Instrukcja Laboratorium Systemów Wbudowanych

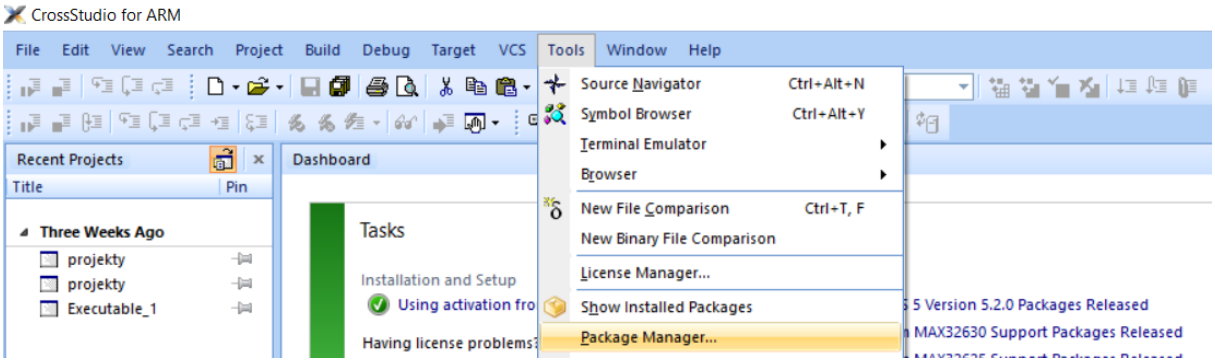
Lab 1.

Tworzenie nowego projektu, obsługa PIO

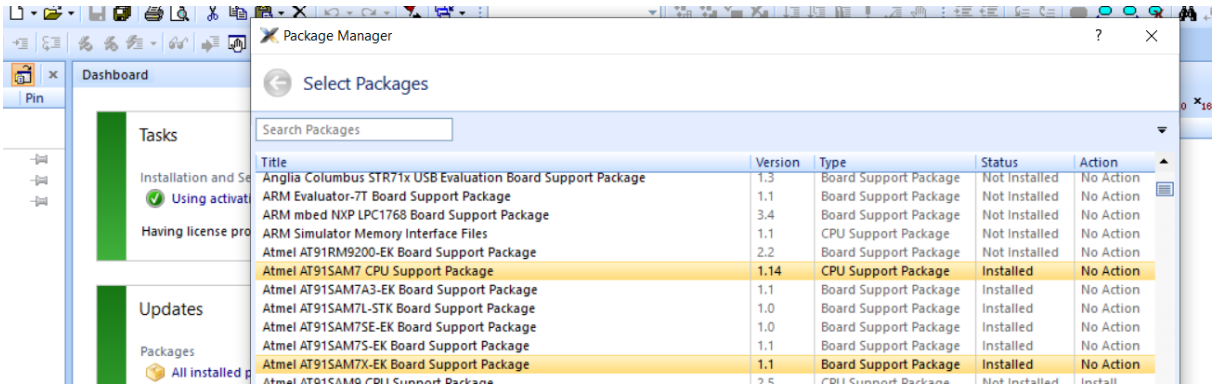
1. Tworzenie nowego projektu

W związku z zaistniałą sytuacją epidemiologiczną, proponuję Państwu pobranie oprogramowania CrossWorks firmy Rowley w okrojonej formie, dostępnej bez opłat na stronie www producenta.

Po pobraniu oprogramowania, pierwszym etapem koniecznym do rozpoczęcia pracy z płytką Olimexu dostępną na laboratorium, wyposażoną w mikrokontroler SAM7X256 jest pobranie tzw. suport package współpracujących z wybranymi typami mikrokontrolerów.



