Week 7: Embedded Programming

11.3. 2015

This week we will learn about embedded programming.

The agenda:
http://academy.cba.mit.edu/classes/embedded_programming/index.html

architectures Harvard, von Neumann RISC, CISC microprocessor, microcontroller FPGA, CPLD ALA **memory** registers SRAM DRAM EEPROM FLASH fuse **peripherals** A/D comparator D/A timer/counter/PWM USART USB ... **word size** 8 16 32 64 **families** 8051 PIC MSP AVR ARM STM32 PSoC, xCORE, Propeller **vendors** Octopart Digi-Key Mouser Newark Farnell SparkFun **AVR processors** ATtiny10 ATtiny45V ATtiny44A ATmega328P ATmega16U2 ATxmega16E5 ATxmega16C4 **tutorials** **data sheets** **packages** DIP SOT SOIC TSSOP TQFP LQFP MLF, CSP, BGA **clocks** RC (10%, 1% calibrated) ceramic (0.5%) quartz (50 ppm) **in-system development** ISP (header, pads, clip) bootloader JTAG, debugWire, PDI ICE **programmers** ISP AVRISP FabISP avrdude JTAG, debugWIRE, PDI Atmel-ICE **assembly language** hex file instruction set, opcodes mnemonics, directives, expressions avr-as gavrasm **C** GCC AVR Libc modules types math avr-libc, binutils-avr, gcc-avr WinAVR CrossPack Atmel Studio **host communication** RS232 bit timing VT100/ANSI/ISO/ECMA terminal Kermit Minicom term.py USB software hardware FTDI cable libFTDI echo hello-world hello.ftdi.44.cad board components traces interior programming hello.ftdi.44.echo.c hello.ftdi.44.echo.c.make hello.ftdi.44.echo.interrupt.c hello.ftdi.44.echo.interrupt.c.make hello.ftdi.44.echo.asm hello.ftdi.44.echo.asm.make **IDE** Atmel Studio Eclipse AVR Firefly Scratch Modkit **boards** Arduino board + C libraries + IDE + bootloader Fabkit Fabio hello.arduino.328P.cad board components traces interior Blink.pde boards.txt programming hello.arduino.328P.blink.c hello.arduino.328P.blink.make programming ATtiny PSoC Maple Tessel BeagleBone PandaBoard Rasberry Pi **Interpreters** Python BASIC FORTH AVRSH JavaScript **debugging** "printf" Atmel Studio gdb, ddd, Insight **STM32** processor STM32F3 data sheet toolchain gcc-arm-none-eabi sudo add-apt-repository ppa:terry.guo/gcc-arm-embedded OpenOCD http://sourceforge.net/projects/openocd/files/latest/download?source=files QStlink2 sudo add-apt-repository ppa:mobyfab/qstlink2 stlink git clone https://github.com/texane/stlink.git programmer ST-Link V2 software ST library STMCube board STM32F3Discovery software blink.zip programming Nucleo read a microcontroller data sheet program your board to do something, with as many different programming languages and programming environments as possible

The assignment for this week::

To read a microcontroller data sheet and program your board to do something, with as many different programming languages and programming environments as possible .

Reviews:

Class:

```
Writing code for microcontrollers.  Make it do something is the
assignment.
```

Aiken computer at Harward: http://history-computer.com/People/AikenBio.html
Memory and code were separate

RISC -

CISC –

Microcontrollers –

VPGAx & CPLDs: http://www.latticesemi.com/Products/FPGAandCPLD.aspx

ALA: http://cba.mit.edu/docs/papers/11.12.Computing.pdf

Memory:

EEPROM –

FLASH:

Fuse: store the configuration of the processor

Peripherals:

A/D
Comparator
D/A
Timer/counter/ PWM
USART
USB

Word size:

Many processors use bigger size than they need

Families:

8051 – stay away (legacy)
PIC: http://www.microchip.com/pagehandler/en-us/products/picmicrocontrollers
MSP: low cost

AVR – we will be focusing on those, developed by students:
http://www.atmel.com/products/microcontrollers/avr/default.aspx
Now possible to write high level programmes and they function sufficiently
ARM – if you need more than the AVR

STM32: http://www.st.com/web/en/catalog/mmc/FM141/SC1169
Interesting if you want to push performance

Vendors:

Annoying buying, out of phase with the economy. Lead time almost always too long.

