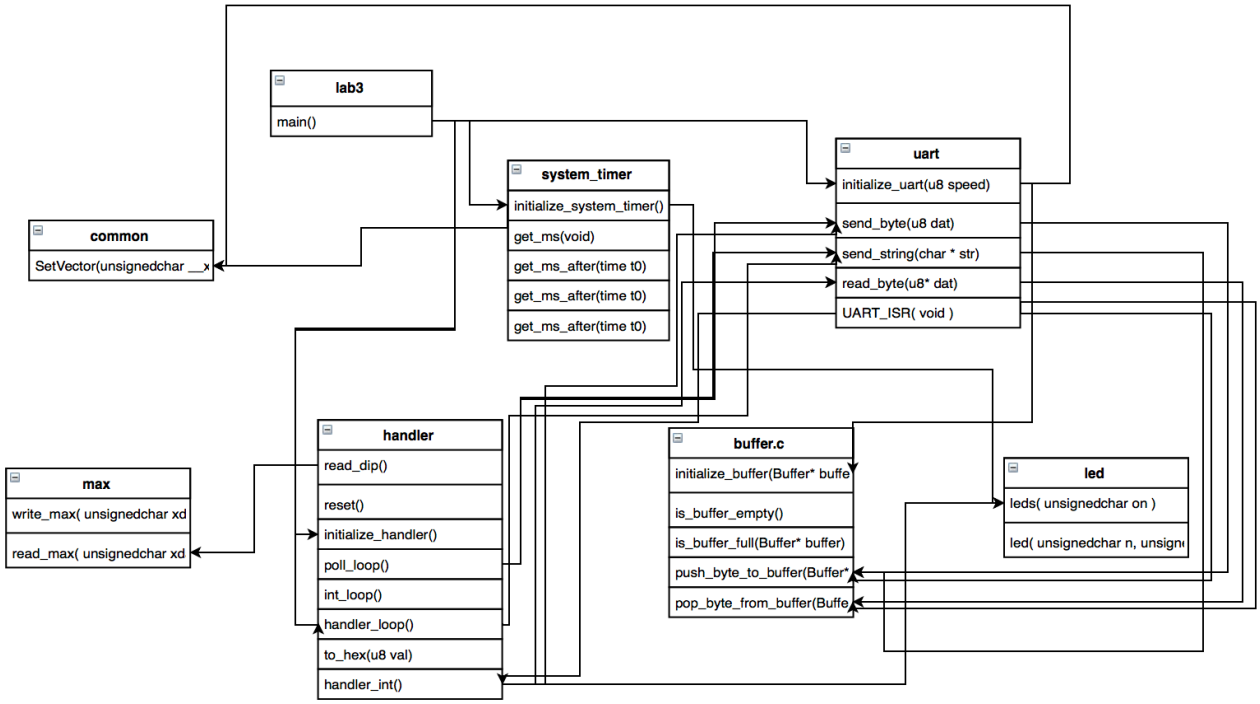
Университет ИТМО
Кафедра ВТ

Лабораторная работа по предмету:
ИУС
Лабораторная работа №3

Выполнили:
Гулямова С. И.
Разумовская А. В.
Шляков А. К.

Санкт - Петербург
2017г.

Диаграмма взаимодействия модулей:



buffer.c

```
void initialize_buffer(Buffer* buffer) {
	buffer->head=0;
	buffer->tail=0;
	buffer->len=0;
}


bool is_buffer_empty(Buffer* buffer) {
	return buffer->len == 0;
}

bool is_buffer_full(Buffer* buffer) {
	return buffer->len == BUFFER_LEN;
}

void push_byte_to_buffer(Buffer* buffer, u8 dat) {

	buffer->mem[buffer->head] = dat;
	buffer->head++;
	buffer->len++;
    if (buffer->head == BUFFER_LEN) {
		buffer->head = 0;
    }

}

u8 pop_byte_from_buffer(Buffer* buffer) {
	u8 dat;
	dat= buffer->mem[buffer->tail];
	buffer->tail++;
	buffer->len--;

	if (buffer->tail == BUFFER_LEN) {
		buffer->tail = 0;
	}

	return dat;

}
```