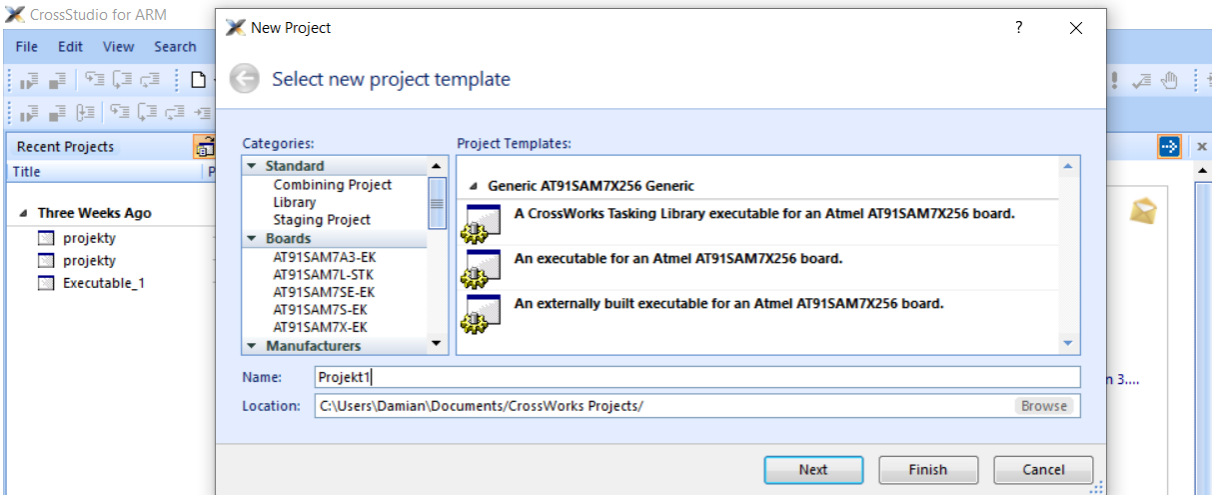
Wybierając z menu Tools -> package menager -> wybieramy dwie opcje:



klikając dwa razy na napis: No Action zmienicie status na Install – przechodząc dalej przyciskiem Next doprowadzicie do pobrania i zainstalowania odpowiednich paczek.

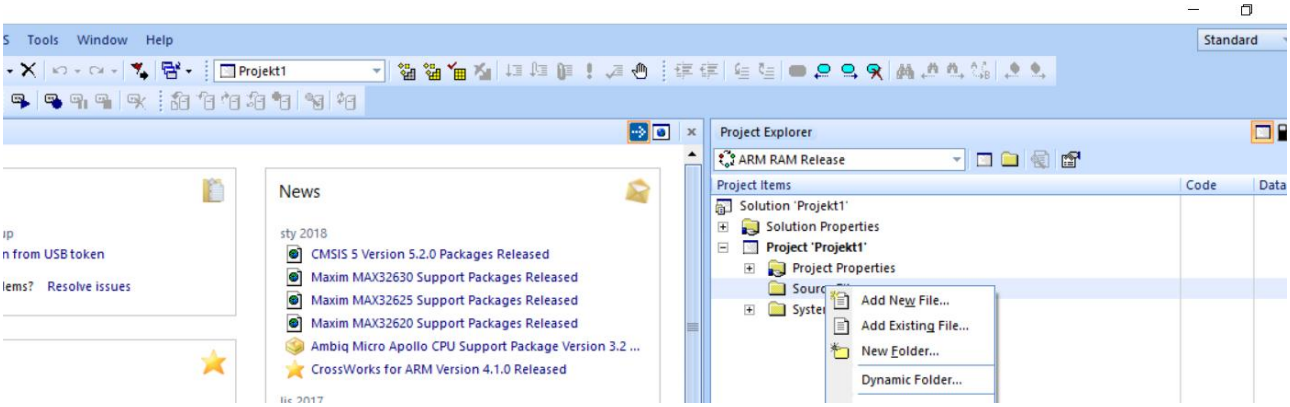
W tym momencie można przejść do tworzenia nowego projektu w sposób, jaki poznaliście na pierwszym oraz drugim spotkaniu jakie odbyliśmy w ramach laboratoriów przed wprowadzeniem stanu zagrożenia epidemicznego.

