Instrukcja do laboratorium Akademii ETI 26.03.2014^{*}

I. Logowanie do systemu

Aby zalogować się do komputera należy podać następującego użytkownika i hasło:

- w sali 308: **1ab1/1ab1**
- w sali 325: *student1/student1*

Po zalogowaniu użytkownicy mają dostęp do następujących zasobów:

- w sali 308: C: \Designs
- w sali 325: D: \Designs

II. Uruchomienie środowiska programistycznego, utworzenie przykładowego projektu i wykonanie kompilacji

Jako środowisko programistyczne wykorzystane jest **Atmel Studio 6.2**. W celu jego uruchomienia należy wybrać menu::

- w sali 308: START/Programy/Atmel/Atmel Studio 6.2

w sali 325: /Wszystkie Programy/Atmel/Atmel Studio 6.2

Po kliknięciu w **Atmel Studio 6.2** uruchomi się zintegrowane środowisko programistyczne. W przypadku gdyby pojawiło się pytanie o aktualizację należy tą aktualizację anulować. Okno środowiska powinno wyglądać jak na poniższym rysunku:

🖗 Start Page - AtmelStudio (Administrator)	the second s		- 0 - X
File Edit View VAssistX ASF Project Debug Too	ls Window Help		
1 - 4	2 · L, D, L, D M - D	國해요~ 말 送來! 고문! 그 한 것 한 한 것 한 것 한	
1 2 2 3 4 5 8 6 4 4 4 4 1 1 1 1 1 1 1 1	· · · · · · · · · · · · · · · · · · ·	Device T No Tool -	
Start Page ×		Solution Explorer	- i ×
		atmete 6	
New Project	Get Started Tools Help Latest News		
New Example Project	welcome Links and Resources		
G Open Project	Welcome to Atmel Studio Get to know Atmel Studio.		
Recent Projects	User Guide Getting Started	1 ASS Sentence - WA Monte - All Advices -	
statmega32	Programming Dialog		
schip_mini_xmege_inq	The second secon	ropenes	***
le dwarmowska		Ap. A. 1995	
🏚 atimegalle5	Atmel Software Framework	(at) (2+) (m)	
a chip_mini_xmega			
Simega_demo			
test_atmega128	Virian Tutorials		
smegaal.zoasu_example			
Close page after project load			
Show page on startup			
Oxford			*1*
Show output from:			
	1916-914100		
Server List Contend			
and the second s			
nasay			

W celu utworzenia nowego projektu klikamy na menu nemu in New Project... . Pojawi się nowe okno, które wypełnimy następująco:



- w części **Installed Templates** upewniamy się, że wybrane jest pole *C/C++*,
- zaznaczamy projekt typu GCC C Executable Project,
- w polu *Name* wpisujemy nazwę naszego projektu np.: *cw_1*
- pole *Location* należy wybrać następująco:
 - dla sali 308 w katalogu: C: \Designs\"katalog projektu"
 dla sali 325 w katalogu: D: \Designs\"katalog projektu"

Przykład prawidłowego wypełnienia okna przedstawiony jest na rysunku poniżej.



Po kliknięciu *OK*, w kolejnym oknie należy wybrać układ mikrokontrolera na który tworzony będzie projekt. Na płytkach prototypowych dostępnych w czasie zajęć umieszczony jest układ z rodziny *AVR ATXMEGA* typu *ATXMEGA*128A4U i taki należy wybrać.

Device Selection							
Device Family:	AVR XMEGA, 8-bit 🔹					Search for device	۶
Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (byte	ŧ	Device Info:		
ATxmega128A1	136	8192	2048	*	Device Name:	ATxmega128A4U	
ATxmega128A1U	136	8192	2048		Speed	32	
ATxmega128A3	136	8192	2048		Ver	16/26	
ATxmega128A3U	136	8192	2048		VCC:	1,0/5,0	
ATxmega128A4U	136	8192	2048	Ξ	Family:	AVR XMEGA	
ATxmega128B1	136	8192	2048		Datashee	ts	
ATxmega128B3	136	8192	2048				
ATxmega128C3	136	8192	2048		Supported To	ols	
ATxmega128D3	136	8192	2048		Atmel-ICE		
ATxmega128D4	136	8192	2048		AVR Drag	on	
ATxmega16A4	20	2048	1024		• <u>Avit Diag</u>		
ATxmega16A4U	20	2048	1024		AVRISP m	<u>KII</u>	
ATxmega16C4	20	2048	1024		AVR ONE		
ATxmega16D4	20	2048	1024				
ATxmega16E5	20	2048	512			2	
ATxmega192A3	200	16384	2048		JTAGICE	mkll	
ATxmega192A3U	200	16384	2048		Simulator		
ATxmega192C3	200	16384	2048		STK600		
AT 10000	200	10004	20.40				
						OK	Cancel

Po kliknięciu **OK** środowisko **Atmel Studio 6.2** będzie w postaci przedstawionej na poniższym rysunku.