common.c

```
void SetVector(unsigned char __xdata * Address, void * Vector)
{

	unsigned char __xdata * TmpVector;
	// Первым байтом по указанному адресу записывается
	// код команды передачи управления ljmp, равный 02h
```

```
	*Address = 0x02;

	// Далее записывается адрес перехода Vector
	TmpVector = (unsigned char __xdata *) (Address + 1);
	*TmpVector = (unsigned char) ((unsigned short)Vector >> 8);
	++TmpVector;
	*TmpVector = (unsigned char) Vector;

	// Таким образом, по адресу Address теперь
	// располагается инструкция ljmp Vector

}

handler.c
u8 read_dip(){
	return read_max(EXT_LO);
}

void reset() {
	number=0;
	state=STATE_NUMBER;
}


void initialize_handler() {
	mode = MODE_POLL;
	reset();
}

void poll_loop() {
	u8 i;
	u8 byte_in;

	while( read_dip()==DIP_POLL_MODE ){
		if( read_byte(&byte_in) ){
			for( i=byte_in;i<='9';i++ ){
				send_byte(i);
			}
			send_string("\r\n");
		}
		delay_ms(1);
	}
	mode=MODE_INT;
}

void int_loop() {
	while( read_dip()!=DIP_POLL_MODE ){
		delay_ms(1);
	}
	mode=MODE_POLL;
}
```

```
void handler_loop() {
	while(1) {
		if( mode==MODE_POLL ) {
			send_string("\r\npoll mode\r\n");
			poll_loop();
		}else{
			send_string("\r\ninteruption mode\r\n");
			int_loop();
		}
	}
}

void error() {
	send_string("\r\nerror\r\n");
	leds(0xAA);
	state=STATE_ERROR;
}

u8 to_hex(u8 val) {
	if( val>9 ) {
		return 'A'+val-10;
	}
	return '0'+val;
}

void handler_int() {
	u8 num;
	u8 sym;

	if( !read_byte(&sym) ){
		error();
		return;
	}

	if( state==STATE_ERROR ){//î÷èùàåì ïîñëå îøèáêè
		reset();
		leds(0x00);
	}

	switch (state) {
		case STATE_NUMBER:
			if(sym>='0' && sym<='9'){
				send_byte(sym);
				num=sym-'0';



if( num > NUMBER_LIMIT-number*10 ) {
				error();
					return;
				}
				number*=10;
```

```
                        number+=num;
                  } else if( sym=='r' ) {
                        state=STATE_CR;
                  }else{
                        error();
                  }
                  break;
            case STATE_CR:
                  if( sym=='n' ) {
                        send_string("\r\nHex:");
                        send_byte(to_hex(number>>4));
                        send_byte(to_hex(number&0x0F));
                        send_string("\r\n");
                        reset();
                  }else{
                        error();
                  }
                  break;
      }
}
```

lab3.c

```
void delay ( unsigned long ms );

void main( void ) {
      initialize_system_timer();
      initialize_uart(S9600);
      initialize_handler();
      EA=1;
      handler_loop();
}
```

led.c

```
void leds( unsigned char on )
{
    write_max( SV, on );
    old_led = on;

}
```

max.c

```
void write_max( unsigned char xdata *regnum, unsigned char val )
{
unsigned char oldDPP = DPP;
    DPP     = MAXBASE;
    *regnum = val;
    DPP     = oldDPP;
}
```

uart.c

```
void UART_ISR( void ) __interrupt ( 4 );
```

```
void initialize_uart(u8 speed) {
      initialize_buffer(&buffer_in);
      initialize_buffer(&buffer_out);
      SetVector( 0x2023, (void *)UART_ISR );
      TH1 =  speed;
    TMOD     |=  0x20;
    TCON     |=  0x40;
    SCON      =  0x50;
    ES=1;

}

void send_byte(u8 dat) {
      ES=0;
      if( !sending_byte ){
            sending_byte=true;
            SBUF=dat;
      }else if( !is_buffer_full(&buffer_out) ){
            push_byte_to_buffer(&buffer_out,dat);
      }
      ES=1;
}

void send_string(char * str){
      ES=0;
      if( !sending_byte ){
            sending_byte=true;
            SBUF=*str;
            str++;
      }

      while( *str ) {
            if( !is_buffer_full(&buffer_out) ){
                  push_byte_to_buffer(&buffer_out,*str);
                  str++;
            }
      }
      ES=1;
}

bool read_byte(u8* dat) {
      bool is_data;
      ES=0;
      is_data=!is_buffer_empty(&buffer_in);
      if( is_data ){
            *dat=pop_byte_from_buffer(&buffer_in);
      }
      ES=1;
      return is_data;
}

void UART_ISR( void ) __interrupt ( 4 ) {
```

```
	u8 dat;
	if( TI ){
		if( is_buffer_empty(&buffer_out) ){
			sending_byte=false;
		}else{
			dat=pop_byte_from_buffer(&buffer_out);
			SBUF=dat;
		}
		TI=0;
	}

	if( RI ){
		RI=0;
		dat=SBUF;
		if( !is_buffer_full(&buffer_in) ){
			push_byte_to_buffer(&buffer_in,dat);
		}
		if( mode==MODE_INT ) {
			handler_int();
		}
	}
}
```