Richting:	Elektronica HBO5
Vak:	Microcontrollers Bb3
Schooljaar:	2010-2011 semester 1
Lesgever:	Peter Van Grieken <peter@team1.be>
Cursist:	Tom Van Looy <tom@ctors.net>
Verslag:	Individuele opdracht 1 Linux development
Beschrijving:	Stel documentatie op over de werking van dev.tools voor microcontrollers onder Linux (min PIC en AVR).

Index

Inleiding

Microcontrollers zijn in feite kleine op zichzelf staande systeempjes die veelal gebruikt worden in embedded toepassingen. De software voor de microcontrollers wordt daarom doorgaans op een computer geschreven. Hier heeft de programmeur immers alle comfort, zoals een groot scherm, een geïntegreerd ontwikkel omgeving en de mogelijkheid om documentatie te raadplegen. In sommige gevallen kan de software op de computer via simulatie getest en dan pas op de microcontroller zelf geladen worden. Producenten van controllers en ontwikkel-kits leveren meestal zelf development software aan, in veel gevallen zal dit enkel software zijn voor het ontwikkelen onder Windows. Dit verslag beschrijft de mogelijkheden om microcontroller software te ontwikkelen op een Linux systeem.

Als we het ontwikkelingsproces volgen, zijn er enkele aspecten die we van elkaar moeten onderscheiden. Het schrijven van C of assembler code kan in principe in elke text editor gebeuren. Enkele voorbeelden hiervan op Linux zijn gedit (GNOME), kate (KDE), VI en Emacs. Zelfs de meest eenvoudige editor heeft meestal al een basis ondersteuning voor programmeertalen aan boord, een voorbeeld hiervan is syntax highlighting.

Het verschil met bijvoorbeeld de MikroC ontwikkel omgeving is dat laatst genoemde een volledige geïntegreerde ontwikkel omgeving, of in het Engels integrated development environment (IDE), is. Zowel de editor, compiler, debugger, simulator en programmer zitten hier in één tool verwerkt. Deze zijn sterk met elkaar verbonden via menu opties, bijvoorbeeld “build and program”. En worden ondersteund door een gecentraliseerd helpsysteem.

Deze aspecten komen in dit verslag allemaal aan bod. De gegeven informatie is theoretisch en onder voorbehoud. Het in de praktijk aftoetsen van de theorie zou me te ver hebben geleid. Betalende, closed-source oplossingen werd niet onderzocht.

Compilatie

Als de code eenmaal geschreven is, moet deze gecompileerd worden tot uitvoerbare code voor de microcontroller. Aangezien de computer op een ander platform draait dan de microcontroller moet de code via een cross-compiler omgezet worden.

AVR

De GNU Compiler Collection (gcc)[1] en GNU binutils[2] (assembler, linker) hebben de mogelijkheid om te compileren voor verschillende ATMEL AVR instructie sets of

1 http://gcc.gnu.org/
2 http://www.gnu.org/software/binutils/

MCU types. GNU stelt ook een C bibliotheek[3] voor AVR ter beschikking, deze bibliotheek bevat bijvoorbeeld wiskundige functies, input/output en string verwerking functies. Maar, ook zaken zoals de AVR opstart procedures, het initialiseren van de vector tabel, stack pointer, etc. worden door deze bibliotheek afgehandeld. We kunnen deze dus best wel gebruiken.

Bij het compileren geven we aan GCC via de -mmcu optie mee voor welk doel we willen compileren. Echter, er zal steeds een bestand afgeleverd worden in het executable and linkable format (ELF)[4]. Dit is feite een door Linux uitvoerbaar bestand, maar je kan dit niet gewoon op je systeem gebruiken. Je kan dit echter wel gebruiken bij het debuggen.

Het genereren van een bestand dat opgeladen kan worden naar de microcontroller kan via het commando avr-objcopy. Deze zet het ELF bestand om naar Intel HEX[5] formaat. Via een tool als AVRDUDE[6] kan je het HEX bestand dan opladen naar de microcontroller. Je moet aan AVRDUDE meegeven welke programmer je gebruikt. Er is ondersteuning voor een hele reeks programmers, bijvoorbeeld voor de AVR dragon[7], of de arduino[8].

Wanneer je op internet zoekt naar bovenstaande tools, zal je al snel tutorials en howto's vinden om hier verder mee aan de slag te gaan. Bovendien kan al deze software eenvoudig geïnstalleerd worden, bijvoorbeeld in Ubuntu via het software center.

