Микропроцессорные устройства

Лекция 5

Микроконтроллеры серии AVR

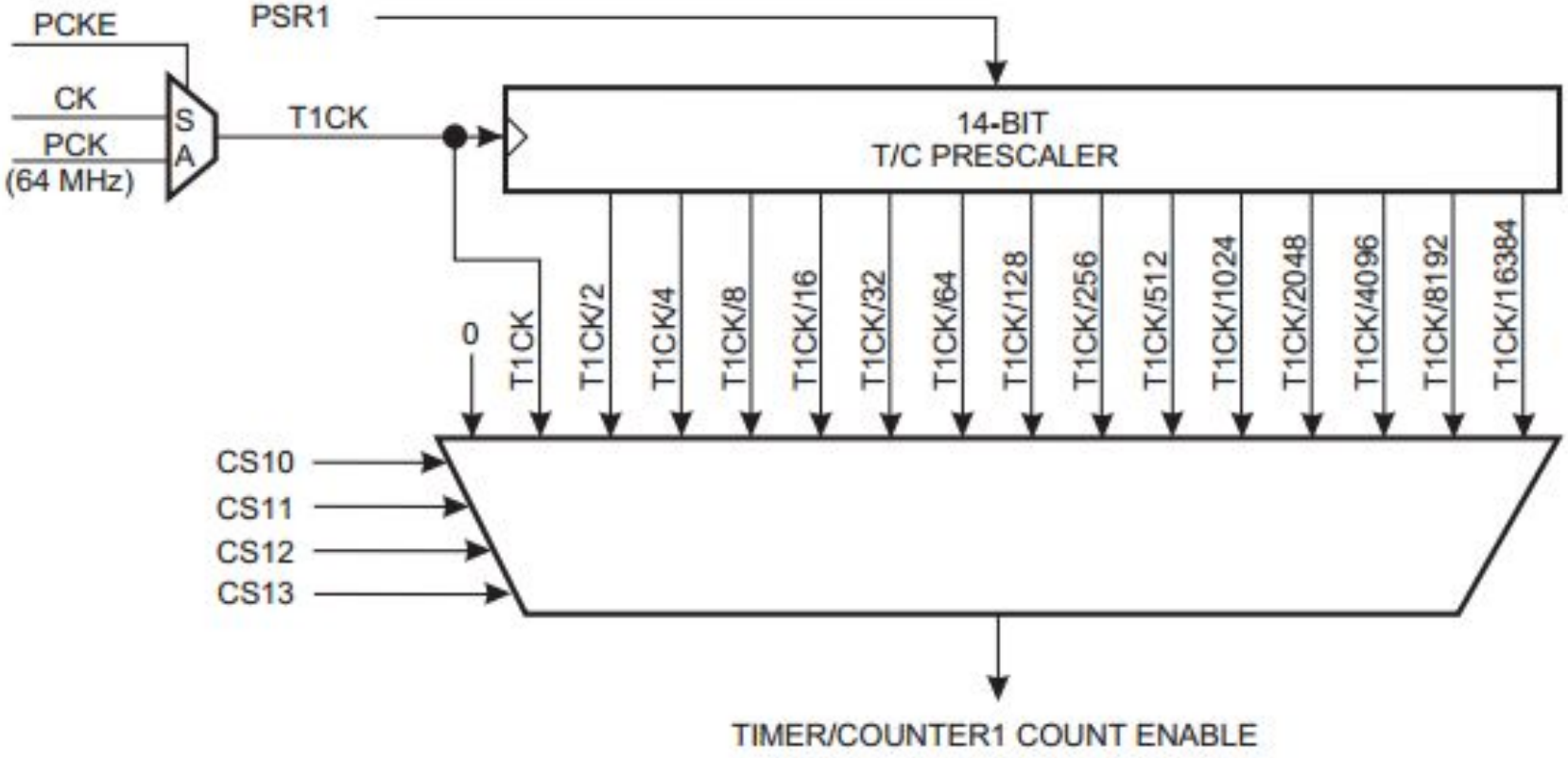
Микроконтроллер ATtiny26 (Векторы прерываний)