Octopart: search engine for parts.

Digikey: http://www.digikey.com/

Mouser

Newark

Farnell

SparkFun

AVR processors:

Different types

Tutorials: https://www.google.com/search?q=avr+tutorial

Data sheets:
http://academy.cba.mit.edu/classes/embedded_programming/doc8183.pdf

Reading the Data Sheet is essential to understand the hardware.

Packages:

DIP
SOT
SOIC
TSSOP
TQFP
LFTP
MLF, CSP, BGA

Clocks:

RC
Ceramic
Quartz (50 ppm)- crystals

Which one – how accurately you have to measure time.

Programmers:

Put að program into the processor

ISP
- AVRISP
- FabISP
- Avrdude

JTAG, debugWIRE, PDI
- Atmel-ICE

In-circuit emulator

Most of us will use ISP or... Atmel-ICE?

Assembly language:

Hex file – to be loaded into the processor
http://fab.cba.mit.edu/about/fab/hello/ftdi/hello.ftdi.44.echo.hex

instruction set:
http://academy.cba.mit.edu/classes/embedded_programming/doc0856.pdf

The higher level the the language, the further you are from it.

C:

Most of us will use C, a compiled language.
GCC, the GNU Compiler Collection: https://gcc.gnu.org/
If you are using Linux, install packages and you are set to go

Atmel Studio

If you use serial communication you have to set timing of the bits correctly

Host communication:

V-USB:

FDTI cable – from digikey is 20 dollars, possible to find for 10 dollars.

Mosi – master out slave in

Sclock
Ground pin
Power pin
Reset

10k resistor to reset from the power supply
1Mhz capacitor
Crystal – for running with accuracy

IDE:

Boards:

The important thing about writing programmes is tha nobody writes it from scratch – you modify it.
Start with a program that works, and make little changes

Main link – check

For this week – load the programme and make little changes to it.

Programming language:

Visual programming of controllers, look nice but you run into problems if doing something complecated

Arduino: set of C libraries, integrated program environment and a bootloader: http://www.arduino.cc/
Fabkit i/o: http://fab.cba.mit.edu/content/projects/fabkit/
Use the arduino workflow... you need to make it. 1-2 dollars.
http://makeyourbot.wikidot.com/fabio-1-1

high-low tech: http://highlowtech.org/?p=1695
AVR tutorial

Maple: http://leaflabs.com/devices/

Raspberry Pi: http://www.raspberrypi.org/

Debugging –

Atmel Studio can talk to ...

All of those can talk to the tool-chain.

AVRDUDE – is important. AVR Downloader/UploaDER

ST has been really agressive...

STM32F3Discovery: http://www.digikey.com/product-detail/en/STM32F3DISCOVERY/497-13192-ND

Assignment:

Program your board and try to do it in as many ways as you can. Run Atmel studio, try ... languages, to see what the environments are like.

Make your own Arduino.
Send message to your board, let your board send meessages.