PIC

Over het programmeren voor de Microchip PIC microcontrollers is meteen een pak minder informatie te vinden. De Small Device C Compiler (sdcc)[9] heeft work-in-progress ondersteuning voor PIC14 en PIC16 controllers. Via de -m en -p flags kan je aangeven voor welk type je wilt compileren. Een andere compiler is CPIK[10]. Deze heeft ondersteuning voor PIC18. Het GNU gputils[11] project levert een assembler en linker aan. Voor sdcc zitten de C header files onder /usr/share/sdcc/include.

Wanneer je je code tot een HEX bestand gecompileerd hebt via deze tools, moet je het HEX bestand op de microcontroller laden. Hiervoor kan je, voor met PICStart+ compatibele programmers, het programma picp[12] gebruiken. Voor de PICkit2 programmer van Microchip, kan je pk2cmd[13] gebruiken. Voor andere

3 http://www.nongnu.org/avr-libc/
4 http://en.wikipedia.org/wiki/Executable_and_Linkable_Format
5 http://en.wikipedia.org/wiki/Hex_file
6 http://www.nongnu.org/avrdude/
7 http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3891
8 http://www.arduino.cc/
9 http://sdcc.sourceforge.net/
10 http://pikdev.free.fr/download-cpik.php3
11 http://gputils.sourceforge.net/
12 http://home.pacbell.net/theposts/picmicro/
13 http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en023805&redirects=pickit2

progammers kan je eens kijken naar PICPgm[14], deze software ondersteunt ook een reeks programmers.

Andere

Ik ben de eigenaar van een DeVaSys USB-I2C/IO[15] interface board. Voor dit verslag heb ik dat board ook even van naderbij bekeken.

Het board bevat een Cypress AN2131QC microcontroller. De AN2131QC valt onder de Cypress EZ-USB familie. De controller bevat een processor die afgeleid is van de Intel 8051, met enkele USB I/O uitbreidingen. Het leuke is dat de assembler code van de 8051 nog steeds werkt voor de AN2131QC microcontroller.

De reeds besproken sdcc compiler kan compileren voor de Intel MCS-51 (8051) familie. We kunnen dus via Linux HEX files maken voor het board. De HEX files kunnen op het board geladen worden via fxload[16]. Dit programma is een specifiek programma om firmware te downloaden naar EZ-USB devices.

Daniel Clemente heeft een handleiding[17] geschreven over het ontwikkelen voor een EZ-USB board in Linux. Als deel van zijn onderzoek schreef hij een bibliotheek met specifieke functies voor de uitbreidingen die Cypress heeft toegevoegd. Zijn handleiding is verder een zeer goed startpunt om het onderwerp verder uit te diepen.

Debugging

Een essentieel deel van software ontwikkeling is debugging. Dit is een methodologisch proces om fouten en defecten in de software te vinden.

AVR

AVR applicaties kunnen gedebugged worden via software emulatie of on-chip via JTAG. Voor AVR heeft de GNU debugger (gdb) een speciale versie ter beschikking, namelijk avr-gdb[18]. Debugging is hierbij gebaseerd op het client-server model. De server kan dan ofwel een simulator zijn, of een JTAG interface. Met avr-gdb kan je vervolgens op de server verbinden.

SimulAVR[19] is een simulator voor de Atmel AVR microcontroller famillie (ATtiny en ATmega). SimulAVR heeft een Python/TCL GUI frontend en kan zowel als avr-gdb remote target, alsook stand alone gedraaid worden.

AVaRICE[20] is een GDB server voor JTAG verbindingen. Hiermee is in-system debugging mogelijk. AVaRICE heeft onder meer ondersteuning voor de AVR

14 http://www.members.aon.at/electronics/pic/picpgm/index.html
15 http://www.devasys.com/usbi2cio.htm
16 http://sourceforge.net/projects/linux-hotplug/
17 http://www.danielclemente.com/placa_pi/index.en.html
18 http://www.gnu.org/software/gdb/
19 http://www.nongnu.org/simulavr/
20 http://sourceforge.net/projects/avarice/

Dragon of AVR MkI/MkII compatibele toestellen. Een andere mogelijkheid voor in-system debugging is uisp[21].

PIC

Voor PIC is het aanbod minder groot. Gpsim[22] uit het gputils project, is een complete software simulator voor de Microchip PIC microcontrollers. Een andere, specifiek voor PIC16F84 microcontrollers, is simulpic[23].

