LABORATORIUM nr 6

Temat: ARM 7 – obsługa wyświetlacza LCD

1. ZESTAW URUCHOMIENIOWY MCB2300

Zestaw uruchomieniowy MCB2300 oparty jest na mikrokontrolerze LPC2378 zawierającym procesor ARM7TDMI-S, realizowanym w 32-bitowej architekturze RISC. Architekturę mikrokontrolera uzupełnia 512 kB systemowej pamięci flash oraz 64 kB statycznej pamięci RAM dostępnej dla procesora. Dodatkowo mikrokontroler zawiera 16 kB statycznej pamięci RAM dla interfejsu Ethernet oraz 8 kB statycznej pamięci RAM dostępnej dla interfejsu USB. System przerwań AVIC (Advanced Vectored Interrupt Controller) obsługuje do 32 przerwań. Oprócz wymienionych interfejsów mikrokontroler wspiera interfejs szeregowy SSP , port I^2S, oraz port kart SD/MMC. Uzupełnieniem wspieranych portów jest 10 bitowy przetwornik A/D oraz 10 bitowy przetwornik D/A, cztery układy czasowe (timery) Budowę strukturalną mikrokontrolera pokazano na rysunku 1.
Architektura mikrokontrolera określona jest wokół dwóch magistrali: zaawansowaną magistralę wysokiej wydajności AHB, służącą do szybkiej komunikacji z pamięcią zewnętrzną oraz wbudowanymi peryferiami, oraz zaawansowaną magistralę peryferyjną APB, służącą do komunikacji z pozostałymi urządzeniami peryferyjnymi.
W mikrokontrolerze zastosowano konwencję **little-endian**.
Procesor ARM posiada 4 GB przestrzeń adresową. Przestrzeń ta została zagospodarowana w sposób pokazany w tabeli 1. Mapa pamięci została pokazana na rysunku 2.

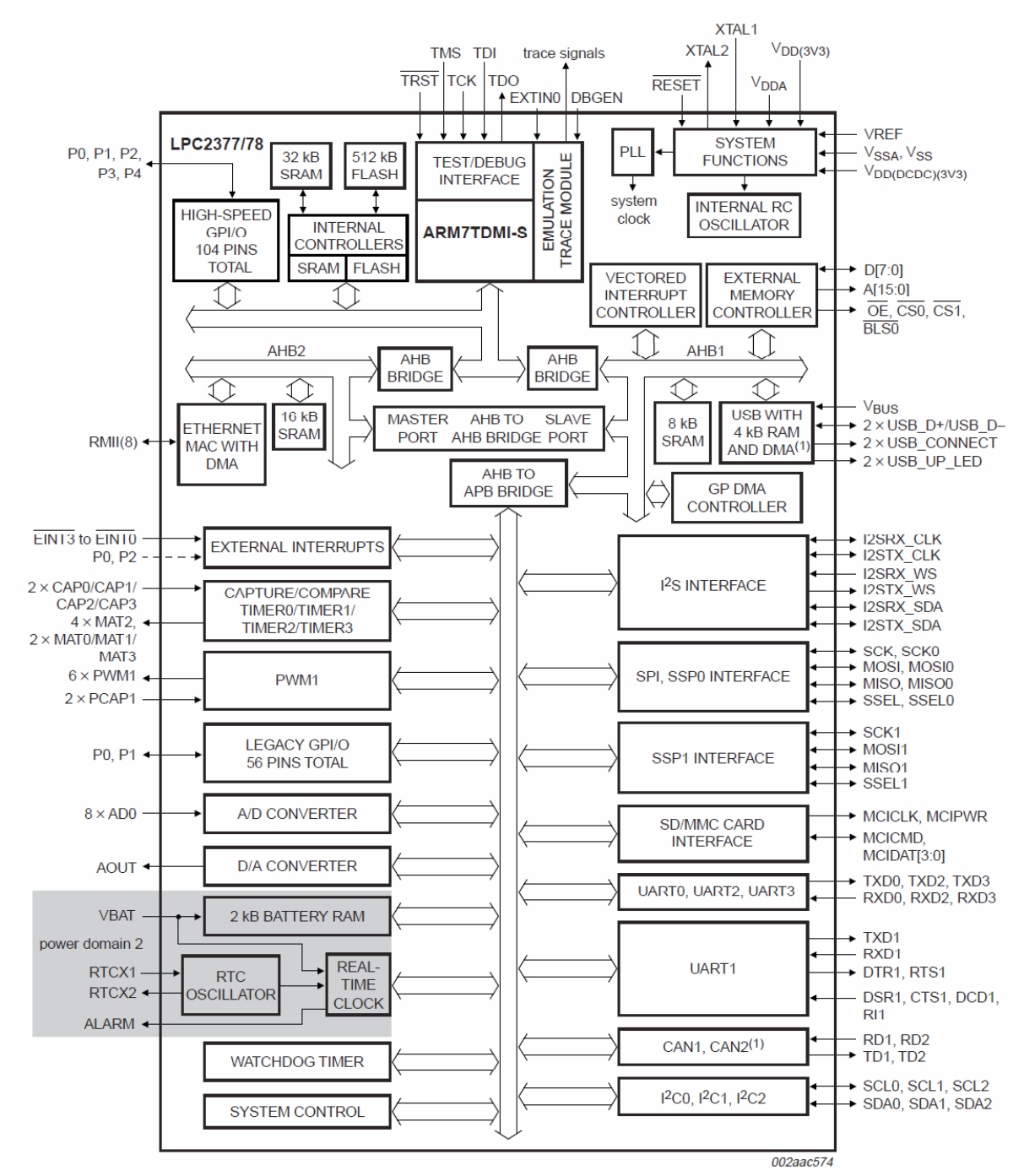
Tabela. 1: Wykorzystanie pamięci mikrokontrolera LPC2378

