MSP430 Family

October 23, 2016, Bulat Valeev

Lection 3. API writing.

Result from previous task

Challenges

Code abstraction levels

Why API?

Task in the class: UART

Hometask

Answer

```
#include "msp430g2553.h"
#define TXLED BIT0
#define RXLED BIT6
#define TXD BIT2
#define RXD BIT1
int main(void)
{
DCOCTL = 0;
BCSCTL1 = CALBC1_1MHZ;
DCOCTL = CALDCO_1MHZ;
P2DIR |= 0xFF;
P2OUT &= 0x00; t
P1SEL |= RXD + TXD ;
```

Answer

```
P1SEL2 |= RXD + TXD ;
P1OUT &= 0x00;
P1DIR |= RXLED + TXLED;
UCA0CTL1 |= UCSSEL_2;
UCA0CTL1 |= UCSWRST;
UCA0BR0=0x68;
UCA0BR1 = 0x00;
UCA0MCTL = UCBRS2 + UCBRS0;
UCA0CTL1 &= ~UCSWRST;
```

Answer

```
while(1)
{
if (IFG2&UCA0RXIFG)
{
IFG2=IFG2&(~UCA0RXIFG);
UCA0TXBUF=UCA0RXBUF;
}
}
}
```

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.

```
UCA0CTL1 |= UCSSEL_2;
```

Code abstraction levels

Standard peripheral layer introduce the registers in the clear for developer form. The developer still should write values in the registers.
C code which you will write will usually written in this level.

```
GPIOB->CRL      |= GPIO_CRL_MODE5_0;
```

Code abstraction levels

Hardware abstract layer allows to create abstract structure with well explained values corresponding to the certain hardware and initialize it once.
The HAL is initialization of the periphery with objects with close connection to the objective oriented programming.

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

Code abstraction levels

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.
API will be the main level of your algorithm construction.

```
GPIO_Init();
```

Why better to use API

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

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(void) {
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 in API: Control the data integrity, Mutex

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

You will use three of them: Received byte,Sent byte ,Buffer is empty.

Datasheet

To learn about UART you can use the MSP43xGxxx Family user guide. The section USCI UART has necessary information, especially registers description

Technical pages for the UART

15.4.2 UCAxCTL1, USCI_Ax Control Register 1

7	6	5	4	3	2	1	0
UCSSELx		UCRXEIE	UCBRKIE	UCDORM	UCTXADDR	UCTXBRK	UCSWRST
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

UCSSELx Bits 7-6 USCI clock source select. These bits select the BRCLK source clock.
- 00 UCLK
- 01 ACLK
- 10 SMCLK
- 11 SMCLK

UCRXEIE Bit 5 Receive erroneous-character interrupt-enable
- 0 Erroneous characters rejected and UCAxRXIFG is not set
- 1 Erroneous characters received will set UCAxRXIFG

UCBRKIE Bit 4 Receive break character interrupt-enable
- 0 Received break characters do not set UCAxRXIFG.
- 1 Received break characters set UCAxRXIFG.

UCDORM Bit 3 Dormant. Puts USCI into sleep mode.
- 0 Not dormant. All received characters will set UCAxRXIFG.
- 1 Dormant. Only characters that are preceded by an idle-line or with address bit set will set UCAxRXIFG. In UART mode with automatic baud rate detection only the combination of a break and synch field will set UCAxRXIFG.

UCTXADDR Bit 2 Transmit address. Next frame to be transmitted will be marked as address depending on the selected multiprocessor mode.
- 0 Next frame transmitted is data
- 1 Next frame transmitted is an address

UCTXBRK Bit 1 Transmit break. Transmits a break with the next write to the transmit buffer. In UART mode with automatic baud rate detection 055h must be written into UCAxTXBUF to generate the required break/synch fields. Otherwise 0h must be written into the transmit buffer.
- 0 Next frame transmitted is not a break
- 1 Next frame transmitted is a break or a break/synch

UCSWRST Bit 0 Software reset enable
- 0 Disabled. USCI reset released for operation.
- 1 Enabled. USCI logic held in reset state.

Technical pages for the UART

15.4.6 UCAxSTAT, USCI_Ax Status Register

7	6	5	4	3	2	1	0
UCLISTEN	UCFE	UCOE	UCPE	UCBRK	UCRXERR	UCADDR UCIDLE	UCBUSY
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	r-0

UCLISTEN Bit 7 — Listen enable. The UCLISTEN bit selects loopback mode.
- 0 Disabled
- 1 Enabled. UCAxTXD is internally fed back to the receiver.