IDE

Steeds werken via de commandolijn is vrij omslachtig, vooral omdat er bij elk commando een reeks flags en opties moeten meegegeven worden. Via GNU make [24] kan je dit proces vereenvoudigen. Make is een build systeem waarbij je met targets werkt. Je kan dus achter commando's als “make hex” en “make program” een reeks aan commando's verbergen, die bij het aanroepen van een bepaald target uitgevoerd moeten worden.

Een geïntegreerde ontwikkelomgeving gaat nog een stap verder. Vergeleken met make krijg je dan eigenlijk targets ter beschikking via menu opties.

De reeds besproken GNU AVR tools zijn zo bijvoorbeeld verwerkt in de WinAVR[25] IDE. Deze is echter enkel voor Windows beschikbaar. Een voorbeeld van een op AVR toegespitste IDE voor Linux is KontrollerLab[26]. Deze omgeving is net zoals WinAVR gebaseerd op de GNU AVR tools.

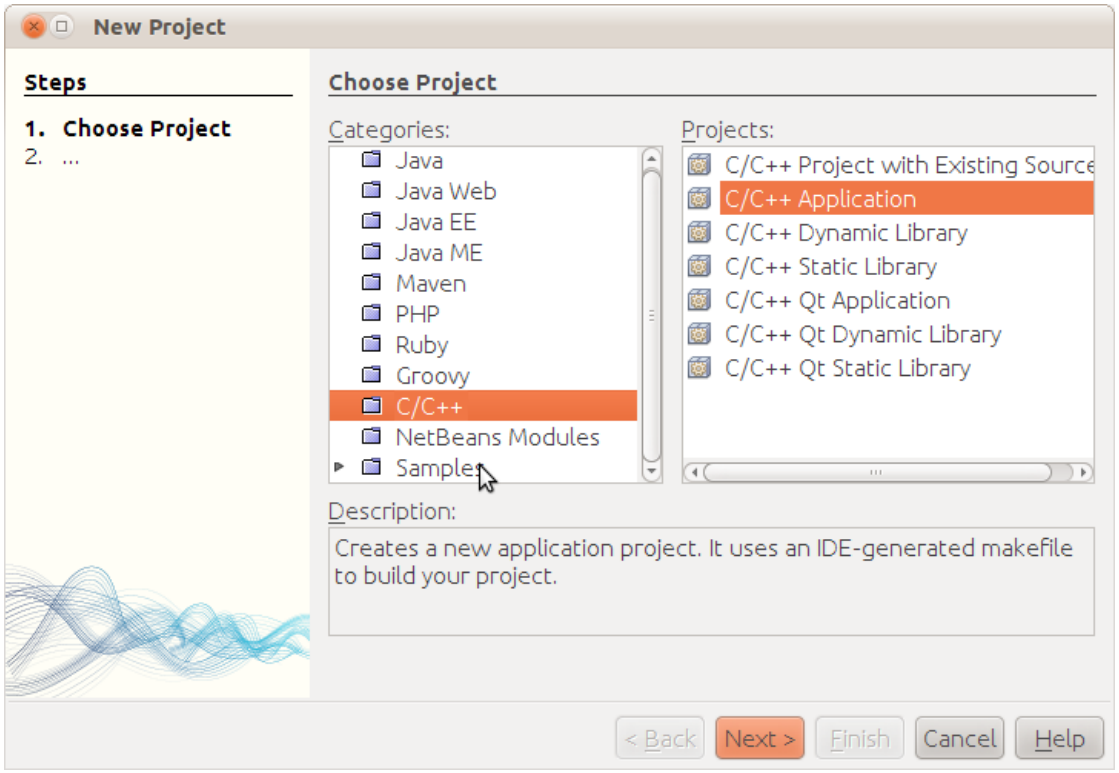
Ook voor PIC bestaan er specifieke IDE's. Een voorbeeld hiervan is PiKdev[27], van de makers van de reeds besproken CPIK compiler. Een ander voorbeeld is MPLAB X[28]. Deze IDE wordt ontwikkeld door Microchip zelf. MPLAB X is de opvolger van MPLAB 8, die enkel voor Windows beschikbaar is. Bij de X versie heeft men de IDE gebaseerd op NetBeans, waardoor deze multi-platform gebruikt kan worden. Microchip heeft compilers voor verschillende targets geïntegreerd. Bijvoorbeeld: PIC18, PIC10/12/16, dsPIC DSCs, PIC24 en PIC32 MCUs. De IDE heeft wel nog maar beperkte functionaliteit en is nog in beta, maar ziet er absoluut veelbelovend uit.

