Казанский национальный исследовательский технический университет им. А.Н. Туполева

MSP430 Family

October 23, 2016, Bulat Valeev

Lection 2. *C* coding in the MCU. Interrupts.

Hometask

Challenges

Interrupts

Typical code structure

IDE Eclipse

Current work

Results

Hometask

Create own project

Hometask

Algorithm for the clock frequency configuration:

Configure multipliers in the BCSCTL1 and DCOCTL

Select DCO in thr BCSCTL2

Choose prescalers in the BCSCTL2

Challenges

What you should know at the end of the day.

- Learn typical structure of the program in the MCU's.
- Know how interrupts are used.
- Make UART configuration

Memory structure in the MCU

The real code in the MCU memory is structured in the presented way. MCU read instructions from up value to down.

Developer uses *goto* instructions to make software cyclic and move to the certain point in the code.

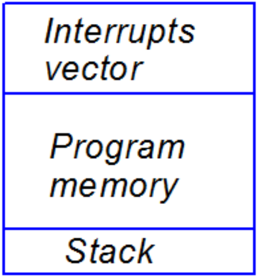


Figure: *Memory structure in firmware*

Interrupts

Interrupts in the main feature of the MCU's, which used everywhere. They allow to avoid linear code propagation and add signalling between periphery and main CPU.

Simple explanation is the following. Interrupt is the event which forces MCU go to the certain point in the code with copying the all variables in the heap.

Interrupts

Number of the periphery in the MCU raise up and CPU can no more control each event.
Periphery is written in the datasheet for each MCU.
Periphery in the MSP430G2553:

- Digitally controlled oscillator
- Flash memory
- Two Timer A timers
- Universal serial communication interface
- Comparator
- Analog-to-Digital converter
- Watchdog
- Brownout detector
- GPIO

Interrupts

Table: Interrupts in the MSP430G2553

Periphery	Number of interrupts	Interrupts
Brownout detector	2	Power-up, External Reset
Flash memory	2	Flash key violation, PC out-of-range, Flash memory access violation
Timer A	12	Timer0x6, Timer1x6
Comparator	1	CAIFG
Watchdog	1	Timer overflow
USCI	4	I2C RX/TX, UART TX/RX
GPIO	2	External interrupts
ADC	1	Measurement complete

Typical code structure

The *C* code in the all MCU's usually has the similar structure and divided in the 7 groups:

- One loop program
- One loop program with interrupts
- State machine program
- Supervisor
- Prioritized supervisor
- Cooperative RTOS
- Preemptive RTOS

More tasks in the program- more complex architecture better to use.

One loop program

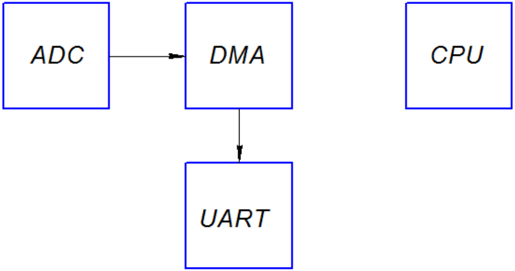
```
void Init(void){
//Here initialized periphery
}
int main(void){
Init(); /Initialization
while(1){
        \\Here starts code
        Led_toggle();
        _delay_ms(100);
        }
return;
}
```

One loop program with interrupts

```
int main(void) {
Init(); /Initialization
while(1){
        \\Here starts some code
        }
return;
}
ISR(UART_RX_vect) { // Interrupt handler
        UART_RX_flag_off();//turn off interrupt fl
        read_uart_byte(); //read received byte
return;
}
```

Interrupts example: ADC+DMA+UART

The direct memory access controller allows to transfer data directly from ADC to the UART with minimal CPU work.



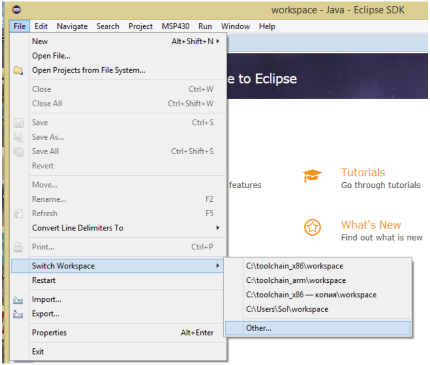
CPU load decreases significantly in this example.

