MSP430 Family

October 23, 2016, Bulat Valeev

Lection 3. API writing.

Challenges

Code abstraction levels

Why API?

Task in the class: UART

Hometask

Challenges

What you should know at the end of the day.

- Learn how write the API for hardware handling
- Understand which functions better to implement if you need to simplify main program
- Study ADC and UART work

Code abstraction levels

The developers write a lot of the code and use number of different MCU families.
Even same MCU family change register addresses and use specific implementations for different hardware.
Any change in the MCU family leads to the code correction.
Code abstraction allows save main principles and change only low level functions.

Code abstraction levels

There are number of abstraction levels:

- Address layer
- Standard peripheral layer
- Hardware abstract layer
- API layer
- Kernel layer

Code abstraction levels

Address layer usually used in the Assembler language and assume the clear work with addresses for registers. It is very powerful,but inflexible way to write a program.
Standard peripheral layer introduce the registers in the clear for developer form. The developer still should write values in the registers.

Code abstraction levels

Hardware abstract layer allows to create abstract structure with well explained values corresponding to the certain hardware and initialize it once.
API introduce well explained sub-functions which allows to be flexible in the MCU family and save software structure with migration from MCU to MCU.

Why better to use API

It is important to write clear code.
The API simplify the program and make it clear.
The developer must write the code which is understandable for others and make code easy to debug.
You must understand your code after 10 month break.

Code comparison

```
// Fast code here
P1DIR |= RXLED + TXLED;
UCA0CTL1 |= UCSSEL_2;
UCA0CTL1 |= UCSWRST;
UCA0BR0=0x68;
UCA0BR1 = 0x00;
UCA0CTL1 &= ~UCSWRST;
// Clear code here
Clock_Initialization();
GPIO_configuration();
UART_configuration();
ADC_configuration();
```

Why better to use API

The price for clear code is overhead.
The written in the API code have big overhead in the reliability, because it will be used in the different situations and developer must be sure in the code.

Place of the API

API implemented as the low level library which perform only basic operations.
You also can use the flexibility in the IC family inside of the API
Example:

```
int UART_receive_byte(voids) {
int recv;
if (chip=='ATmega8'){
        recv=UDR0;
}
if (chip=='ATmega16'){
        recv=UDR;
}
return recv;
}
```

API for UART

Which API better to write for UART?

- Initialize
- Set baud-rate
- Receive string
- Send byte
- Send string

What should not be done: Control the data integrity

Task in the class

You must implement UART API for the project lab 2 in the eclipse workspace.

Hint: You can use interrupts and global variables.

Use initial baudrate = 9600

UART interrupts

UART has four interrupts:

- Received byte
- Sent byte
- Buffer is empty
- There is an error

Initialize

Set baud-rate

Receive string

Send byte

Send string

Hometask ADC API

Write API for ADC

- Initialize
- Change channel
- Get conversion sample (without interrupt)
- Change ADC window

Hint:The task "Get conversion sample" should calculate mean value over N samples.

Thanks for your attention