File -> New Project ->

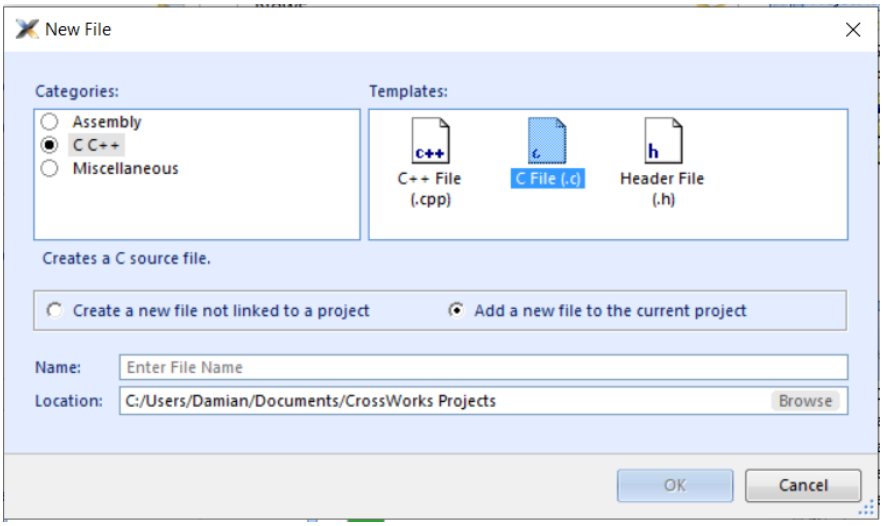


Wybieramy opcję: **An executable for an Atmel AT91SAM7X256 board**. W zależności od wersji oprogramowania, okno może wyglądać inaczej (mam nadzieję, iż pamiętacie Państwo, że okno w wersji dostępnej w Lab 309 wyglądało nieco inaczej) – nie zmienia to faktu iż należy doprowadzić do wyboru odpowiedniej opcji. Dalej przechodząc przyciskiem Next aż do momentu w którym dostaniecie Państwo 8 opcji wyboru oznaczone znacznikami, proszę pozostawić zaznaczoną jedynie opcję: ☑ ARM RAM Release

Tworzenie pliku .c programu.
W oknie Project Explorer należy na **Source Files** kliknąć prawym przyciskiem myszy i wybrać opcję **Add New File**.



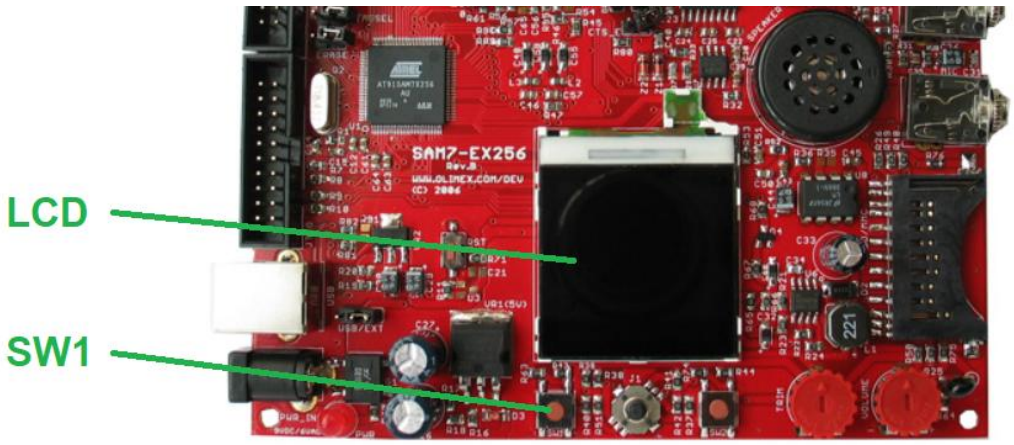
W oknie które się pojawiło, wybieramy C file, nadajemy nazwę plikowi i tworzymy go.



W oknie głównym pojawi się pusty plik w którym można wpisać kod programu. Środowisko automatycznie dołącza do projektu pliki rozruchowe dla odpowiedniej rodziny procesorów ARM – można je obejrzeć w drzewie projektu.

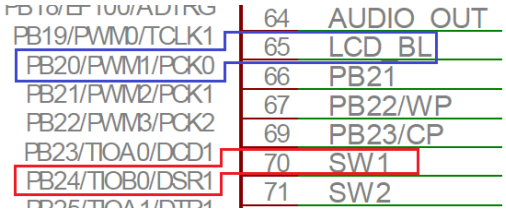
2. Tworzenie programu.

Zadaniem programu jest włączenie podświetlenia wyświetlacza LCD (pracującego jak dioda LED). Program ma potem oczekiwać na wciśnięcie przycisku SW1 który zgasi podświetlenie.

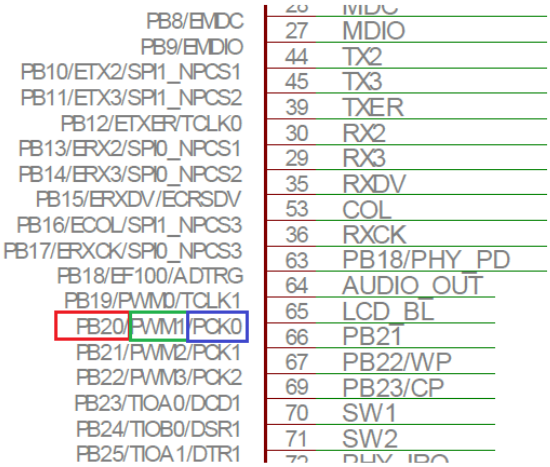


Realizację zadania należy zacząć od analizy schematu elektronicznego płytki ewaluacyjnej dostępnego na stronie: https://www.olimex.com/Products/ARM/Atmel/SAM7-EX256/resources/SAM7-EX256-sch.gif