State machine program

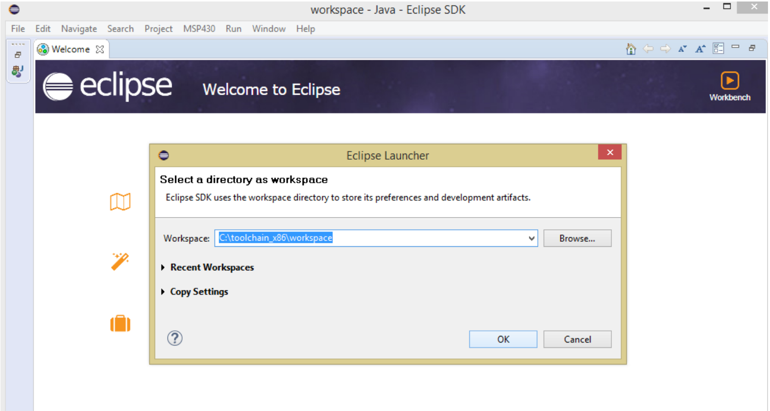
```
int main(void){
Init(); /Initialization
while(1){
        if (flag_1){
                flag_1=0;//do something
                flag_2=1;
        }
        if (flag_2){
                flag_2=0;//do something
                flag_1=1;
        }
}
return;
}
```

Start work with IDE

```
C:\toolchain_x86\launcheclipse.bat
```



Start work with IDE


workspace - Java - Eclipse SDK
File Edit Navigate Search Project MSP430 Run Window Help
Welcome
eclipse
Welcome to Eclipse
Workbench

Eclipse Launcher

Select a directory as workspace

Eclipse SDK uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\toolchain_x86\workspace

Figure: *Eclipse*

Start work with IDE

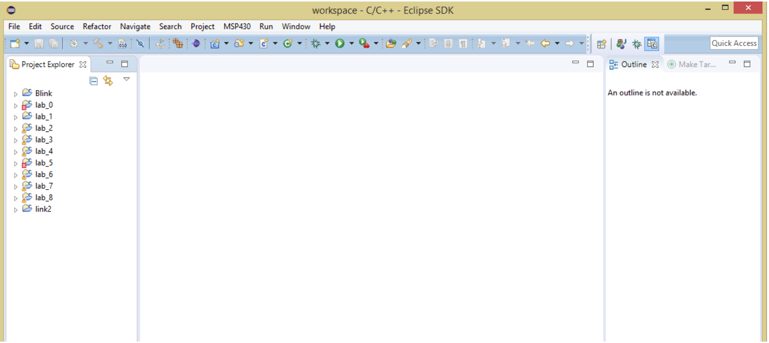
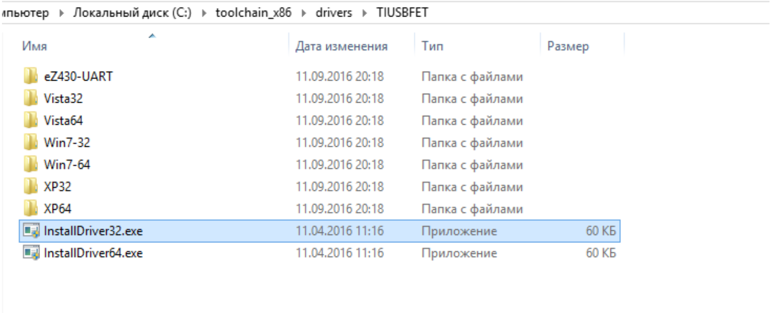

workspace - C/C++ - Eclipse SDK
File Edit Source Refactor Navigate Search Project MSP430 Run Window Help
Quick Access
Project Explorer
Blink
lab_0
lab_1
lab_2
lab_3
lab_4
lab_5
lab_6
lab_7
lab_8
link2
Outline
Make Tar...
An outline is not available.