Zakres adresów	Zastosowanie	Detale zakresu	Opis
0x0000 0000 – 0x3FFF FFFF	Pamięć systemowa oraz dla szybkich portów I/O	0x0000 0000 – 0x0007 FFFF 0x3FFF C000 – 0x3FFF FFFF	512 kB flash szybkie rejestry GPIO
0x4000 0000 – 0x7FFF FFFF	RAM wewnętrzny	0x4000 0000 – 0x4000 7FFF 0x7FD0 0000 – 0x7FD0 1FFF 0x7FE0 0000 – 0x7FE0 3FFF	RAM (do 32 kB) USB RAM Ethernet RAM
0xF000 0000 – 0xFFFF FFFF	Peryferia AHB	0xFFE0 0000 – 0xFFE0 3FFFF 0xFFE0 4000 - 0xFFE0 7FFF 0xFFE0 8000 - 0xFFE0 BFFF 0xFFE0 C000 - 0xFFE0 FFFF 0xFFFF F000 - 0xFFFF FFFF	Kontroler Ethernet Kontroler DMA Kontroler pam. zew. Kontroler USB VIC

Zestaw uruchomieniowy należy podłączyć do komputera w następujących krokach:

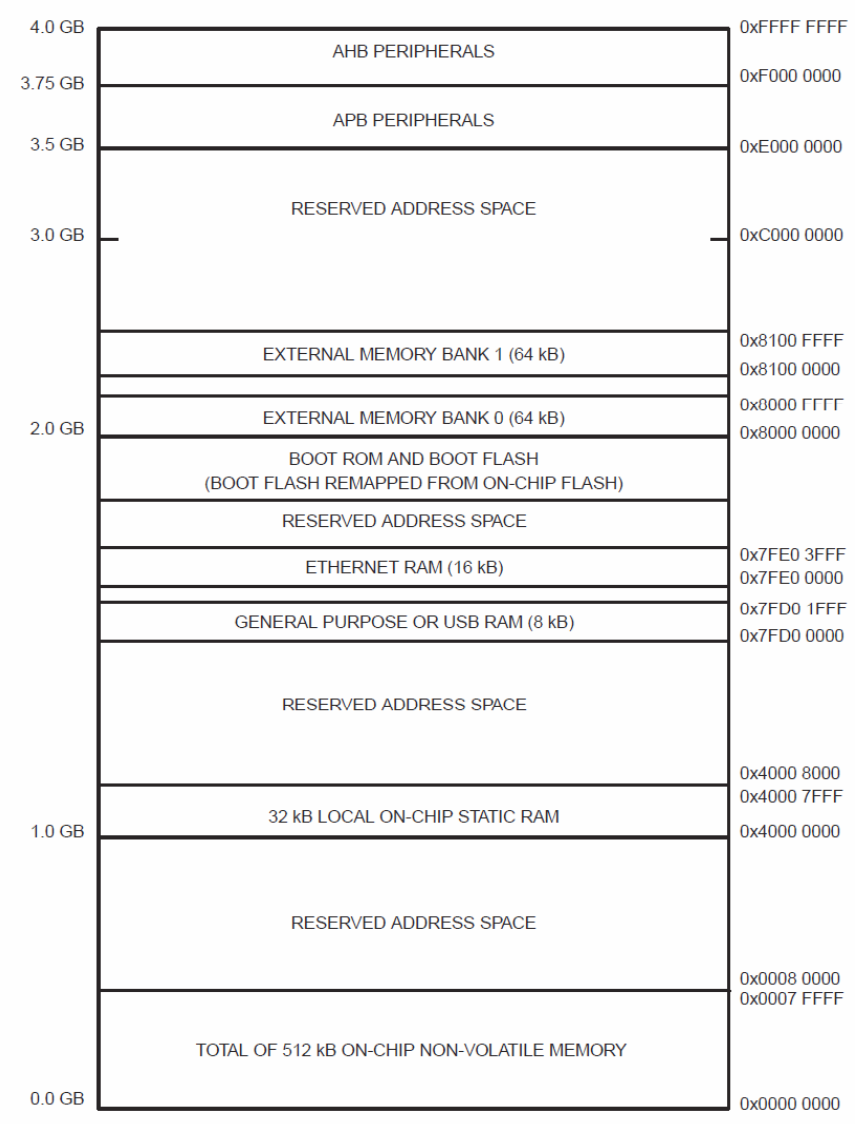
- Adapter ULINK2 USB-JTAG należy połączyć przewodem USB w wolnym portem USM komputera;
- Połączyć adapter z przyłączem JTAG, znajdującym się na płytce drukowanej zestawu uruchomieniowego;
- Połączyć zestaw uruchomieniowy z komputerem za pomocą drugiego kabla USB. Połączenie to służy do zasilenia zestawu uruchomieniowego. Po wykonaniu tego kroku powinna się świecić dioda Power.

Wszelkie dodatkowe informacje dotyczące zestawu uruchomieniowego, a w szczególności jego konfiguracji, znajdują się w dokumencie *MCB2300 User's Guide*, który jest dostępny w Keil\ARM\Hlp, plik mcb2300.chm.