For Friday: prepare, areas i/o pins, guides, downloading arduino, atmel studio, GCC – compiler

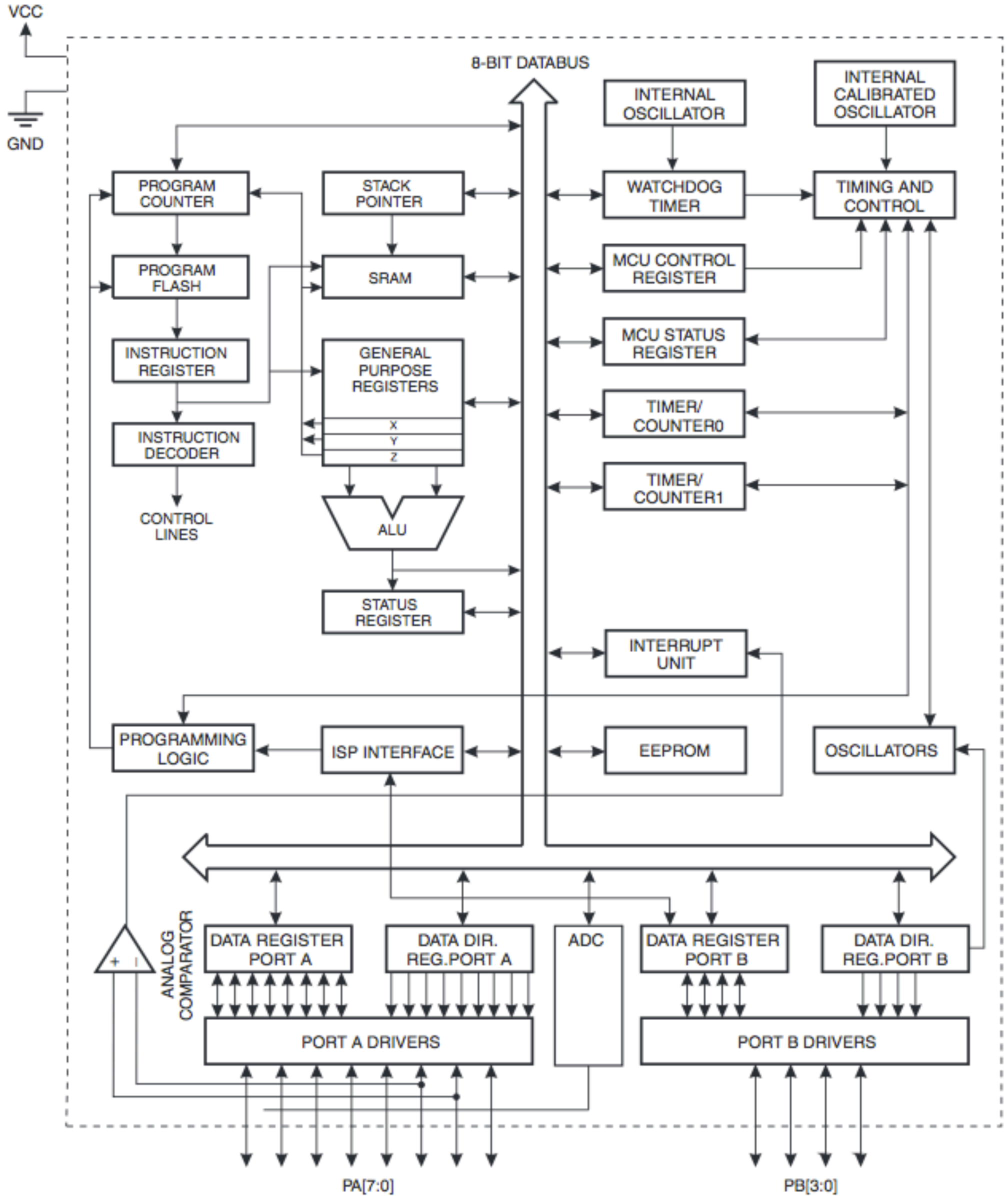
13.03.2015

Programming

Data sheet
Write a program for the boards so we can use the button to do something

In the Datasheet:
Fig. 2.1 Block Diagram – overview of the microcontroller
Figure 2-1.
Block Diagram

Figure 2-1. Block Diagram



Button – PB1   PB0
LED – PA7 8PCINT)

We want to listen to the button

We want to drive the LED – the pin will be output. 1 high or 5v

When the current flows we have got light.

