XMEGA

Warsztaty CHIP

Rok akademicki 2014/2015

Plan warsztatów:

- Wprowadzenie do Atmel Studio (20/11/2014)
- Porty I/O (20/11/2014)
- Przerwania (27/11/2014)

Wykorzystana literatura:

- [1] Dokumentacja ATMEL(www.atmel.com):
  http://www.atmel.com/images/doc8050.pdf
  http://www.atmel.com/Images/doc8075.pdf
  http://www.atmel.com/Images/doc8169.pdf i inne
- [2] Tomasz Francuz, „AVR praktyczne projekty”, Helion 2013.
- [3] Kurs internetowy Leon Instruments:
  http://www.leon-instruments.pl/2013/04/poznaj-atxmega.html

Porty I/O podstawowe informacje:

- Standard napięciowy 3.3V (lub niższy przy niższym zasilaniu uC).
- Max. prąd 25mA/wyprowadzenie, suma wszystkich prądów <200mA.
- Po resecie procesora wszystkie wyprowadzenia I/O są ustawione w tryb wejściowy bez podciągania.
- Wyprowadzenia I/O można obsługiwać identycznie jak w Atmega jednak XMEGA daje więcej możliwości.

Porty I/O c.d.

- Wyprowadzenia I/O są pogrupowane w porty, zazwyczaj 8 bitowe, czasami krótsze.
- Każdy port procesora ma 22 skojarzone z nim rejestry.
- Odpowiedniki tych rejestrów pomiędzy poprzednimi rodzinami AVR a polami struktur w rodzinie XMEGA:

  PORT -> OUT, PIN -> IN, DDR ->DIR

- Np.:

  PORTA ⇔ PORTA.OUT

  PINA ⇔ PORTA.IN

  DDRA ⇔ PORTA.DIR

- Pozostałe rejestry XMEGA nie mają odpowiedników w starszych rodzinach.

Porty I/O c.d.

- Pole DIR – ustawienie kierunku, 0=IN, 1=OUT,

```
PORTA.DIR = 0xFF; // ustawienie 8 x out
PORTA.OUT = 0b11001101;
```

- Pole OUT – wyjście sygnału,

```
PORTA.DIR = 0xFF; // ustawienie 8 x out
PORTA.OUT = 0b11001101; //zapis do portu
```

- Pole IN – odczyt stanu,

```
uint8_t data;
PORTA.DIR = 0x00; // ustawienie 8 x out
data = PORTA.IN; // zapis wejścia do zmiennej data
```

Porty I/O c.d.

- Pola DIRSET, OUTSET – ustawienie 1 na danym bicie,

```
PORTA.DIRSET = PIN1_bm; // ustawienie PIN1 jako wyjście
PORTA.OUTSET = 0b11001101; //ustawienie wskazanych bitów
```

- Pola DIRCLR, OUTCLR – ustawienie 0 na danym bicie,

```
PORTA.DIRCLR = PIN7_bm; // ustawienie PIN7 jako wejście
PORTA.OUTCLR = 0b11001101; //wyzerowanie wskazanych bitów
```

- Pola DIRTGL, OUTTGL – zmiana stanu danego bitu,

```
PORTA.DIRTGL = PIN7_bm; // zmiana trybu pracy PIN7
PORTA.OUTTGL = 0b11001101; //zmiana stanu bitów wskazanych
jako „jedynka”
```

Porty I/O c.d.

- Pola PIN0CTRL – PIN7CTRL,
- Sterowanie inwersja wejsicia i wyjscia

```
PORTA.PIN0CTRL = PORT_INVEN_bm; //PIN0 zostal zanegowany
```

- Sterowanie SLEW RATE,

```
PORTA.PIN7CTRL = PORT_SRLEN_bm; //PIN7 ma zmniejszony SR (50-150%)
```

- Sterowanie „utrzymaniem bitu”,

```
PORTA.PIN0CTRL = PORT_OPC_BUSKEEPER_gc;
PORTA.PIN0CTRL = PORT_OPC_PULLDOWN_gc;
```

Table 2-1. Output/pull configurations

Symbol	Output Configuration	Pull Configuration
PORT_OPC_TOTEM_gc	Totempole	(N/A)
PORT_OPC_BUSKEEPER_gc	Totempole	Bus keeper on input and output
PORT_OPC_PULLDOWN_gc	Totempole	Pull-down on input
PORT_OPC_PULLUP_gc	Totempole	Pull-up on input
PORT_OPC_WIREDOR_gc	Wired-OR	(N/A)
PORT_OPC_WIREDAND_gc	Wired-AND	(N/A)
PORT_OPC_WIREDORPULL_gc	Wired-OR	Pull-down
PORT_OPC_WIREDANDPULL_gc	Wired-AND	Pull-up

Porty I/O c.d.

- Konfigurowanie wielu PINów jednocześnie,

```
PORTCFG.MPCMASK = 0x0F;    // ustawienie maski których pinów ma
                           // dotyczyć konfiguracja (tu: 3-0)
PORTC.PIN0CTRL = PORT_OPC_PULLUP_gc; // wpis do jednego z pinów
                                     // automatycznie przechodzi
                                     // do pozostałych
```

- Konfigurowanie przerwań na portach

```
PORTC.PIN0CTRL = PORT_ISC_BOTHEDGES_gc;  // IRQ wyzwalane na obu
                                         // zboczach PIN0
PORTC.INT0MASK  = PIN0_bm | PIN1_bm;    // piny [0-1] to INT0
```

Symbol	Configuration
PORT_ISC_BOTHEDGES_gc	Sense both edges
PORT_ISC_RISING_gc	Sense rising edge
PORT_ISC_FALLING_gc	Sense falling edge
PORT_ISC_LEVEL_gc	Sense low level (transparent for events)
PORT_ISC_INPUT_DISABLE_gc	Digital input buffer disabled

Porty I/O c.d.