Do dyspozycji użytkownika oddane są diody LED, wyświetlacz LCD, dwa przyciski, potencjometr, głośniczek, zestaw portów COM oraz CAN, port Ethernet, gniazdo kart SD/MMC, oraz port USB pełniący dwie funkcje – zasila zestaw uruchomieniowy, oraz transmituje dane do/z komputera.



Rys. 1. Budowa strukturalna mikrokontrolera LPC2378

Procesor ARM7TDMI-S posiada 144 wyprowadzenia (piny). Funkcje poszczególnych pinów są konfigurowalne, a pojedynczy pin może pełnić od dwóch do czterech funkcji (patrz rysunek 3 oraz tabela 98 w [1]). Standardowo wykorzystywana jest funkcja pinów jako 32 bitowych portów nadawczo/odbiorczych z indywidualną kontrolą kierunku pracy portu (nadawanie lub odbiór). Porty te są oznaczone jako P0.[0-31] (port 0), dla portu 1 nie są dostępne piny 2, 3, 5, 6, 7, 11, 12 i 13., dla portu 2 nie są dostępne piny od 14 do 31, dla portu 3 nie są dostępne piny od 8 do 22 oraz od 27 do 31, zaś dla portu 4 nie są dostępne piny od 16 do 23, 26 i 27.



Rys. 2. Mapa pamięci mikrokontrolera LPC2378

W przypadku pinów spełniających kilka funkcji, wyboru funkcji dokonujemy za pomocą rejestrów **PINSEL**. Dla omawianego mikrokontrolera zdefiniowane rejestry PINSEL0 do PINSEL4 oraz PINSEL6 do PINSEL10, np. bity 0 i 1 rejestru PINSEL1 opisują pin P0.16, w tabeli 2 pokazano ustawienia tych bitów i przypisane im funkcje.

