iets lastiger verkrijgbaar. Meestal zijn de 28 pens voetjes 14mm breed, maar wij moeten een 7.5mm voetje hebben.
	1 x 10K Ohm weerstand (kleur code: bruin,zwart,oranje) 3 x 470 Ohm weerstand(kleur code: geel, paars, bruin) 1 x 1K Ohm weerstand (kleur code: bruin, zwart, rood) 1 x 220 Ohm weerstand (kleur code: rood, rood, bruin) 1 x 4Mhz Kristal 2 x 27pf keramische condensator
	Een willekeurige 5 pens connector/voet voor de programmeur. Meestal koop ik zulke strips van connectoren en breek er 5 af.
	matrix bord
	1 x DB25 connector om aan te sluiten op de paralle poort.
	1 x LED

	Een breadboard. We zullen er hier verder geen gebruik van maken, maar het kan handig zijn als je verder wilt expermenteren de AVR. Ik stel voor dat je de Microcontroller samen met het kristal en de condensators op op het matrix bord laat en de input/output lijnen via korte draden verbindt met het breadboard.

Behalve bovenstaande onderdelen heb je ook nog een electronisch gestabiliseerde 5V gelijkspanning voeding nodig. Eventueel zou je ook een 4.5V batterij kunnen gebruiken als stroombron.

De programmeur hardware bouwen

De AT90S4433 biedt de mogelijkheid van 'in circuit programming' (ISP). Dat wil zeggen dat je de Microcontroller niet hoeft te verwijderen van bord om hem te programmeren. Het is mogelijk om kant en klare programmeur hardware aan te schaffen voor zo'n 50-150 Euro. Het is echter niet nodig om zoveel te investeren in een programmeur. Met Linux, de uisp software en een vrije paralle poort kun je zelf een heel goede en eenvoudige programmeur bouwen. Het is een eenvoudige kabel. De bedrading voor de programmeur is als volgt:

pen op de AVR	**Pen op de parallele poort**
Reset (1)	Init (16)
MOSI (17)	D0 (2)
MISO (18)	Busy (11)
SCK (19)	Strobe (1)
GND	GND (18)

De kabel zou niet langer moeten zijn 70cm.

Software schrijven

De AT90S4433 kan geprogrammeerd worden in plain C met de hulp van gcc. Kennis van AVR assembler kan handig zijn maar niet vereist. De AVR libv komt met een avr-libc-referentie, waar de meeste functies in gedocumenteerd staan. Harald Leitner heeft een document geschreven met veel nuttige voorbeelden over hoe de AVR en GCC te gebruiken: (haraleit.pdf, 286Kb, origineel van http://www.avrfreaks.net/AVRGCC/). Vanaf Atmel's website, (www.atmel.com, ga naar: avr products -> 8 bit risc-> Datasheets), de complete datasheet is te downloaden (lokale kopie: avr4433.pdf, 2361Kb) . Het beschrijft alle registers en het gebruik van de CPU.

Iets om goed in de gaten te houden tijdens het gebruik van de 4433 is dat er slechts 128 Bytes RAM en 4K EEPROM beschikbaar zijn. Dit betekend dat je geen grote datastructuren of strings moet gaan declareren. Je programma's zouden geen diep geneste functie aanroepen of recusion moeten gebruiken. Een regel als

char string[90];
is al teveel. Een integer is 16 bit. Als je een kleine integer nodig hebt, gebruik dan
unsigned char i; /* 0-255 */
Je zult echter nog verbaasd staan over wat voor grote programma's je kunt gebruiken. Het is echt een heel krachtige processor!

Veel beter dan alle theorie is een voorbeeld. We zullen een programma schrijven dat de LED zal laten knipperen met een 0.5 seconde interval. Niet erg nuttig, maar goed om mee te beginnen en om de ontwikkel omgeving en de programmeur te testen.

```
void main(void)
{
    /* enable PD5 as output */
    sbi(DDRD,PD5);
    while (1) {
        /* led on, pin=0 */
        cbi(PORTD,PD5);
        delay_ms(500);
        /* set output to 5V, LED off */
        sbi(PORTD,PD5);
        delay_ms(500);
    }
}
```

Het bovenstaande stukje code laat zien hoe simpel het is om een programma te schrijven. Alleen het eigenlijke programma is weergegeven, de delay_ms functie is opgenomen in de volledige lijst (avrledtest.c). Om pen PD5 als output te gebruiken moet je de PD5 bit in het data direction register voor poort D (DDRD) zetten. Hierna kun je PD5 op 0V zetten met de functie sbi(PORTD,PD5) (set bit PD5). De waarde van "PD5" is gedefinieerd in io4433.h welke is opgenomen via io.h. Je hoeft je er geen zorgen over te maken. Als je al eerder programma's voor multi user / tasking systemen zoals Linux hebt geschreven, weet je dat je nooit een 'non blocking' oneindige loop moet programmeren. Dit zou een verspilling van CPU tijd zijn en zorgt voor een flinke vertraging van het systeem. In het geval van de AVR ligt dit anders. We hebben niet zoiets als meerder taken, en er draait geen ander programma. Er is zelfs geen besturingssysteem aanwezig. Het is daarom vrij normaal om een oneindige lus te gebruiken.