Vector No	Program Address	Source	Interrupt Definition
1	$000	RESET	Hardware Pin and Watchdog Reset
2	$001	INT0	External Interrupt Request 0
3	$002	I/O Pins	Pin Change Interrupt
4	$003	TIMER1, CMPA	Timer/Counter1 Compare Match 1A
5	$004	TIMER1, CMPB	Timer/Counter1 Compare Match 1B
6	$005	TIMER1, OVF1	Timer/Counter1 Overflow
7	$006	TIMER0, OVF0	Timer/Counter0 Overflow
8	$007	USI_STRT	USI Start
9	$008	USI_OVF	USI Overflow
A	$009	EE_RDY	EEPROM Ready
B	$00A	ANA_COMP	Analog Comparator
C	$00B	ADC	ADC Conversion Complete

Микроконтроллер ATtiny26 (Векторы прерываний)

```
$000    rjmp    RESET          ; Reset handler
$001    rjmp    EXT_INT0       ; IRQ0 handler
$002    rjmp    PIN_CHANGE     ; Pin change handler
$003    rjmp    TIM1_CMP1A     ; Timer1 compare match 1A
$004    rjmp    TIM1_CMP1B     ; Timer1 compare match 1B
$005    rjmp    TIM1_OVF       ; Timer1 overflow handler
$006    rjmp    TIM0_OVF       ; Timer0 overflow handler
$007    rjmp    USI_STRT       ; USI Start handler
$008    rjmp    USI_OVF        ; USI Overflow handler
$009    rjmp    EE_RDY         ; EEPROM Ready handler
$00A    rjmp    ANA_COMP       ; Analog Comparator handler
$00B    rjmp    ADC            ; ADC Conversion Handler
```

Микроконтроллер ATtiny26 (Таймер 1)



Микроконтроллер ATtiny26 (Таймер 1)

Bit	7	6	5	4	3	2	1	0	
$2E ($4E)	MSB							LSB	TCNT1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
$38 ($58)	–	OCF1A	OCF1B	–	–	TOV1	TOV0	–	TIFR
Read/Write	R	R/W	R/W	R	R	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
$39 ($59)	–	OCIE1A	OCIE1B	–	–	TOIE1	TOIE0	–	TIMSK
Read/Write	R	R/W	R/W	R	R	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
$30 ($50)	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	PWM1A	PWM1B	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
$2F ($4F)	CTC1	PSR1	–	–	CS13	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Микроконтроллер ATtiny26 (Таймер 1)

Bit	7	6	5	4	3	2	1	0	
$2F ($4F)	CTC1	PSR1	–	–	CS13	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

CS13	CS12	CS11	CS10	Description Asynchronous Mode	Description Synchronous Mode
0	0	0	0	Timer/Counter1 is stopped.	Timer/Counter1 is stopped.
0	0	0	1	PCK	CK
0	0	1	0	PCK/2	CK/2
0	0	1	1	PCK/4	CK/4
0	1	0	0	PCK/8	CK/8
0	1	0	1	PCK/16	CK/16
0	1	1	0	PCK/32	CK/32
0	1	1	1	PCK/64	CK/64
1	0	0	0	PCK/128	CK/128
1	0	0	1	PCK/256	CK/256
1	0	1	0	PCK/512	CK/512
1	0	1	1	PCK/1024	CK/1024
1	1	0	0	PCK/2048	CK/2048
1	1	0	1	PCK/4096	CK/4096
1	1	1	0	PCK/8192	CK/8192
1	1	1	1	PCK/16384	CK/16384

Микроконтроллер ATtiny26 (Таймер 1)