UCFE Bit 6 — Framing error flag
- 0 No error
- 1 Character received with low stop bit

UCOE Bit 5 — Overrun error flag. This bit is set when a character is transferred into UCAxRXBUF before the previous character was read. UCOE is cleared automatically when UCxRXBUF is read, and must not be cleared by software. Otherwise, it will not function correctly.
- 0 No error
- 1 Overrun error occurred

UCPE Bit 4 — Parity error flag. When UCPEN = 0, UCPE is read as 0.
- 0 No error
- 1 Character received with parity error

UCBRK Bit 3 — Break detect flag
- 0 No break condition
- 1 Break condition occurred

UCRXERR Bit 2 — Receive error flag. This bit indicates a character was received with error(s). When UCRXERR = 1, on or more error flags (UCFE, UCPE, UCOE) is also set. UCRXERR is cleared when UCAxRXBUF is read.
- 0 No receive errors detected
- 1 Receive error detected

UCADDR Bit 1 — Address received in address-bit multiprocessor mode.
- 0 Received character is data
- 1 Received character is an address

UCIDLE — Idle line detected in idle-line multiprocessor mode.
- 0 No idle line detected
- 1 Idle line detected

UCBUSY Bit 0 — USCI busy. This bit indicates if a transmit or receive operation is in progress.
- 0 USCI inactive
- 1 USCI transmitting or receiving

Technical pages for the UART

15.4.1 UCAxCTL0, USCI_Ax Control Register 0

7	6	5	4	3	2	1	0
UCPEN	UCPAR	UCMSB	UC7BIT	UCSPB	UCMODEx		UCSYNC
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

UCPEN	Bit 7	Parity enable
	0	Parity disabled.
	1	Parity enabled. Parity bit is generated (UCAxTXD) and expected (UCAxRXD). In address-bit multiprocessor mode, the address bit is included in the parity calculation.
UCPAR	Bit 6	Parity select. UCPAR is not used when parity is disabled.
	0	Odd parity
	1	Even parity
UCMSB	Bit 5	MSB first select. Controls the direction of the receive and transmit shift register.
	0	LSB first
	1	MSB first
UC7BIT	Bit 4	Character length. Selects 7-bit or 8-bit character length.
	0	8-bit data
	1	7-bit data
UCSPB	Bit 3	Stop bit select. Number of stop bits.
	0	One stop bit
	1	Two stop bits
UCMODEx	Bits 2-1	USCI mode. The UCMODEx bits select the asynchronous mode when UCSYNC = 0.
	00	UART mode
	01	Idle-line multiprocessor mode
	10	Address-bit multiprocessor mode
	11	UART mode with automatic baud rate detection
UCSYNC	Bit 0	Synchronous mode enable
	0	Asynchronous mode
	1	Synchronous mode

Initialize

```
void UART_INIT(void)
{
UCA0CTL1 |= UCSWRST;
UCA0BR0=0x68;
UCA0BR1 = 0x00; // 1MHz 115200
UCA0MCTL = UCBRS2 + UCBRS0;
UCA0CTL1 &= ~UCSWRST; UC0IE |= UCA0RXIE;
}
```

Set baud-rate

```
void UART_INIT(int byte1,int byte2)
{
UCA0CTL1 |= UCSWRST;
UCA0BR0=byte2;
UCA0BR1 = byte1;
UCA0CTL1 &= ~UCSWRST;
}
```

Receive string

```
char buffer[32];
int counter=0;
char flag=0;
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
        if (UCA0RXBUF!=0x00){
        buffer[counter]=UCA0RXBUF;
        counter++;
        }
        else {
        counter=0;
        flag=1;
        }
}
```

Send byte

```
coid UART_send_byte(int data) {
UCA0TXBUF = data;
return
}
```

Send string

```
void UART_send_string(int string,int length) {
int len;
len=length;
while (len){
if (IFG2&UCA0TXIFG) {
IFG2=IFG2&(~UCA0TXIFG);
UCA0TXBUF = string[len--];
}
}
}
```

Hometask ADC API

Write API for ADC

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

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

Deadline is xx.xx.xxxx.

Thanks for your attention

Reference slide

- https://www.drive2.ru/b/2253235/
- http://easyelectronics.ru/rabota-s-stm32f10x-standard-peripherals-library.html
- https://habrahabr.ru/post/249395/
- http://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-configurators-and-code-generators/stm32cubemx.html