Figure: *Eclipse*

Launchpad drivers install

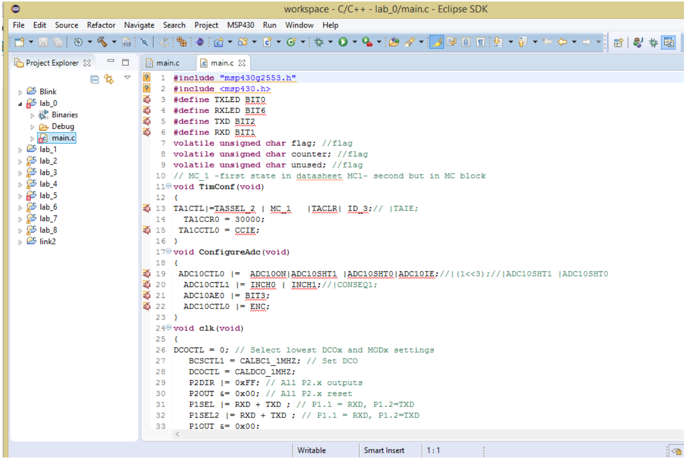


мпьютер ▸ Локальный диск (C:) ▸ toolchain_x86 ▸ drivers ▸ TIUSBFET

Имя	Дата изменения	Тип	Размер
eZ430-UART	11.09.2016 20:18	Папка с файлами	
Vista32	11.09.2016 20:18	Папка с файлами	
Vista64	11.09.2016 20:18	Папка с файлами	
Win7-32	11.09.2016 20:18	Папка с файлами	
Win7-64	11.09.2016 20:18	Папка с файлами	
XP32	11.09.2016 20:18	Папка с файлами	
XP64	11.09.2016 20:18	Папка с файлами	
InstallDriver32.exe	11.04.2016 11:16	Приложение	60 КБ
InstallDriver64.exe	11.04.2016 11:16	Приложение	60 КБ

Figure: *Eclipse*

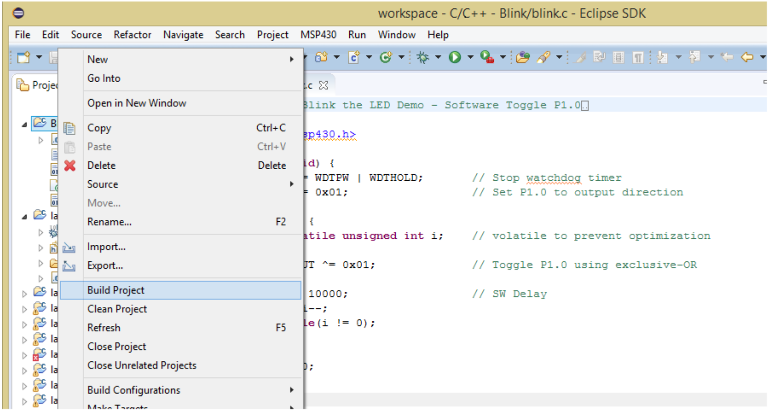
Project lab 0



```c
#include "msp430g2553.h"
#include <msp430.h>
#define TXLED BIT0
#define RXLED BIT6
#define TXD BIT2
#define RXD BIT1
volatile unsigned char flag; //flag
volatile unsigned char counter; //flag
volatile unsigned char unused; //flag
// MC_1 -first state in datasheet MC1- second but in MC block
void TimConf(void)
{
TA1CTL|=TASSEL_2 | MC_1   |TACLR| ID_3;// |TAIE;
  TA1CCR0 = 30000;
 TA1CCTL0 = CCIE;
}
void ConfigureAdc(void)
{
 ADC10CTL0 |=  ADC10ON|ADC10SHT1 |ADC10SHT0|ADC10IE;//|(1<<3);//|ADC10SHT1 |ADC10SHT0
  ADC10CTL1 |= INCH0 | INCH1;//|CONSEQ1;
  ADC10AE0 |= BIT3;
  ADC10CTL0 |= ENC;
}
void clk(void)
{
DCOCTL = 0; // Select lowest DCOx and MODx settings
   BCSCTL1 = CALBC1_1MHZ; // Set DCO
   DCOCTL = CALDCO_1MHZ;
   P2DIR |= 0xFF; // All P2.x outputs
   P2OUT &= 0x00; // All P2.x reset
   P1SEL |= RXD + TXD ; // P1.1 = RXD, P1.2=TXD
   P1SEL2 |= RXD + TXD ; // P1.1 = RXD, P1.2=TXD
   P1OUT &= 0x00;
```