Podłączenie podświetlenia wyświetlacza do PINu mikrokontrolera oznaczone jest etykietą LCD_BL. **[PB20]**
Podłączenie przycisku po lewej od joysticka jest oznaczone jako SW1. **[PB24]**



Oznaczenie PB20 oznacza Parallel Input Output Controller B linia 20. Pozostałe oznaczenia związane są z funkcjami peryferyjnymi wybranych PINów. PIN 65 może pracować jako linia I/O ogólnego przeznaczenia, bądź linia wyjściowa kanału 1 PWM lub PCK0. Jest to tak zwane multipleksowanie funkcji PINu.



(electronic scheme of board)

10.5 PIO Controller B Multiplexing

Table 10-3. Multiplexing on PIO Controller B

PIO Controller B			
I/O Line	Peripheral A	Peripheral B	Com
PB0	ETXCK/EREFCK	PCK0	
PB1	ETXEN		
PB2	ETX0		
PB18	EF100	ADTRG	
PB19	PWM0	TCLK1	
PB20	PWM1	PCK0	
PB21	PWM2	PCK1	
PB22	PWM3	PCK2	
PB23	TIOA0	DCD1	
PB24	TIOB0	DSR1	
PB25	TIOA1	DTR1	

(page 33 datasheet)

Dodatkowo konieczne jest zapoznanie się z notą katalogową mikrokontrolera Atmel SAM7X256.
Rozdział 27 (strona 219 noty katalogowej) dotyczy układów Parallel Input Output Controller. Przy realizacji zadania, szczególnie potrzebne jest zaznajomienie się z podrozdziałami: 27.3.1, 27.3.3, 27.4.2, 27.4.4.

Aby program mógł korzystać z definicji rejestrów zawartych procesorze należy na początku dodać linijkę dodająca te definicje:

```
#include <targets/AT91SAM7.h>


int main (void){
PMC_PCER = 1<<3;        //załączenie zegara systemowego do wybranego peryferium (w tym przypadku PIOB). Zapoznać się z rozdziałem 25.5 str. 173 oraz peripheral identifiers str. 30
PIOB_PER = 1<<20;       //załączenie kontroli kontrolera I/O nad wybranym PINem – w tym przypadku PIN 65 będzie kontrolowany przez linię 20 kontorlera I/O B  27.4.2
PIOB_PER |= 1<<24;      // j.w.
PIOB_OER = 1<<20;       // linia 20 portu B pracuje jako wyjście 27.4.4
PIOB_ODR = 1<<24;       // linia 24 portu B pracuje jako wejściowa  27.4.4
PIOB_SODR = 1<<20;      // stan linii 20 portu B ustawiony na 1 (na PINie 65 ustawiony został stan wysoki)

while(1){               //pętla nieskończona tak, by program oczekiwał na wciśnięcie przycisku
if ( ( PIOB_PDSR & 1<<24 ) == 0 ){      // sprawdzenie stanu linii 24 Portu B (PIN 70), jeśli na linii 24 jest stan wysoki – na bicie 24 rejestru PDSR pojawia się 1, jeśli stan jest linii jest niski – na bicie pojawia się 0
    PIOB_CODR = 1<<20;      // stan linii 20 portu B ustawiony na 1 (na PINie 65 ustawiony został stan niski)
}
 }
  }
```

Rozdział peripheral identifiers ze strony 30 noty katalogowej zawiera tabelę identyfikującą wybrany układ peryferyjny z jego „Identifierem”.

Table 10-1. Peripheral Identifiers

Peripheral ID	Peripheral Mnemonic	Peripheral Name	External Interrupt
0	AIC	Advanced Interrupt Controller	FIQ
1	SYSC(1)	System Controller	
2	PIOA	Parallel I/O Controller A	
3	PIOB	Parallel I/O Controller B	
4	SPI0	Serial Peripheral Interface 0	
5	SPI1	Serial Peripheral Interface 1	
6	US0	USART 0	
7	US1	USART 1	

25.9.4 PMC Peripheral Clock Enable Register

Register Name: PMC_PCER

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	-	-

- **PIDx: Peripheral Clock x Enable**

0 = No effect.

1 = Enables the corresponding peripheral clock.