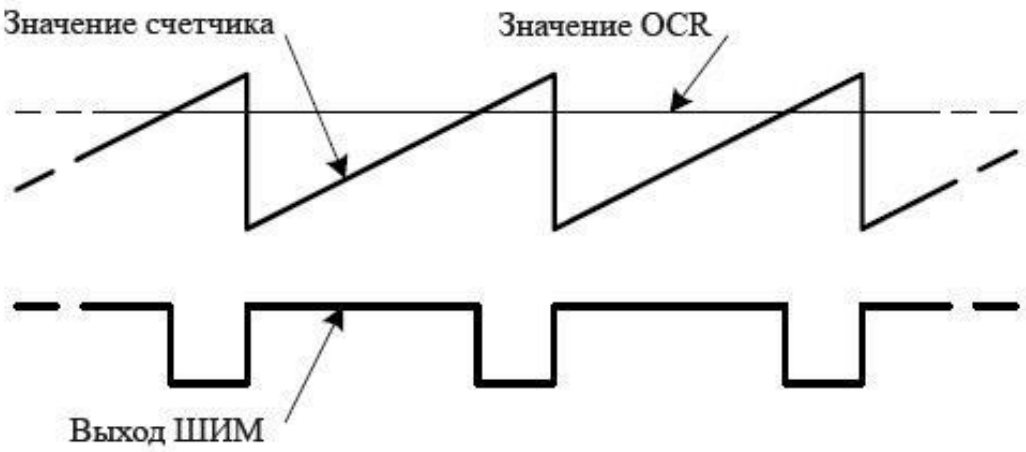
Bit	7	6	5	4	3	2	1	0	
$2D ($4D)	MSB							LSB	OCR1A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
$2C ($4C)	MSB							LSB	OCR1B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
$2B ($4B)	MSB							LSB	OCR1C
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
$29 ($29)	–	–	–	–	–	PCKE	PLLE	PLOCK	PLLCSR
Read/Write	R	R	R	R	R	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0/1	0	

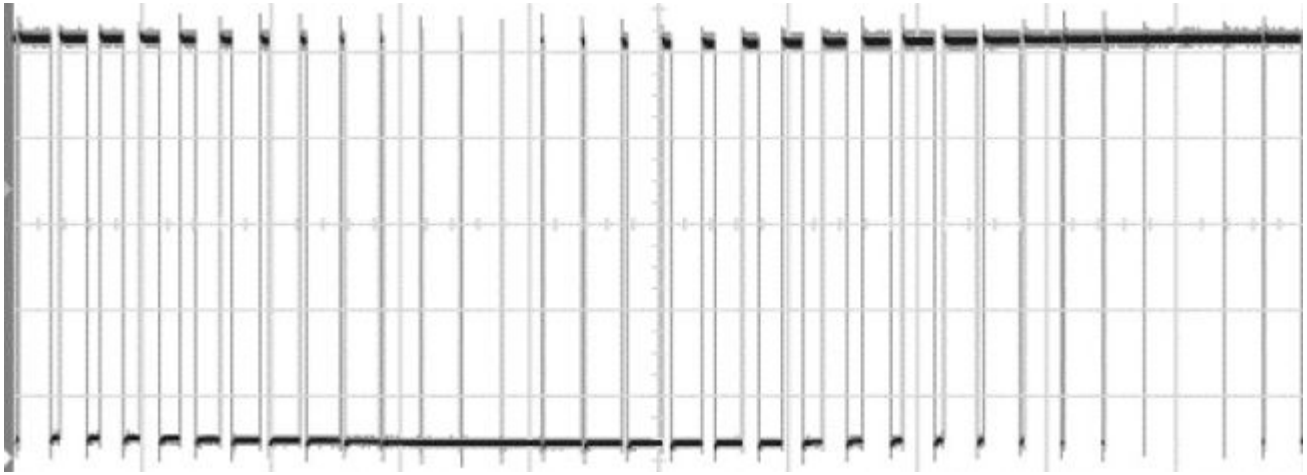
Микроконтроллер ATtiny26 (Таймер 1)