🗠 cw 1 - AtmelStudio (Administrator)			
File Edit View VAssistX ASF Project Build Debug Tools Window Help			
1 + 色 - G J J J J A L O - C - J - L 図 Q I A M Debug ・ 図 ・ 図 ・ 図 ・ 2 目 3 目 8 4 1 - 1 目 法 目 1 日 2 日	두 목 두 수 및 및 및 _		
No State (Sta			
	- Solution Emlorer		* 1 ×
or contraction of the contractio			
*	Solution 'cw_1' (1 project)		
* Created: 2015-03-17 11:40:39	a cw_1		
Author: bpa	Dependencies		
	 Bit Libraries 		
	C cw1.c		
#include <avr io.h=""></avr>			
e art main(void)			
while(1)	E		
(
//TODO:: Please write your application code			
	🔍 ASF Explorer 👄 VA View 👘 VA Outline 🔫 Solution	Explorer	
	Properties		
	cw_1.c File Properties		
	20 21 III		
	Advanced		A
	Build Action	Compile	
	Custom Compilation Setting		E
	Misc		
		False	
	Ella Mana	and a	
100 % - 4	*		
Const			* 1 Y
show output norms into a recogning			
			*
Constitute = Constantiate = D Output			÷
Section ESC - Sectionity - Option			
Ready			

Kolejnym krokiem jest napisanie programu obsługi mikrokontrolera (w lewym górnym oknie IDE) i jego kompilacja (klawisz **F7**). Po kompilacji kod programu można zaprogramować w mikrokontrolerze. Przykłady programów podane są w części końcowej niniejszych materiałów.

III. Zaprogramowanie mikrokontrolera na płytce uruchomieniowej

Płytki wykorzystywane w czasie zajęć laboratoryjnych Akademii ETI można programować na 2 sposoby:

- poprzez wykorzystanie programatora podłączonego do złącza PDI (6 pinów przy przycisku *Reset*),
- poprzez złącze USB i wykorzystanie *Bootloadera*.

W czasie zajęć wykorzystana zostanie druga metoda. Kolejność postępowania jest następująca:

- podłączamy płytkę kablem USB do komputera,
- wprowadzamy mikrokontroler z tryb Bootloaredra poprzez równoczesne naciśnięcie przycisków Reser i Flip a następnie puszczenie przycisku Reset przed zwolnieniem przycisku Flip,



uruchamiamy oprogramowanie FLIP (dostępne na pulpicie komputerów) a następnie wciskamy Ctrl+S i wybieramy docelowy układ ATxmega128A4U, następnie wciskamy Ctrl+U i wybieramy przycisk Open, po powyższych czynnościach okno programu FLIP powinno wyglądać jak na poniższym rysunku:

🚮 Atmel Flip	A 10-10-6				
File Buffer Device Settings Help					
🤝 😴 🧳	i 👫 畅 🕹	🔄 🏄 🏄			
Operations Flow	FLASH Buffer Information	ATxmega128A4U			
🕥 🔽 Erase	Size 128 KB	Signature Bytes IE 97 46 00			
	Range 0x0 - 0x0	Device Boot Ids 00 00			
	Checksum 0xFF				
Image: Blank Check	Reset Before Loading	Bootloader Ver. 1.0.4			
Program	HEX File:				
Verify	AIMEL,				
Run	Select EEPROM	Start Application 🔽 Reset			
USB ON					

- wciskamy klawisze Ctrl+L i wybieramy plik HEX, który chcemy wprogramować do mikrokontrolera, w przypadku przykładu z punktu powyższego plik ten znajdowałby się w lokalizacji:
 C:\Designs\AS62\cw 1\cw 1\Debug\cw 1.hex,
- wciskamy klawisz *Run* co powinno spowodować wgranie nowej zawartości do mikrokontrolera,
- po wciśnięciu klawisza *Start Application* układ XMEGA na płytce wychodzi z trybu *Bootloadera* i uruchamia wgrany kod.

```
Zadanie 1. Zapalanie/gaszenie LED.
```

