NTNU – Trondheim
Norwegian University of
Science and Technology

Tutorial Lecture for Exercise 2
TDT4258 Energy Efficient Computer Design Lab

Stefano Nichele
Department of Computer and Information Science
2013, February 15th

Exercise 1

- Deadline: today 12:00 ☹ – It's Learning
- 1 report + code each group
- (Brief) presentation to vit.ass the week after submission (every group will present at least one exercise, you can be selected for presentation more than once). The presentations will be held in the lab.

Exercise 2

- Deadline: 15th March, It’s Learning
- Counts 20%
- The submission should contain:
  - The code you have produced
  - Report (IN ENGLISH)
    - Clear description (pictures, diagrams, flow charts, etc.)
    - Well-structured
    - Step-by-step description
    - Should be delivered on It’s Learning. If you have delivery problems send the report and the code to vit.ass by email (nichele@idi.ntnu.no)

Exercise 2

- Create a program that plays different sound effects when different buttons are pushed

Task requirements

- To be written in C language
- Run directly on STK1000
- At least 3 different sounds
- ABDAC should be fed with data in an interrupt routine (Audio Bitstream Digital-to-analog converter)
- You should use the GNU tools
  - GNU Compiler Collection (GCC)
  - Write a Makefile
  - Debugging with GDB

C – What and Why

- High-level language with good low-level opportunities
- Much easier to use than assembler
- ”C” syntax (familiar, Java has taken a lot of syntax from C)
- Not object oriented

C – What and Why

1. *The 3rd letter of the English alphabet*
2. *ASCII 1000011*
3. *C is often described, with a mixture of fondness and disdain varying according to the speaker, as a language that combines all the elegance and power of assembly language with all the readability and maintainability of assembly language.*

The Jargon File

(http://www.catb.org/~esr/jargon/html/C/C.html)

Fast C course: Pointers

- Poiters are variables that hold memory addresses

- Example:

```
int a = 5;          // variable of type int
int *p;             // pointer to int
p = &a;             // set p to point to a
*p = 42;            // modify the value pointed by p
```

NTNU – Trondheim
Norwegian University of
Science and Technology

Fast C course: Struct

- A struct is a collection of variables
  - Declaration:

```
struct foobar {
int foo;
double bar;
};
```

  - Usage:

```
struct foobar f;
f.foo = 8;
f.bar =  1.4;
```

  - Pointer to the struct:

```
struct foobar *fp = &f;
fp->foo = 7;                    /* same as (*fp).foo = 7 */
```

Fast C course: #include

- #include inserts the content of another file
- Used to include the header files, which contain declarations
- Examples:

```
#include <math.h>
#include <avr32/ap7000.h>
```

Compiling for AVR32

- Compiling:

```
avr32-gcc –Wall –g –c –o object.o source.c
```

- Linking:

```
avr32-gcc –o program.elf object.o
```

I/O in C

- A struct for each I/O port, declared in <avr32/ap7000.h>
- Struct avr32_pio_t has a field for each I/O register
- Pointers to this structure have to be declared as volatile
- Example:

```
volatile avr32_pio_t *pioc = &AVR32_PIOC;
pioc->per = 0xFF;
pioc->oer = 0xFF;
pioc->codr = 0xAA;
pioc->sodr = 0x55;
```

NTNU – Trondheim
Norwegian University of
Science and Technology

Interrupts in C

- Interrupt routine: __int_handler *interruptroutine(void);
- Setting up the interrupts:

```
#include <sys/interrupts.h>
set_interrupts_base((void*)AVR32_INTC_ADDRESS);
register_interrupt(handler, group, line, priority);
init_interrupts();
```

Interrupt: example

```
#include <sys/interrupts.h>
#include <avr32/ap7000.h>
__int_handler *dac_int_handler(void) {
/* Process interrupt */
return 0;
}
int main() {
set_interrupts_base((void*)AVR32_INTC_ADDRESS);
register_interrupt((__int_handler)(dac_int_handler),
AVR32_DAC_IRQ/32, AVR32_DAC_IRQ%32, INT0);
init_interrupts();
/* ... */
}
```

NTNU – Trondheim
Norwegian University of
Science and Technology

DAC (Digital to Analog Converter)

- Built-in DAC in the microcontroller
- Uses a clock to generate interrupts regularly
- An interrupt routine is feeding the DAC with audio samples
- DAC shares some pins with PIOB; PIOB must be set up to allow this