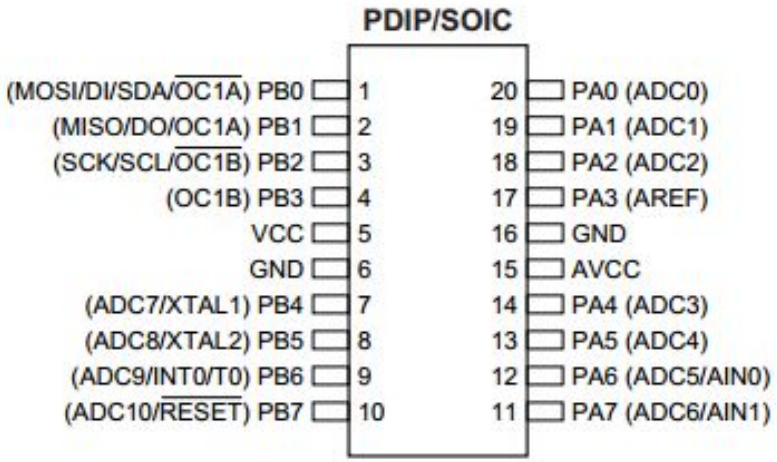


Bit	7	6	5	4	3	2	1	0	
$2D ($4D)	MSB							LSB	OCR1A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

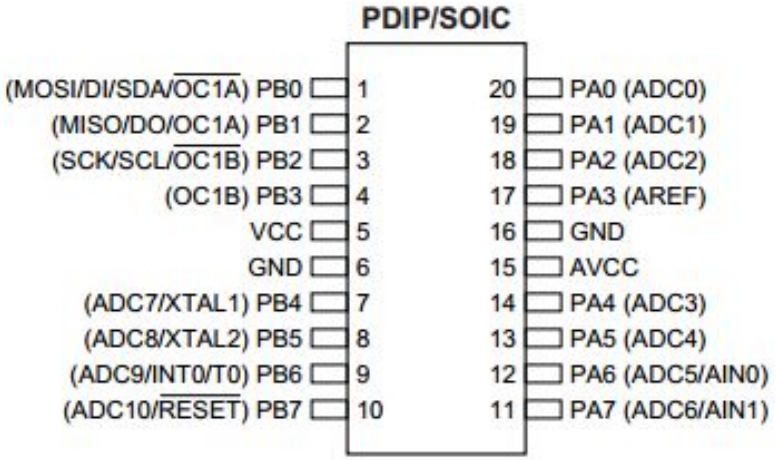
Bit	7	6	5	4	3	2	1	0	
$2E ($4E)	MSB							LSB	TCNT1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