- Zdarzenia na portach można przyporządkować do przerwania o nazwie INT0 i INT1.
- Przerwaniom INT0 oraz INT1 przypisuje się jeden z 3 możliwych poziomów (HI, MED, LO), np.:

```
PORTC.INTCTRL = PORT_INT0LVL_MED_gc | // Medium for INT0
                PORT_INT1LVL_LO_gc;   // Low for INT1
```

- Można także dokonać remapowania PINów – nie jest to omówione w tych materiałach.

Przerwania

- Przerwanie jest jednym z ważniejszych elementów występującym w mikrokontrolerach i systemach komputerowych.
- Przerwanie polega na zaprzestaniu wykonywania głównego programu i wejściu do specjalnej funkcji obsługi danego przerwania. Po zakończeniu przerwania następuje wznowienie wykonywania programu głównego.
- Zgłoszenie przerwania powoduje zapamiętanie stanu procesora na stosie a następnie wykonanie funkcji obsługi przerwania i na koniec odtworzeniu stanu procesora ze stosu.
- Zapamiętaniu na stosie podlegają następujące elementy: rejestry procesora, flagi, stan licznika rozkazów.

Przerwania c.d.

- Elementy nie podlegające zapamiętaniu: wartości zmiennych, wartości portów I/O i rejestrów konfiguracyjnych bloków sprzętowych.
- Co może wywołać przerwanie? Są to: bloki sprzętowe mikorontrolera (np.. ADC, DMA, USART, SPI, I2C), timery, wyprowadzenia I/O, zablokowanie zegara uC.
- Aby użyć przerwania skojarzonego z danym wektorem przerwania należy wykonać następujące czynności:

  - zdefiniować funkcję obsługi przerwania,

  - przypisać poziom obsługi danego przerwania,

  - odblokować dane przerwanie,

  - odblokować ustalony wcześniej poziom obsługi w kontrolerze przerwań PMIC,

  - odblokować globalną maskę przerwań.

Definicja funkcji obsługi przerwania na przykładzie przerwania od I/O

```
ISR(PORTC_INT0_vect)  //obsługa przerwania INT0 występującego
{                     //na PORTC
        if (pb0_pressed())
                led0(1);
        else
                led0(0);
        if (pb1_pressed())
                led1(1);
        else
                led1(0);
}
```

Skonfigurowanie I/O jako IRQ oraz przypisanie poziomu obsługi

```
void pb_irq_init()
{
PORTC.DIRCLR     = PIN0_bm | PIN1_bm | PIN2_bm | PIN3_bm;
PORTCFG.MPCMASK = PIN0_bm | PIN1_bm | PIN2_bm | PIN3_bm;
PORTC.PIN0CTRL   = PORT_OPC_PULLUP_gc | PORT_ISC_BOTHEDGES_gc;
   // pins [0-3]  pullup and both edge irq request

PORTC.INT0MASK   = PIN0_bm | PIN1_bm;     // pins [0-1] call INT0
PORTC.INT1MASK   = PIN2_bm | PIN3_bm;     // pins [2-3] call INT1
PORTC.INTCTRL    = PORT_INT0LVL_MED_gc | // Medium level of INT0
                   PORT_INT1LVL_LO_gc;    // and low level for INT1
}
```

Odblokowanie poziomu przerwania i globalne zezwolenie

```
// turning ON IRQs of MED and LO level
PMIC.CTRL = PMIC_MEDLVLEN_bm | PMIC_LOLVLEN_bm;

// global IRQ ON
sei();

//global OFF - cli();
cli();
```

Przerwania c.d.

- Wszystkie przerwania oprócz przerwania uszkodzenia kwarcu generatora są maskowalne (można je wyłączyć).
- Przerwanie o poziomie wyższym może uruchomić się w czasie obsługi przerwania o poziomie niższym.
- Można również ustalić dynamiczny priorytet, wówczas ostatnio obsłużone przerwanie uzyskuje najniższy priorytet a przerwania o wyższym adresie obsługi mają wyższy priorytet (Round Robin)

```
PMIC.CTRL |= PMIC_RREN_bm;
```

- Włączenie / wyłączenie globalnego bitu przerwań instrukcjami:

```
// global IRQ ON
sei();
```

```
//global OFF - cli();
cli();
```

Przerwania c.d.

- Do wymiany danych pomiędzy funkcjami obsługi przerwań a programem głównym często używa się zmiennych globalnych – trzeba im nadać atrybut volatile inaczej mogą zostać zoptymalizowane i zapis danych do nich z funkcji obsługi przerwania się nie odbędzie !!!

```
volatile uint8_t a;
```

- Należy zapewnić atomowy dostęp/zapis do danych więcej niż 1 – bajtowych do wrażliwych danych, których zapis przerwany przerwaniem mógłby spowodować nieprawidłową pracę programu.