Figure: *Eclipse*

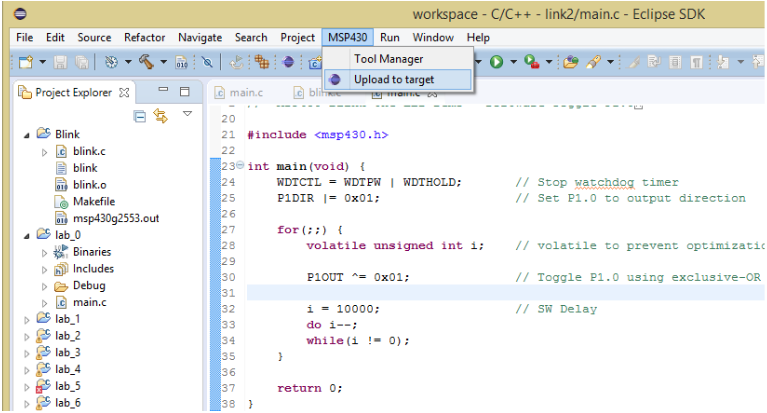
Compilation of the project



```
Blink the LED Demo - Software Toggle P1.0

sp430.h>

id) {
= WDTPW | WDTHOLD;          // Stop watchdog timer
= 0x01;                     // Set P1.0 to output direction

 {
atile unsigned int i;       // volatile to prevent optimization

UT ^= 0x01;                 // Toggle P1.0 using exclusive-OR

 10000;                     // SW Delay
i--;
le(i != 0);

);
```

Figure: *Eclipse*

Upload on chip



```
#include <msp430.h>

int main(void) {
    WDTCTL = WDTPW | WDTHOLD;          // Stop watchdog timer
    P1DIR |= 0x01;                     // Set P1.0 to output direction

    for(;;) {
        volatile unsigned int i;       // volatile to prevent optimizatic

        P1OUT ^= 0x01;                 // Toggle P1.0 using exclusive-OR

        i = 10000;                     // SW Delay
        do i--;
        while(i != 0);
    }

    return 0;
}
```

Figure: *Eclipse*

Previous UART algorithm

The resulting algorithm to configure the UART is presented here

- Turn off the UART
- Set the baud-rate
- Set the low frequency baud-rate settings
- Configure GPIO pins
- Turn on UART
- Turn on RX interrupt

Previous UART algorithm

The resulting algorithm to configure the UART is presented here

- Set the UCSWRST bit
- Configure the UCA0BR0 and UCA0BR1 bytes
- Configure the UCA0MCTL byte
- Configure P1SEL and P2SEL bytes
- Reset the UCSWRST bit
- Set the UCA0RXIE bit

Coding work: UART

Put the code in the project of the lab 1 main.c file

```
void uart_conf(void) //uart initialization
{
    // Your code should be here
    return;
}
```

Result code

Here presented answer code

```c
void uart_conf(void){ //uart initialization
UCA0CTL1 |= UCSWRST;//Turn of the UART
UCA0BR0=0x68;//Set the baud-rate
UCA0BR1 = 0x00; //Set the baud-rate
UCA0MCTL = UCBRS2 + UCBRS0;
//Set the low frequency baud-rate settings
P1SEL |= RXD + TXD ; // Configure P1SEL and P2SEL
P1SEL2 |= RXD + TXD ;
UCA0CTL1 &= ~UCSWRST; //Turn on UART
UC0IE |= UCA0RXIE; // Turn on RX interrupt
return;
}
```

Task: UART Echo

Write the program which will loopback each received byte via UART.
Use baudrate 9600.
Write code in the lab 1-Hometask project.
Deadline is xx.xx.xxxx.

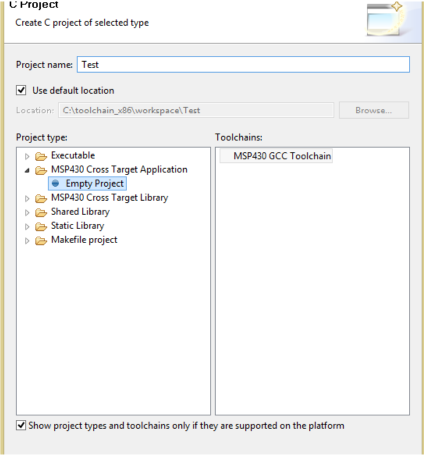
Algorithm of the solution

The main program structure is next:

- Initialize DCO
- Initialize GPIO
- Initialize UART
- Start infinite loop in the main cycle
- Send each symbol back in the interrupt cycle

Thanks for your attention

Create own project



C Project

Create C project of selected type

Project name: Test

☑ Use default location

Location: C:\toolchain_x86\workspace\Test

Browse...