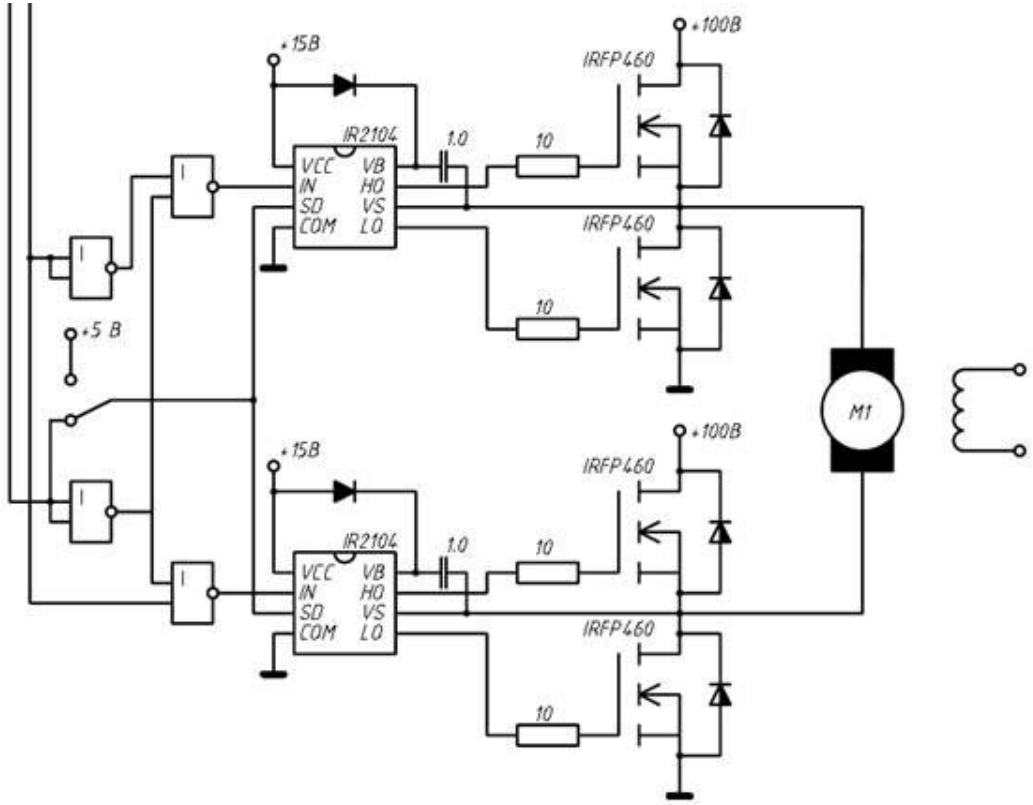
Микроконтроллер ATtiny26 (ШИМ)

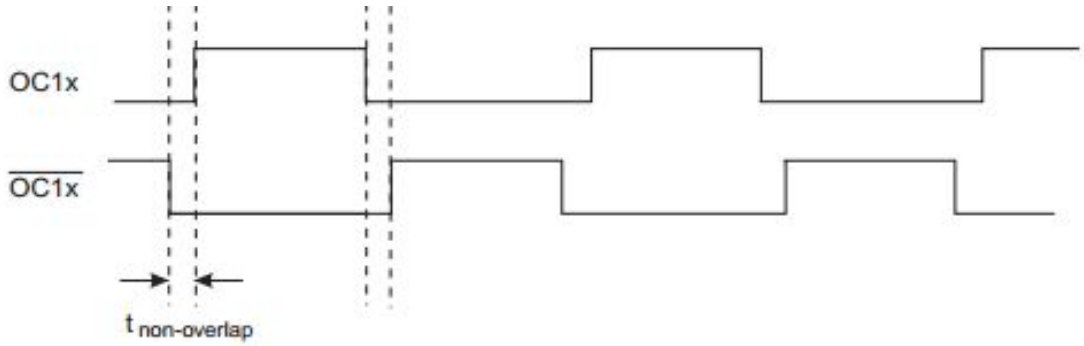




Микроконтроллер ATtiny26 (ШИМ)







AVR директивы и функции ассемблера

Входная строка может иметь одну из четырёх форм:

```
[метка:] директива [операнды] [Комментарий]
[метка:] инструкция [операнды] [Комментарий]
Комментарий
Пустая строка
```

Комментарий имеет следующую форму:

```
; [Текст]
```

Позиции в квадратных скобках необязательны. Текст после точки с запятой (;) и до конца строки игнорируется компилятором. Метки, инструкции и директивы более детально описываются ниже.

```
;*-------------------------------------------------------------------*
; Начальные установки
.org    $013
AAA:   reti                             ;

MAIN:
        cli                             ; запрет прерываний

        ldi     temp1,high(RAMEND)      ; Установили стек
        out     SPH,temp1               ;
        ldi     temp1,low(RAMEND)       ; Установили стек
        out     SPL,temp1               ;

        ldi     temp1,0                 ; Выдали 0 во все биты на выходе
        out     portb,temp1             ;
        out     portd,temp1             ;
        out     portc,temp1             ;
```