Kod programu prezentowany jest poniżej.

```
/*
* Dioda
 * Program zapala diodę i zapala/gasi drugą przyciskiem PC0
 * Możliwe modyfikacje/zadania: dodanie jednego przycisku, który zgasi palącą się diodę i drugiego,
które ją z powrotem zapali
#include <avr/io.h>
int main(void)
{
        //konfiguracja portu B (tu są podłączone diody)
        PORTB.DIR = PIN0 bm PIN3 bm;
                                      //ustawienie pinów 0 i 3 jako wyjścia (wpisanie 1 do rejestru)
        //zapalenie diody
        PORTB.OUTSET = PIN0_bm;
                                        //ustawienie 1 na pinie 0 (podanie 3.3V)
        //konfiguracja portu C (tu są przyciski)
        PORTC.DIR = ~PIN0 bm;
                                        // ustawienie pinu 0 jako wejście (wpisanie 0 do rejestru)
        PORTC.PINOCTRL = PORT_OPC_PULLUP_gc;
                                                // włączenie pull-up'u na pinie jako, że przycisk po
                                                 // wciśnięciu zwiera do masy i wymusza stan 0
    while(1)
    {
                if (0 == (PORTC.IN & PIN0_bm))
                                                        //gdy przycisk jest wciśnięty (przyciskzwiera
                                                         //pin do masy -> stan 0 za pinie) to:
                        PORTB.OUTTGL = PIN3 bm;
                                                         // zmiana stanu diody
    }
}
```



```
Kod programu prezentowany jest poniżej.
* Dioda migająca
 * Program używa timera w trybie normalnym do włączania/wyłączania diody
 * Możliwe modyfikacje/zadania: zmiana szybkości migania diody; zmiana ilości migających diod ->
napisanie kodu tak, aby diody świeciły naprzemian
 */
#include <avr/io.h>
#include <avr/interrupt.h>
int main(void)
{
        // ustawienie portu i zapalenie diody
        PORTB.DIRSET = PIN0_bm;
        PORTB.OUTSET = PIN0_bm;
        // konfiguracja przerwań
        TCC0.INTCTRLA
                             TC_OVFINTLVL_L0_gc;
                                                        // przepełnienie timera ma generować
                         =
                                                        // przerwanie o niskim priorytecie (LO)
        PMIC.CTRL
                               PMIC_LOLVLEN_bm;
                                                        // odblokowanie przerwań o niskim priorytecie
                          =
                                                        // LO
                                                        // globalne odblokowanie przerwań
        sei();
        // konfiguracja timera
                               TC WGMODE_NORMAL_gc;
        TCC0.CTRLB
                                                        // tryb normalny
                          =
        TCC0.CTRLA
                          =
                               TC_CLKSEL_DIV1024_gc;
                                                        // ustawienie preskalera (podzielenie zegara
                                                        // 2MHz przez 1024) i uruchomienie timera
        TCC0.PER = 1000;
                                                        // ustawienie pojemności licznika na 1000
   while(1)
    {
    }
```

```
}
ISR(TCC0_OVF_vect) {
                                                        // przerwanie pochodzące od przepełnienia TCC0
        PORTB.OUTTGL
                               PIN0_bm;
                                                        // zamiana stanu diody
}
```

