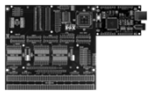


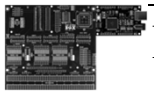
# EEL 3744

## Menu

- Wonders of the include file
- Bit Masks
- Bit Positions
- Group Configurations


See examples on web-site:
`ATxmega128a1udef.inc`

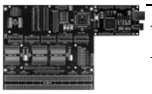




# EEL 3744 Wonders of the Include File

- XL, XH, YL, YH, ZL, ZH defined as r26-r31, near end of include file
- See SRAM_END, RAMEND, SRAM_SIZE, etc.
- CPU registers, e.g., CPU_SREG
- PORTs, e.g., PORTF_DIRSET and PORTF_OUTSET
- Bitmasks (_bm) allow a user to change only specific bits without knowing the bit positions
- Bit positions (_bp) allow the programmer to change specific bits
- Group Configurations (_gc) allow the programmer to change how a particular system works without remember bit positions

`ATxmega128a1udef.inc`



## EEL 3744 Bit Mask & Bit Position

- Bitmasks (_bm) allow a user to change only specific bits without knowing the bit positions
- Ex:

ATxmega128a1udef.inc

```
.equ CPU_Z_bm = 0x02
.equ CPU_C_bm = 0x01
```

- Bit positions (_bp) allow the programmer to change specific bits
- Ex:

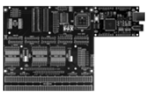
```
.equ CPU_Z_bp = 1
.equ CPU_C_bp = 0
```

Status Register (SREG)

See doc8331, section 3.14.9

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |





## EEL 3744 Example Using BM

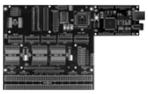
ATxmega128a1udef.inc

- The following two lines are in the include file

```
.equ OSC_RC32MEN_bm = 0x02
   ; Internal 32 MHz RC Osc Enable bit mask
```

- So if you want to make the oscillator at 32 MHz you could use below without memorizing bit positions

```
ldi r16, OSC_RC32MEN_bm
sts OSC_CTRL, r16
```



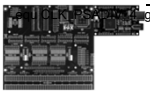
EEL 3744

# Example Using BP

- Since the RED_LED is on PortD bit 4 and the GREEN_LED on PortD5, could use the following to turn on the Red and clear the Green (active-low) LEDs

```
.equ RED_LED_BP   = 4
.equ GREEN_LED_BP = 5
...
ldi    r16,(1 << RED_LED_BP)
sts PORTD_OUTCLR, r16
ldi    r16,(1 << GREEN_LED_BP)
sts PORTD_OUTSET, r16
```





gc = (0x03<<2) ; Divide by 4

EEL 3744

# Group Configuration

ATxmega128a1udef.inc

- Group Configurations (_gc) allow the programmer to change how a particular system works without remember bit positions
- Ex: Prescaler A Division Factor (for System Clock)

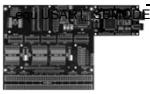
```
.equ CLK_PSADIV_2_gc = (0x01<<2) ; Divide by 2
.equ CLK_PSADIV_4_gc = (0x03<<2) ; Divide by 4
.equ CLK_PSBCDIV_1_1_gc = (0x00<<0) ;Divide B by 1 and C by 1
.equ CLK_PSBCDIV_1_2_gc = (0x01<<0) ;Divide B by 1 and C by 2
```

> Configure for pre-scaler A divide by 2 and pre-scalers B & C divide by 1:

```
ldi r16,(CLK_PSADIV_2_gc | CLK_PSBCDIV_1_1_gc)
sts CLK_PSCTRL, r16
```

CLK_PSCTRL

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | PSADIV[4:0] | | | | | PSBCDIV | |




EEL 3744

# Group Configuration

- Ex: For asynchronous UART on PC0, 6-bits, odd parity, 2 stop bits
  > The following is in the include file

```
.equ USART_CMODE_ASYNCHRONOUS_gc = (0x00<<6)
     ; Asynchronous Mode
.equ USART_CHSIZE_6BIT_gc = (0x01<<0)
     ; Character size: 6 bit
.equ USART_PMODE_ODD_gc = (0x03<<4)
     ; Odd Parity
```

  > Add the following for the stop bits (for some reason, **NOT** in the include file)

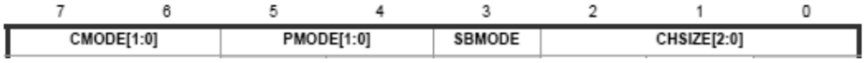
```
.equ USART_SBMODE_2BIT_gc = (0x01<<3)
.equ USART_SBMODE_1BIT_gc  = (0x00<<3)
```

  > Then use below to finish the initialization

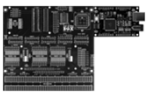
```
ldi r16, USART_CMODE_ASYNCHRONOUS | USART_CHSIZE_6BIT_gc |
        USART_PMODE_ODD_gc | USART_SBMODE_2BIT_gc
sts USARTC0_CTRLC, r16
```

ATxmega128a1udef.inc

USARTC0_CTRLC





EEL 3744

# *The End!*