Er zijn ook universele IDE's beschikbaar. Als we in de categorie multi-platform en multilingual zoeken, kan je stellen dat Eclipse[29] en NetBeans[30] hier het grootste marktaandeel hebben. Zowel Eclipse als NetBeans maken C ontwikkeling via plugins mogelijk. Beide IDE's werken achter de schermen met een make file die via menu's kan geconfigureerd worden.

---

21 http://www.nongnu.org/uisp/
22 http://gpsim.sourceforge.net/
23 http://packages.ubuntu.com/natty/simulpic
24 http://www.gnu.org/software/make/
25 http://winavr.sourceforge.net/
26 http://sourceforge.net/projects/kontrollerlab/
27 http://pikdev.free.fr/
28 http://devupdates.microchip.com/mplab/
29 http://www.eclipse.org/
30 http://netbeans.org/

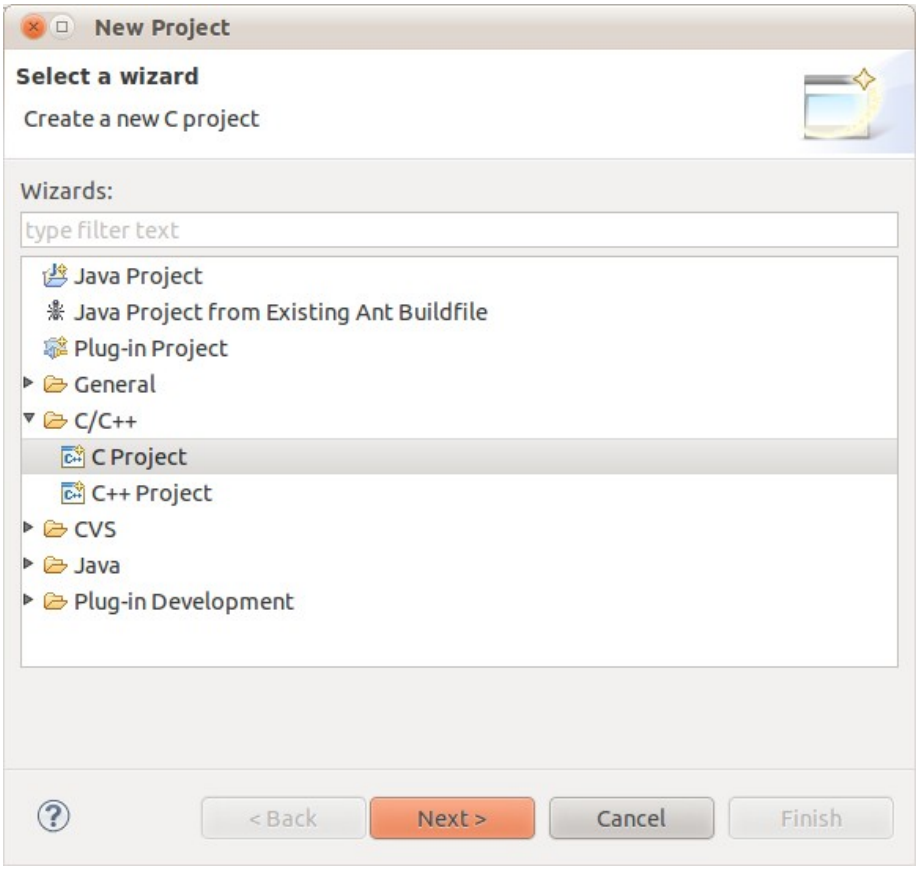
In NetBeans kan je C ontwikkeling activeren via de opties “tools” -> “plugins” -> “available plugins” -> “C/C++”. Vervolgens heb je de mogelijkheid om een C project aan te maken.



C project in NetBeans

Via de project-eigenschappen kan je het project verder configureren om te werken voor AVR. De compiler kan gekozen worden (avr-gcc), extra commandline-opties kunnen gezet worden (type MCU). De assembler en linker opties kunnen ook gewijzigd worden. Bij de opties voor het packagen kan je bijvoorbeeld commando's ingeven voor het bouwen van de HEX file. Het project kan op deze manier eigenlijk voor zowel AVR als PIC geconfigureerd worden.