You get information from the button, it talks to something and we get output

When we press the button, send voltage to the LED light

If it is pressed than do something, if it is not pressed do nothing

If there is current then it can be measured

Pull-up resistor – is a kind of a controller (for the current not to run wild). The microcontroller has a built-in pull-up resistor. How we turn this on and off is down to the programming

Port
Pin
DDR (data direction register)

Port A is used if you want to output something – make it 1 or 0 and turn it on and off

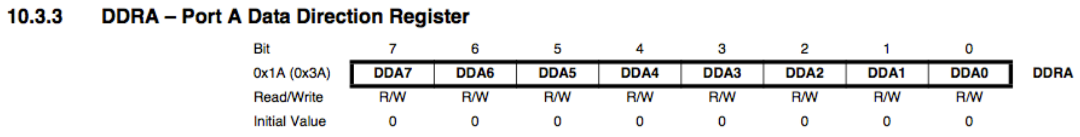
PIN – you can use to read from the port

DDR – determines if it is an input or output, 0 is in 1 is out

What the program needs to do – we have to make sure

Output on PA2
Input on PB2

When i start the program

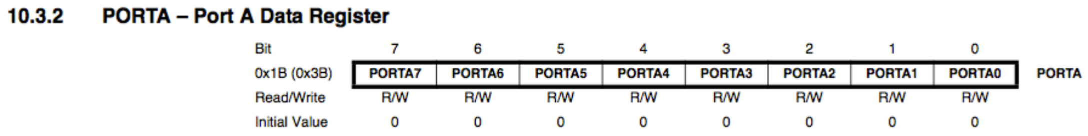


10.3.3 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x1A (0x3A)	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

DDRA (switch) – Set the PA2 to 1 (rafmagn á)

Write down where the LED and the BUTTON is



10.3.2 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

We write to get to 1 in the location we want, we write: 1≤≤PA2
= sendu straum til / um 7 bil PA2

Setup

Set PA2 to output
Enable the output to PB2
Set PB2 to input

Now we are set to run the main program

Loop

(We want to read our input – our PB2)

Read PB2

Then 2 options: on – off

When I press the button, I want the light to turn on, when I release it I want to turn if off

Push button, latching button – we have got a push button

- on = LED off
- off – LED on

Then we tell the program to continue, continuously

Programming in Arduino

I followed a tutorial on high-low tech – Programming an Attiny w/ Arduino 1.6: http://highlowtech.org/?p=1695

I started by downloading the Arduino 1.6; ide-1.6x.zip, unzipping it and locating the content of the attiny folder in a hardware folder that I created in Documents/Arduino/hardware/

I restarted the Arduino development environment.

I did set Tools – Board – Attiny
Then I specified the Clock (and the Processor – Attiny (internal 20 MHz clock)

Then I connected the Attiny & ISP to the computer for power.

Starting to program in Arduino...

...by opening the Blink file (File-Example-01.Basics-Blink) and changing the value of the digital pin

Reference online or off-line:
file:///Applications/Arduino.app/Contents/Resources/Java/reference/arduino.cc/en/Reference/HomePage.html

Uploading the sketch to the board – until uploaded.
If you then connect an LED between pin 0 and ground, you should see it blink on and off. In my case the LED did not light up

Then select Tool - Burn Bootloader – Done burning bootloader.

Upload again.

File – examples – digital – button.

Change the value for buttonPin og ledPin:

```
const int buttonPin = 3;     // the number of the pushbutton pin
const int ledPin =  7;      // the number of the LED pin
```

To activate pull-up resistor in microcontroller add this to setup (last line):

```
digitalWrite(buttonPin, HIGH);
```

Upload to send this to the microcontroller.
Then the button lights up continually.

We wanted to change the function of the button in such a way that if the button was pressed once it would light the LED, if the button was pressed twice the LED would not light.

This I did not manage to accomplish :-(

There is always another day...