



Тема:

**Основные команды и директивы ATmega16
(продолжение).**

**к.т.н., доцент каф.501
Мазуренко А.В.**

EQU – Присвоить символическому имени постоянное значение

Синтаксис: ***.EQU*** <символическое_имя> = <значение или выражение>

SET – Присвоить символическому имени значение (переменное)

Синтаксис: ***.SET*** <символическое_имя> = <значение или выражение>

MACRO – Определить начало макроса

Синтаксис: ***.MACRO*** <имя макроса>

ENDMACRO - Определить конец макроса

Синтаксис: ***.ENDMACRO***

INCLUDE - Вложить другой файл

Синтаксис: ***.INCLUDE*** "имя_файла"

EXIT - Выйти из файла (закончить компиляцию файла)

Синтаксис: ***.EXIT***

“m16def.inc” – Файл описания символических имен ATmega16:

```
-----  
;***** Specify Device  
.device ATmega16  
  
;***** I/O Register Definitions  
.equ  SREG  = $3f  
.equ  SPH   = $3e  
.equ  SPL   = $3d  
  
.....  
.equ  RAMEND = $45F  
  
.....  
.equ  SPMRaddr = $028 ;Store Program Memory Ready Interrupt Vector Address  
-----
```

Файлы описания символических имен AVR-МК размещаются в папке установки AVR Studio:

... \Atmel\AVR Tools\AvrAssembler\Appnotes

Основные функции компилятора AVR Studio:

- **LOW**(*<значение или выражение>*) – возвращает младший байт *<значения или выражения>*.
- **HIGH**(*<значение или выражение>*) – возвращает второй байт *<значения или выражения>*.
- **BYTE2**(*<значение или выражение>*) – возвращает то же значение что и функция HIGH.
- **BYTE3**(*<значение или выражение>*) – возвращает третий байт *<значения или выражения>*.
- **BYTE4**(*<значение или выражение>*) – возвращает четвёртый байт *<значения или выражения>*.
- **EXP2**(*<значение или выражение>*) – возвращает 2 в степени *<значения или выражения>*.

Основные операторы компилятора AVR Studio:

Обозначение	Описание
!	Логическое НЕ
~	Побитное НЕ
-	Унарный МИНУС
*	Умножение
/	Деление
+	Суммирование
-	Вычитание
<<	Сдвиг влево
>>	Сдвиг вправо
&	Побитное И
^	Побитное ИСКЛЮЧАЮЩЕЕ ИЛИ
	Побитное ИЛИ
&&	Логическое И
	Логическое ИЛИ

MOV - Copy Register – Копировать регистр

Операция: $Rd \leftarrow Rr$

Синтаксис:	Операнды:	Счетчик команд:
mov Rd,Rr	$0 \leq d,r \leq 31$	$PC \leftarrow PC+1$

Флаги на которые воздействует команда: не воздействует
Количество тактов выполнения операции: 1.

MOVW - Copy Register Word – Копировать пару регистров

Операция: $Rd+1:Rd \leftarrow Rr+1:Rr$

Синтаксис:	Операнды:	Счетчик команд:
movw Rd+1:Rd,Rr+1:Rr	$d,r \in [0,2,\dots,30]$	$PC \leftarrow PC+1$

Флаги на которые воздействует команда: не воздействует
Количество тактов выполнения операции: 1.

OUT - Store Register to I/O Location – Загрузить данные из регистра общего назначения в регистр ввода\вывода

Операция: $I/O(AdrIO) \leftarrow Rr$

Синтаксис:	Операнды:	Счетчик команд:
out AdrIO,Rr	$0 \leq r \leq 31, 0 \leq AdrIO \leq 63$	$PC \leftarrow PC+1$

Флаги на которые воздействует команда: не воздействует
Количество тактов выполнения операции: 1.