Zadanie 3. Przetwornik ADC.

```
Kod programu prezentowany jest poniżej.
/*
 * ADC.c
 * Program zapala diody w miarę, jak napięcie na ADC rośnie.
 * Możliwa modyfikacja/zadanie: diody zapalają się, jak napięcie na ADC maleje/zapalają się w różnej
kolejności
 */
#include <avr/io.h>
#include <avr/interrupt.h>
uint16_t adc_result;
int main(void)
        * porty - konfiguracja
        */
        PORTA.DIRCLR = PIN0_bm;
                                                                   // pin0 portu A jako wejście, tu
                                                                   // podłączony jest potencjometr
        PORTB.DIRSET = PIN0_bm | PIN1_bm | PIN2_bm | PIN3_bm;
                                                                   // piny od 0 do 3 jako wyjścia
        PORTB.OUTCLR = PIN0_bm |PIN1_bm |PIN2_bm |PIN3_bm;
                                                                   // wyłączenie diod
        * konfiguracja ADC
        */
        ADCA.PRESCALER = ADC_PRESCALER_DIV16_gc;
        // 2MHZ/16=125kHz ustalenie zegara dla ADC
        ADCA.CH0.INTFLAGS = 0x01;
        ADCA.INTFLAGS = ADC CH0IF bm;
        // zerowanie flag przerwań
        PMIC.CTRL = PMIC_HILVLEN_bm;
        // odblokowanie przerwań o wysokim priorytecie (ADC ma wysoki
        // priorytet)
        ADCA.CH0.INTCTRL = ADC_CH_INTLVL1_bm | ADC_CH_INTLVL0_bm;
        //włączenie przerwań ADC, domyślnie zgłaszane po zakończeniu konwersji
ADCA.REFCTRL = ADC_REFSEL_INTVCC2_gc;
        // napięcie odniesienia Vcc/2
        ADCA.CTRLB = ADC_CONMODE_bm | ADC_FREERUN_bm;
        // tryb signed (ze znakiem) (aby móc używać gain'a), freerun - pomiary wykonują się ciągle
        ADCA.CH0.CTRL = ADC_CH_GAIN_DIV2_gc | ADC_CH_INPUTMODE_DIFFWGAIN_gc;
        // wzmocnienie kanału 0 równe 0.5, aby dopasować napięcie 3.3 do referencyjnego oraz
        // ustawienie pomiaru różnicowego ze wzmocnieniem
        ADCA.CH0.MUXCTRL = ADC_CH_MUXPOS_PIN0_gc | ADC_CH_MUXNEG_INTGND_MODE4_gc;
        // podłączenie wew. GND do wejścia ujemnego ADC
        ADCA.CTRLA = ADC_ENABLE_bm;
        // włączenie ADC
        sei();
        // odblokowanie przerwań
        /*
        .
* MAIN
        */
        while(1)
        {
                cli();
                // W zależności od wyniku zapalane są różne diody, wartości dobrane tak, aby zapalały
```

{

```
// się mniej więcej "równomiernie"
                 if (adc_result < 400 || adc_result > 2048)
                         // adc_result > 2048 wynika z tego, że masa wewnętrzna podłączona do wejścia
                         // negatywnego ADC
                         {
                         \dot{//} ma większy potencjał od masy układu i po ustawieniu potencjometru na
                         // najwyższą wartość rezystancji
                                 PORTB.OUTCLR = PIN0_bm |PIN1_bm |PIN2_bm |PIN3_bm;
                                 // wynik czasami miał wartości ujemne (pojawiała się 1 na 12 bicie
                                 //wyniku i zapalały się wszystkie
                         }
                                 // diody - program wchodził w ostatniego if'a)
                 else if (adc_result > 400 && adc_result < 800)</pre>
                 {
                         PORTB.OUTSET = PIN0_bm;
                         PORTB.OUTCLR = PIN1_bm | PIN2_bm | PIN3_bm;
                 }
                 else if (adc_result > 800 && adc_result < 1200)
                 {
                         PORTB.OUTSET = PIN0_bm | PIN1_bm;
                         PORTB.OUTCLR = PIN2_bm|PIN3_bm;
                 }
                 else if (adc_result > 1200 && adc_result < 1600)</pre>
                 {
                         PORTB.OUTSET = PIN0_bm | PIN1_bm | PIN2_bm;
                         PORTB.OUTCLR = PIN3_bm;
                 }
                 else if (adc_result > 1600)
                         PORTB.OUTSET = PIN0_bm |PIN1_bm |PIN2_bm |PIN3_bm;
                 sei();
    }
}
* obsługa przerwania
*/
        ISR(ADCA CH0 vect) {
                 adc_result = ADCA.CH0.RES & 0b000011111111111; // odczytanie wyniku, maska użyta,
                                                                   // gdyż 12 bitowy wynik zapisany jest
                                                                   // w 16 bitowym rejestrze,
                                                                   // gdzie niekoniecznie są 0 na
                                                                   // ostatnich 4 bitach
        }
```

Przygotowali: Bogdan Pankiewicz Adam Popik Gdańsk 17.03.2015