Note: PID2 to PID31 refer to identifiers as defined in the section "Peripheral Identifiers" of the product datasheet.

Note: Programming the control bits of the Peripheral ID that are not implemented has no effect on the behavior of the PMC.

Następnie, jeśli chcesz użyć PIN 65 jako linii kontrolera wejścia / wyjścia, należy włączyć tę funkcję w rejestrze PIOx_PER. Warto zauważyć, że po zresetowaniu jest to funkcja domyślna dla PINów które mogą być kontrolowane przez linie I/O.

27.6.1 PIO Controller PIO Enable Register

Name: PIO_PER

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: PIO Enable**

0 = No effect.

1 = Enables the PIO to control the corresponding pin (disables peripheral control of the pin).

Zauważcie, że “0” oznacza no effect.

27.6.4 PIO Controller Output Enable Register

Name: PIO_OER

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Enable**

0 = No effect.

1 = Enables the output on the I/O line.

27.6.10 PIO Controller Set Output Data Register

Name: PIO_SODR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Set Output Data**

0 = No effect.

1 = Sets the data to be driven on the I/O line.

Rozdziały noty katalogowej do których odnosi się przykład

27.4.2 I/O Line or Peripheral Function Selection

When a pin is multiplexed with one or two peripheral functions, the selection is controlled with the registers PIO_PER (PIO Enable Register) and PIO_PDR (PIO Disable Register). The register PIO_PSR (PIO Status Register) is the result of the set and clear registers and indicates whether the pin is controlled by the corresponding peripheral or by the PIO Controller. A value of 0 indicates that the pin is controlled by the corresponding on-chip peripheral selected in the PIO_ABSR (AB Select Status Register). A value of 1 indicates the pin is controlled by the PIO controller.

If a pin is used as a general purpose I/O line (not multiplexed with an on-chip peripheral), PIO_PER and PIO_PDR have no effect and PIO_PSR returns 1 for the corresponding bit.

After reset, most generally, the I/O lines are controlled by the PIO controller, i.e. PIO_PSR resets at 1. However, in some events, it is important that PIO lines are controlled by the peripheral (as in the case of memory chip select lines that must be driven inactive after reset or for address lines that must be driven low for booting out of an external memory). Thus, the reset value of PIO_PSR is defined at the product level, depending on the multiplexing of the device.

27.4.4 Output Control

When the I/0 line is assigned to a peripheral function, i.e. the corresponding bit in PIO_PSR is at 0, the drive of the I/O line is controlled by the peripheral. Peripheral A or B, depending on the value in PIO_ABSR, determines whether the pin is driven or not.

When the I/O line is controlled by the PIO controller, the pin can be configured to be driven. This is done by writing PIO_OER (Output Enable Register) and PIO_ODR (Output Disable Register). The results of these write operations are detected in PIO_OSR (Output Status Register). When a bit in this register is at 0, the corresponding I/O line is used as an input only. When the bit is at 1, the corresponding I/O line is driven by the PIO controller.

The level driven on an I/O line can be determined by writing in PIO_SODR (Set Output Data Register) and PIO_CODR (Clear Output Data Register). These write operations respectively set and clear PIO_ODSR (Output Data Status Register), which represents the data driven on the I/O lines. Writing in PIO_OER and PIO_ODR manages PIO_OSR whether the pin is configured to be controlled by the PIO controller or assigned to a peripheral function. This enables configuration of the I/O line prior to setting it to be managed by the PIO Controller.

Similarly, writing in PIO_SODR and PIO_CODR effects PIO_ODSR. This is important as it defines the first level driven on the I/O line.

27.4.5 Synchronous Data Output

Controlling all parallel busses using several PIOs requires two successive write operations in the PIO_SODR and PIO_CODR registers. This may lead to unexpected transient values. The PIO controller offers a direct control of PIO outputs by single write access to PIO_ODSR (Output Data Status Register). Only bits unmasked by PIO_OWSR (Output Write Status Register) are written. The mask bits in the PIO_OWSR are set by writing to PIO_OWER (Output Write Enable Register) and cleared by writing to PIO_OWDR (Output Write Disable Register).

After reset, the synchronous data output is disabled on all the I/O lines as PIO_OWSR resets at 0x0.

27.4.8 Inputs

The level on each I/O line can be read through PIO_PDSR (Pin Data Status Register). This register indicates the level of the I/O lines regardless of their configuration, whether uniquely as an input or driven by the PIO controller or driven by a peripheral.

Reading the I/O line levels requires the clock of the PIO controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.