COM – One’s Complement – Дополнить до единицы (получить обратный код)

Операция: $Rd \leftarrow \$FF - Rd$

Синтаксис:	Операнды:	Счетчик команд:
com Rd	$0 \leq d \leq 31$	$PC \leftarrow PC+1$

Флаги на которые воздействует команда: S, V←0, N, Z, C←1

Количество тактов выполнения операции: 1.

NEG – Two’s Complement – Дополнить до двух (получить дополнительный код)

Операция: $Rd \leftarrow \$00 - Rd$

Синтаксис:	Операнды:	Счетчик команд:
neg Rd	$0 \leq d \leq 31$	$PC \leftarrow PC+1$

Флаги на которые воздействует команда: H, S, V, N, Z, C

Количество тактов выполнения операции: 1.

INC - Increment – Увеличить на 1 (инкрементировать)

Операция: $Rd \leftarrow Rd + 1$

Синтаксис:	Операнды:	Счетчик команд:
inc Rd	$0 \leq d \leq 31$	$PC \leftarrow PC+1$

Флаги на которые воздействует команда: H, S, V, N, Z

Количество тактов выполнения операции: 1.

DEC - Decrement – Уменьшить на 1 (декрементировать)

Операция: $Rd \leftarrow Rd - 1$

Синтаксис:	Операнды:	Счетчик команд:
dec Rd	$0 \leq d \leq 31$	$PC \leftarrow PC + 1$

Флаги на которые воздействует команда: H, S, V, N, Z
Количество тактов выполнения операции: 1.

CP - Compare – Сравнить

Операция: $Rr1 - Rr2$

Синтаксис:	Операнды:	Счетчик команд:
cp Rr1,Rr2	$0 \leq r1, r2 \leq 31$	$PC \leftarrow PC + 1$

Флаги на которые воздействует команда: H, S, V, N, Z, C
Количество тактов выполнения операции: 1.

CPC - Compare with Carry – Сравнить с учетом переноса

Операция: $Rr1 - Rr2 - C$

Синтаксис:	Операнды:	Счетчик команд:
cpc Rr1,Rr2	$0 \leq r1, r2 \leq 31$	$PC \leftarrow PC + 1$

Флаги на которые воздействует команда: H, S, V, N, Z, C
Количество тактов выполнения операции: 1.

CPI - Compare with Immediate – Сравнить с константой

Операция: $Rr - K8$

Синтаксис:	Операнды:	Счетчик команд:
<code>cpi Rr,K8</code>	$0 \leq r \leq 16$	$PC \leftarrow PC+1$

Флаги на которые воздействует команда: H, S, V, N, Z, C

Количество тактов выполнения операции: 1.

TST – Test for Zero or Minus – Проверить на нулевое или отрицательное значение

Операция: $Rr \wedge Rr$

Синтаксис:	Операнды:	Счетчик команд:
<code>tst Rr</code>	$0 \leq r \leq 31$	$PC \leftarrow PC+1$

Флаги на которые воздействует команда: S, $V \leftarrow 0$, N, Z

Количество тактов выполнения операции: 1.

SUBI – Subtract Immediate – Вычесть непосредственное значение

Операция: $Rd \leftarrow Rd - K8$

Синтаксис:	Операнды:	Счетчик команд:
<code>subi Rd,K8</code>	$16 \leq d \leq 31$	$PC \leftarrow PC+1$

Флаги на которые воздействует команда: H, S, V, N, Z, C

Количество тактов выполнения операции: 1.

SBCI – Subtract Immediate with Carry – Вычесть непосредственное значение с учетом переноса
 Операция: $Rd \leftarrow Rd - K8 - C$

Синтаксис:	Операнды:	Счетчик команд:
sbcI Rd,K8	$16 \leq d \leq 31$	$PC \leftarrow PC+1$

Флаги на которые воздействует команда: H, S, V, N, Z, C
Количество тактов выполнения операции: 1.