Tabela. 2: Przykład zastosowania rejestru PINSEL1 dla bitów 1:0

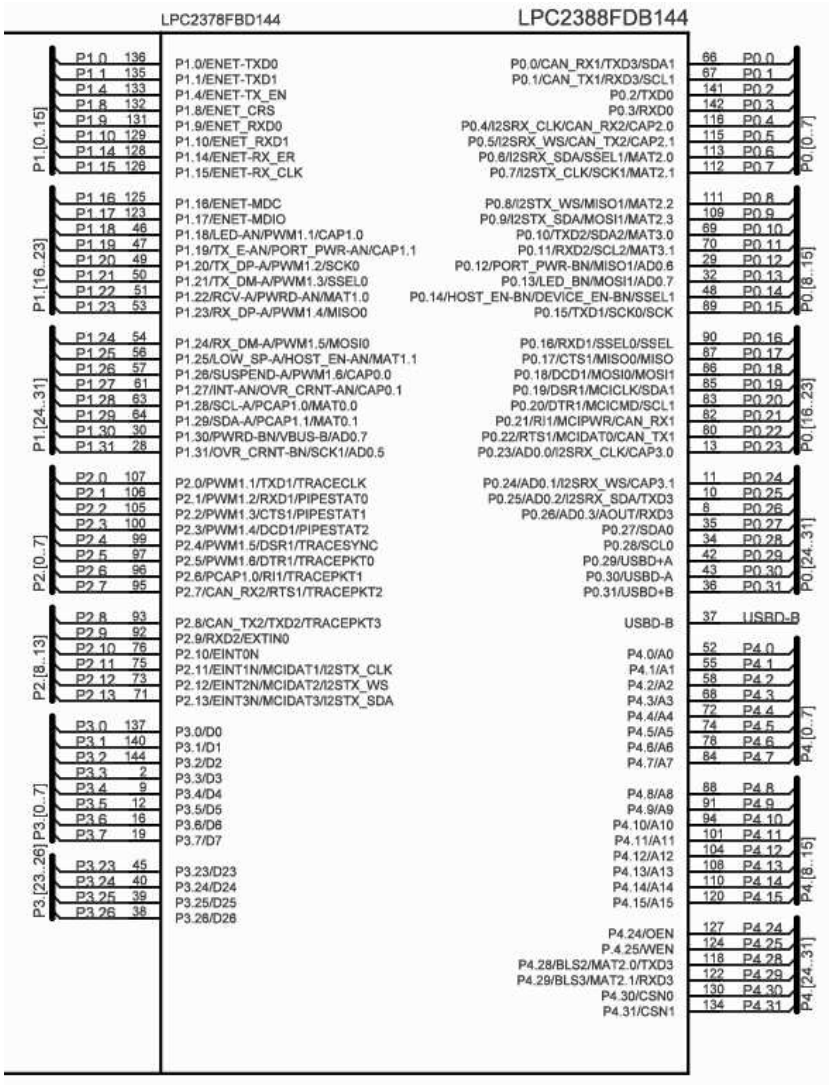
PINSEL1	Nazwa	Funkcja dla 00	Funkcja dla 01	Funkcja dla 10	Funkcja dla 11	Wartość po resecie
1:0	P0.16	GPIO port 0.16	RXD1 (wejście dla poru szeregowego)	SSEL0 (ramka synchronizująca lub wybór dla urządzenia podrzędnego dla szybkiej magistrali szeregowej nr 0 (SSP)	SSEL (wybór urządzenia podrzędnego dla magistrali interfejsu szeregowego SPI	00

Szczegółowe ustawienia poszczególnych rejestrów PINSEL omówiono w dokumentacji [1] w tabelach 9-106, 9-108, 9-109, 9-111, 9-113, 9-114, 9-116, 9-117, 9-119, 9-120. Po resecie wartości rejestrów PINSEL są zerowane,

czyli najczęściej piny pełnią funkcję portów ogólnego przeznaczenia GPIO (General Purpose Input/Output). LPC2300 posiada pięć 32-bitowych portów GPIO. PORT0 oraz PORT1 są sterowane przez dwie grupy rejestrów omówionych poniżej. Porty PORT2, PORT3 i PORT4 stanowią porty dodatkowe. W tabeli 3 omówiono rejestry sterujące pracą portów PORT0 i PORT1.