Compileren en laden

Voordat je begint, controleer of /usr/local/atmel/bin voorkomt in je PATH. Indien nodig, open je .bash_profile of .tcshrc en voeg de volgende regel toe:

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

We gebruiken de parelle poort en uisp om de AVR te programmeren. Uisp gebruikt de ppdev interface van de kernel. Daarom is het nodig dat de volgende kernel modules zijn geladen:

/sbin/lsmod
parport_pc

ppdev
parport

Controleer met het commando /sbin/lsmod dat ze zijn geladen, anders kun je ze laden (als root) met

modprobe parport
modprobe parport_pc
modprobe ppdev

Het is misschien een goed idee om deze commando's automatisch te laten uitvoeren tijdens het opstarten. Je kunt ze toevoegen aan een rc script (voor Redhat bijv. /etc/rc.d/ rc.local).
Om gebruik te kunnen maken van de ppdev interface als normale gebruiker moet root je schrijf toegang geven. Dit kan met het commando:

chmod 666 /dev/parport0

Wees ook zeker dat er geen printer daemon draait op de parallele poort. Indien je er een hebt draaien, stop deze voordat je de programmeer kabel aansluit. Nu is alles klaar om te compileren en onze Microcontroller te programmeren.

Het pakket voor ons test programma (avrledtest-0.1.tar.gz) bevat een make file. Het enige wat je hoeft te doen is dit te typen:
make
make load
Dit zal de software compileren en laden. Ik zal niet ingaan op details van alle commando's. Je kunt ze bekijken in de Makefile en ze zijn altijd hetzelfde. Ikzelf kan ze me niet allemaal herinneren. ik weet alleen dat ik gebruik kan maken van "make load". Als je een ander programma wilt schrijven, vervang dan iedere keer dat avrledtest voorkomt door de naam van jouw programma.

Een paar interessante binutils

Interessanter dan het eigenlijke compilatie proces zijn sommige van de binutils.

avr-objdump -h avrledtest.out

Geeft de grootte weer van de verschillende secties in je programma. .text is de instructie code en laadt in de flash EEPROM. .data is geïnitialiseerde data zoals static char str[]="hello";
.bss niet geïnitialiseerde globale data. Beide zijn nul in ons geval. De .eeprom is voor variabelen opgeslagen in eeprom. Ik heb hier geen gebruik van gemaakt. stab en stabstr is debug info en zal niet in de AVR komen.

```
avrledtest.out:     file format elf32-avr

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000008c  00000000  00000000  00000094  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  00800060  0000008c  00000120  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00800060  0000008c  00000120  2**0
```

```
                  ALLOC
  3 .eeprom       00000000  00810000  00810000  00000120  2**0
                  CONTENTS
  4 .stab         00000750  00000000  00000000  00000120  2**2
                  CONTENTS, READONLY, DEBUGGING
  5 .stabstr      000005f4  00000000  00000000  00000870  2**0
                  CONTENTS, READONLY, DEBUGGING
```

Je kunt ook het commando avr-size gebruiken om dit in meer gecomprimeerde vorm te krijgen:

avr-size avrledtest.out

```
   text    data     bss     dec     hex filename
    140       0       0     140      8c avrledtest.out
```

Als je werkt met de AVR moet je in de gaten houden dat text+data+bss niet groter is dan 4K en data+bss+stack (je kunt de grootte van de stack niet zien, deze hangt af van hoeveel geneste functie aanroepen je hebt) mag niet groter zijn dan 128 Bytes.

Ook interessant is het commando

avr-objdump -S avrledtest.out

Het zal een assembler listing van je code genereren.

Conclusie

Nu weet je genoeg om je eigen projecten te beginnen met de AVR hardware en GCC. Er zullen ook verdere artikelen verschijnen in LinuxFocus met meer complexere en vooral interessantere hardware.

Referenties

- Libc en uisp: /www.amelek.gda.pl/avr/libc/
- GCC en binutils: ftp://gatekeeper.dec.com/pub/GNU/
- avrfreaks (let op, sommige mensen op die site gebruiken nog steeds Windows!?): http://www.avrfreaks.net/
- de tavrasm assembler voor Linux: www.tavrasm.org
- AVR webring: R.webring.com/hub?ring=avr&list
- Voorgecompileerde versie van gcc: combio.de/avr/
- All software en documenten genoemd in dit artikel
- De website van atmel: www.atmel.com/

Site onderhouden door het LinuxFocus editors team	Vertaling info:
© Guido Socher	en --> -- : Guido Socher (homepage)
"some rights reserved" see linuxfocus.org/license/	en --> nl: Guus Snijders <ghs/at/linuxfocus.org>
http://www.LinuxFocus.org	

2005-01-14, generated by lfparser_pdf version 2.51