Project type:

- Executable
- MSP430 Cross Target Application
  - Empty Project
- MSP430 Cross Target Library
- Shared Library
- Static Library
- Makefile project

Toolchains:

MSP430 GCC Toolchain

☑ Show project types and toolchains only if they are supported on the platform

Figure: *Eclipse*

Create own project

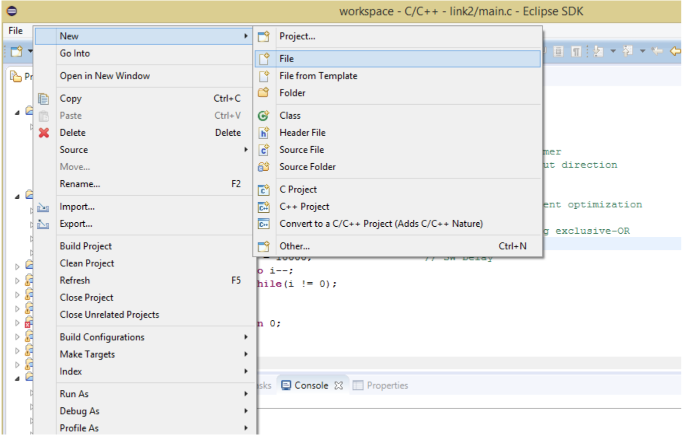
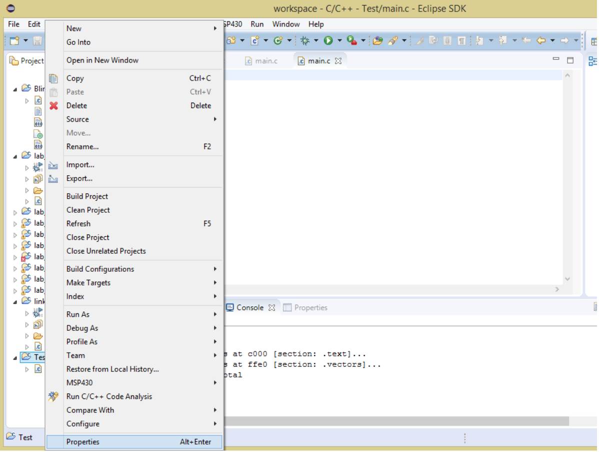


Figure: *Eclipse*

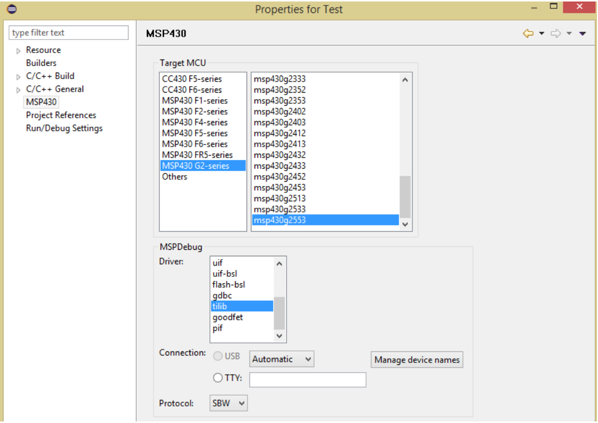
Create own project



```
s at c000 [section: .text]...
s at ffe0 [section: .vectors]...
otal
```

Figure: *Eclipse*

Create own project



MSP430

Target MCU: MSP430 G2-series, msp430g2553

MSPDebug

Driver: tilib

Connection: ○ USB Automatic

○ TTY:

Protocol: SBW

Figure: *Eclipse*

Reference slide

- http://easyelectronics.ru/avr-uchebnyj-kurs-programmirovanie-na-si-chast-1.html
- http://easyelectronics.ru/avr-uchebnyj-kurs-programmirovanie-na-si-chast-2.html
- http://easyelectronics.ru/avr-uchebnyj-kurs-programmirovanie-na-si-chast-3.html
- http://easyelectronics.ru/avr-uchebnyj-kurs-programmirovanie-na-si-chast-4.html
- http://easyelectronics.ru/avr-uchebnyj-kurs-programmirovanie-na-si-rabota-s-pamyatyu-adresa-i-ukazateli.html

Reference slide

- http://easyelectronics.ru/avr-uchebnyj-kurs-arxitektura-programm.html
- http://easyelectronics.ru/avr-uchebnyj-kurs-konechnyj-avtomat.html
- http://easyelectronics.ru/avr-uchebnyj-kurs-vytesnyayushhij-dispetcher-zadach.html
- http://easyelectronics.ru/avr-uchebnyj-kurs-vtoraya-programma.html