Setup DAC

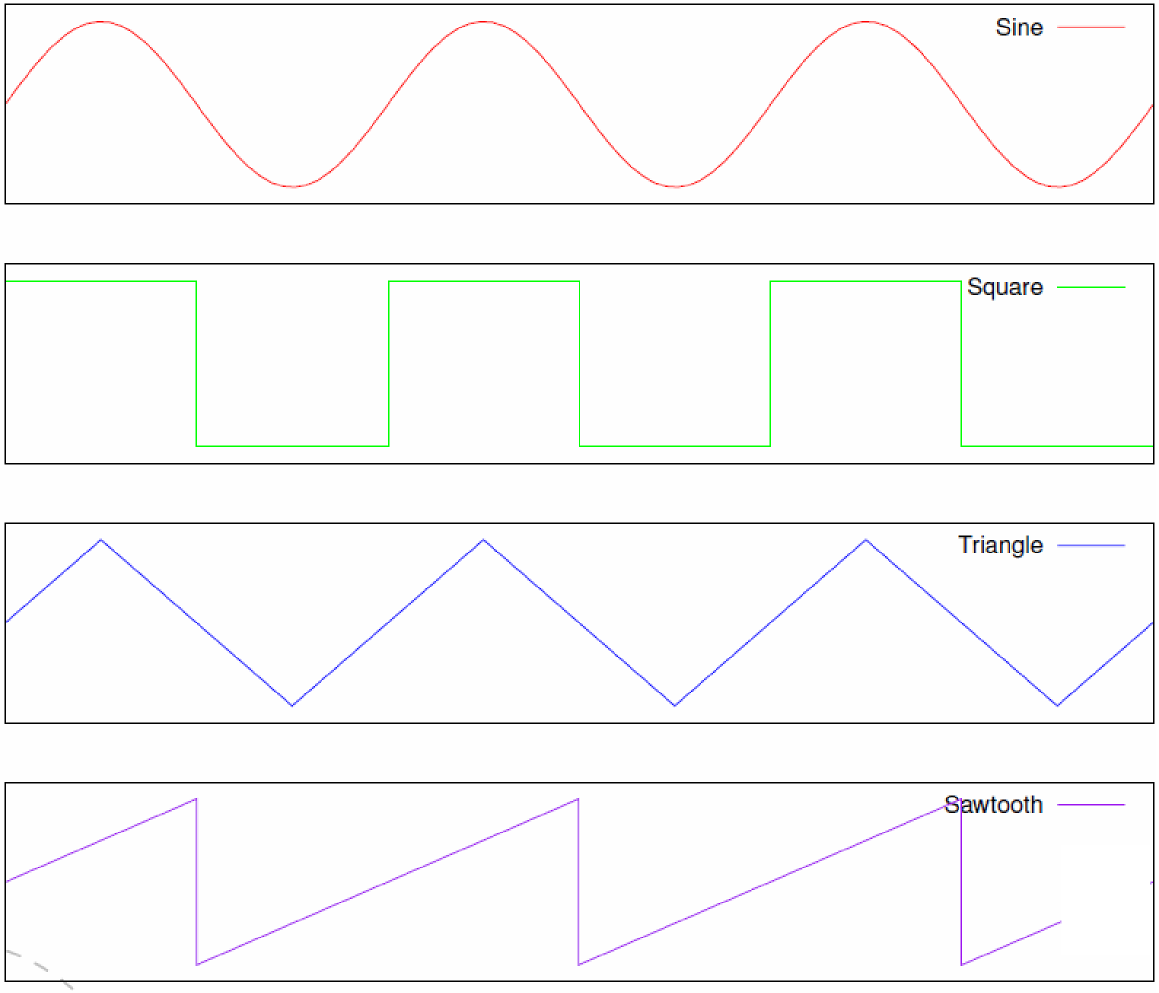
- Set bit 20 and 21 in PIOBs PDR (PIO disable) and ASR (peripherial A select)
- Set up clock:

```
volatile avr32_pm_t *sm = &AVR32_PM;
sm->gcctrl[6] = <value>;
```

- Turn on the DAC and interrupts:

```
volatile avr32_abdac_t *dac = &AVR32_ABDAC;
dac->CR.en = 1;
dac->IER.tx_ready = 1;
```

Sound waves



Recommended actions

- Start early
- Start with the given code, write Makefile
- Solve exercise 1 in C
- Try to send noise to DAC (use Rand()) to see if you have done the set up correctly
- Code reuse: split it into separate functions, the code which addresses HW should be in a separate function (helpful for Exercise 3)

Short C course now