RCALL – Relative Call to Subroutine – Относительный вызов подпрограммы

Операция: $PC \leftarrow PC+k+1$

Синтаксис:	Операнды:	Счетчик команд:	Стек:	Указатель стека:
rcall k	$-2K \leq k \leq +2K$	$PC \leftarrow PC+k+1$	$STACK \leftarrow PC+1$	$SP \leftarrow SP-2$

Флаги на которые воздействует команда: не воздействует

Количество тактов выполнения операции: 3.

CALL – Long Call to Subroutine – Длинный относительный вызов подпрограммы

Операция: $PC \leftarrow PC+k+1$

Синтаксис:	Операнды:	Счетчик команд:	Стек:	Указатель стека:
call k	$0 \leq k \leq 64K$	$PC \leftarrow PC+k+1$	$STACK \leftarrow PC+2$	$SP \leftarrow SP-2$

Флаги на которые воздействует команда: не воздействует

Количество тактов выполнения операции: 4.

RET – Return from Subroutine – Возврат из подпрограммы

Операция: $PC \leftarrow STACK(SP)$

Синтаксис:	Операнды:	Счетчик команд:	Указатель стека:
ret	-	$PC \leftarrow STACK(SP)$	$SP \leftarrow SP+2$

Флаги на которые воздействует команда: не воздействует

Количество тактов выполнения операции: 4.

RETI – Return from Interupt – Возврат из подпрограммы-обработчика прерывания

Операция: $PC \leftarrow STACK(SP)$

Синтаксис:	Операнды:	Счетчик команд:	Указатель стека:
reti	-	$PC \leftarrow STACK(SP)$	$SP \leftarrow SP+2$

Флаги на которые воздействует команда: $I \leftarrow 1$

Количество тактов выполнения операции: 4.

BRPL – Branch if Plus – Перейти если положительное

Операция: If $Rr > 0$ ($N = 0$) then $PC \leftarrow PC+k+1$ else $PC \leftarrow PC+1$

Синтаксис:	Операнды:	Счетчик команд:
brpl k	$-64 \leq k \leq +63$	$PC \leftarrow PC+k+1$, если условие выполняется $PC \leftarrow PC+1$, если условие не выполняется

Флаги на которые воздействует команда: не воздействует

Количество тактов выполнения операции: 2 (если условие выполняется);
1 (если условие не выполняется).

BRMI –Branch if Minus – Перейти если отрицательное

Операция: If $Rr < 0$ ($N = 1$) then $PC \leftarrow PC+k+1$ else $PC \leftarrow PC+1$

Синтаксис:	Операнды:	Счетчик команд:
brmi k	$-64 \leq k \leq +63$	$PC \leftarrow PC+k+1$, если условие выполняется $PC \leftarrow PC+1$, если условие не выполняется

Флаги на которые воздействует команда: не воздействует

Количество тактов выполнения операции: 2 (если условие выполняется);
1 (если условие не выполняется).

Условия задачи 1:

В ячейках ОЗУ начиная с адреса **AdrSi** находится знаковое двухбайтное число **Si** ($|Si| < 1000_{(10)}$) записанное в прямом коде.

Задание:

Написать ПО МК ATmega16, которое бы выполняло:

- 1) сложение числа **Si** и двухбайтной знаковой константы **Ki** ($|Ki| < 1000_{(10)}$);
- 2) запись всех байтов результата в прямом коде в ячейки ОЗУ с адресами, начиная с адреса **AdrResi**.

В программе считывание и запись чисел из\в ОЗУ оформить в виде макроса, а преобразование чисел оформить в виде подпрограммы.

В программе пункты задания 1 и 2 выполнить два раза:

- 1) для значения констант **K1** = + 100 и адресов: **AdrS1**=0x60, **AdrRes1** = 0x6A;
- 2) для значения констант **K2** = -10 и адресов: **AdrS2**=0x70, **AdrRes2** = 0x7A;