AVR директивы ассемблера

Директива	Описание
BYTE	Зарезервировать байты в ОЗУ
CSEG	Программный сегмент
DB	Определить байты во флэш или EEPROM
DEF	Назначить регистру символическое имя
DEVICE	Определить устройство для которого компилируется программа
DSEG	Сегмент данных
DW	Определить слова во флэш или EEPROM
ENDM, ENDMACRO	Конец макроса
EQU	Установить постоянное выражение
ESEG	Сегмент EEPROM
EXIT	Выйти из файла
INCLUDE	Вложить другой файл
LIST	Включить генерацию листинга
LISTMAC	Включить разворачивание макросов в листинге
MACRO	Начало макроса
NOLIST	Выключить генерацию листинга
ORG	Установить положение в сегменте
SET	Установить переменный символический эквивалент выражения

Все директивы предваряются точкой.

AVR директивы ассемблера (пример)

```
;*-------------------------------------------------------*
; Константы
;*-------------------------------------------------------*
.equ    PPP2 = 208      ; UBRR для скорости 416-2400 208-4800 103-9600
.equ    PPP3 = 50       ; 50мс - синхропауза
.equ    PPP4 = 2        ; 2мс - время удержания стоп бита в конце посылки

.equ    dreb_4=3        ; постоянные по подсчету дребезга для импульсных входов
.equ    dreb_5=6        ;
.equ    dreb_6=2        ; гистерезис

;*-------------------------------------------------------*
; Переменные используемые в программе
;*-------------------------------------------------------*
.dseg                   ;
timer:   .byte   5      ; счетчики для таймеров0
                        ; Всегда первый в определении к ним привязка ОЗУ

clok:     .byte  25     ; Разные часы
                        ; +0 - (1мс)
                        ; +1 - (1мс)
```

```
        lds     temp1,(clok+9)          ; счетчик синхропаузы
        cpi     temp1,PPP3              ;
        brsh    PTXD_OUT5               ;
        rjmp    PTXD_OUTW               ;
PTXD_OUT5:
        ldi     temp1,3                 ; изменили режим запроса
        sts     Zapros_TXD,temp1        ;
        clr     temp1                   ;
        sts     (Data_TXD+20),temp1     ; сброс счетчика передаваемых байт
```

AVR операторы ассемблера

Приоритет	Символ	Описание
14	!	Логическое отрицание
14	~	Побитное отрицание
14	-	Минус
13	*	Умножение
13	/	Деление
12	+	Суммирование
12	-	Вычитание
11	<<	Сдвиг влево
11	>>	Сдвиг вправо
10	<	Меньше чем
10	<=	Меньше или равно
10	>	Больше чем
10	>=	Больше или равно
9	==	Равно
9	!=	Не равно
8	&	Побитное И
7	^	Побитное исключающее ИЛИ
6	|	Побитное ИЛИ
5	&&	Логическое И
4	||	Логическое ИЛИ

AVR функции ассемблера

LOW(выражение) возвращает младший байт выражения
HIGH(выражение) возвращает второй байт выражения
BYTE2(выражение) то же что и функция
HIGHBYTE3(выражение) возвращает третий байт выражения
BYTE4(выражение) возвращает четвёртый байт выражения
LWRD(выражение) возвращает биты 0-15 выражения
HWRD(выражение) возвращает биты 16-31 выражения
PAGE(выражение) возвращает биты 16-21 выражения
EXP2(выражение) возвращает 2 в степени (выражение)
LOG2(выражение) возвращает целую часть log2(выражение)

```
;*-------------------------------------------------------------------*
; Начальные установки
.org    $013
AAA:   reti                             ;

MAIN:
        cli                             ; запрет прерываний

        ldi     temp1,high(RAMEND)      ; Установили стек
        out     SPH,temp1               ;
        ldi     temp1,low(RAMEND)       ; Установили стек
        out     SPL,temp1               ;

        ldi     temp1,0                 ; Выдали 0 во все биты на выходе
        out     portb,temp1             ;
        out     portd,temp1             ;
        out     portc,temp1             ;
```