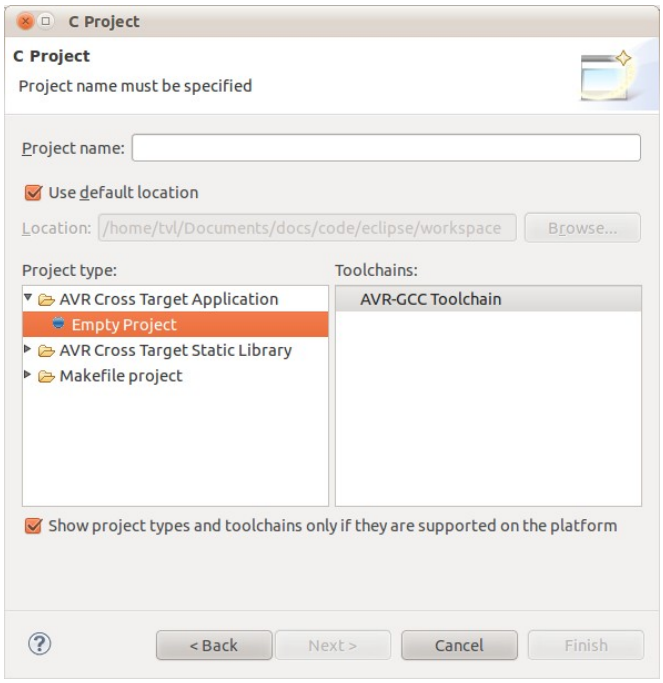
Voor Eclipse is er een specifieke plugin[31] beschikbaar voor AVR. Deze kan je installeren via de opties “help” -> “install new software” -> “add site”. Daar vul je dan “AVR Eclipse Plugin” en “http://avr-eclipse.sourceforge.net/updatesite” in. Vervolgens kan je de “AVR Eclipse Plugin” selecteren en installeren. Daarna heb je de mogelijkheid om een C project aan te maken.



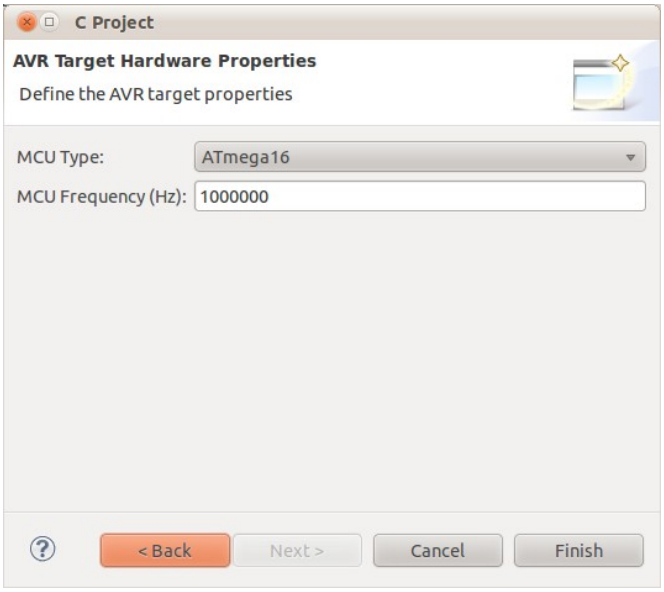
C project in Eclipse

De plugin is specifiek voor AVR, je krijgt dan ook meteen de keuze om de AVR-GCC toolchain te gebruiken voor je project. Hiermee stel je de compiler, assembler, linker, debugger etc. in. Dit is wat er in NetBeans via de project-eigenschappen manueel werd ingesteld. Ook het MCU type kan je meteen kiezen.

31 http://avr-eclipse.sourceforge.net/wiki/index.php/The_AVR_Eclipse_Plugin



Project type



MCU type

Uiteindelijk kom je terecht in een standaard Eclipse omgeving, met een aantal extra tabs, en menu opties voor AVR. In het hoofdmenu staat zo bijvoorbeeld een optie om het project naar je microcontroller te laden. De website[32] beschrijft ook een aantal debug mogelijkheden, gebaseerd op GDB.

PIC ontwikkeling via Eclipse kan ook, maar ik heb geen plugin gevonden waar je evenveel comfort hebt als bij de AVR plugin. Voor PIC kan je best vertrekken via een leeg Makefile project en vervolgens het project zelf configureren.

Conclusie

Er zijn best wel wat mogelijkheden voor zowel AVR als PIC ontwikkeling. Maar, als je nog een microcontroller moet kiezen, kan je best voor een AVR type gaan. En, eerst even kijken welke development tools deze ondersteunen.

Op vlak van debugging zijn er voor AVR de meeste mogelijkheden.

Ook op IDE vlak wordt AVR in het algemeen beter ondersteund dan PIC. Als je verregaande microcontroller ondersteuning verwacht van je IDE, kan je best voor Eclipse gaan.

32 http://avr-eclipse.sourceforge.net/wiki/index.php/Debugging