Tabela. 3: Opis rejestrów sterujących pracą portów PORT0 I PORT1

Nazwa	Opis	Dostęp	Reset	Rejestr i adres
IOPIN	Z tego rejestru odczytywana jest aktualna konfiguracja portu GPIO, w szczególności kierunek przepływu danych (wejście/wyjście). Każdy z pinów ustawiany jest niezależnie.	R/W	Brak	IO0PIN – 0xE002 8000 IO1PIN – 0xE002 8010
IOSET	Rejestr ten kontroluje stan pinów wyjściowych w połączeniu z rejestrem IOCLR. Zapisanie 1 powoduje pojawienie się stanu wysokiego na pinie.	R/W	0x0	IO0SET – 0xE002 8004 IO1SET – 0xE002 8014
IODIR	Rejestr służy do indywidualnej kontroli kierunku każdego portu	R/W	0x0	IO0DIR – 0xE002 8008 IO1DIR – 0xE002 8018
IOCLR	Kontroluje stan pinów wyjściowych. Zapisanie 1 powoduje wywołanie stanu niskiego na danym pinie portu i wyczyszczenie jego wartości w rejestrze IOSET.	WO	0x0	IO0CLR – 0xE002 800C IO0CLR – 0xE002 801C

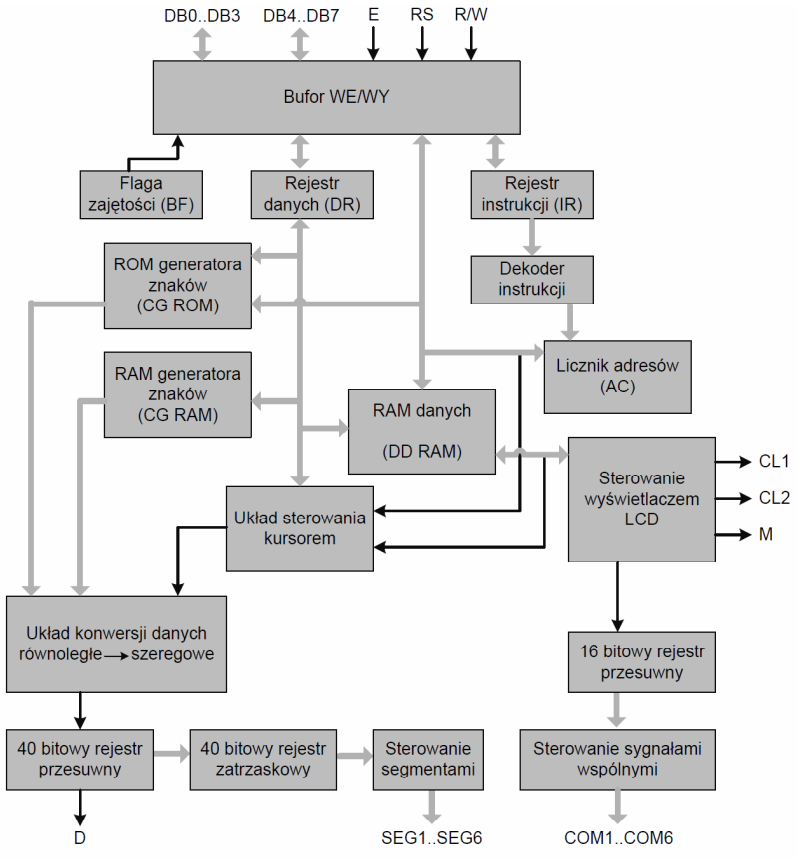


Rys. 3. Schemat wyprowadzeń wraz z opisami dla ARM7TDMI-S [1]

2. OBSŁUGA WYŚWIETLACZA LCD

Wyświetlacz ciekłokrystaliczny typ AC162BYA zastosowany w ćwiczeniu jest wyświetlaczem matrycowym zawierającym moduł kontrolera i układ wykonawczy wykonany w technologii LSI, pozwalający wyświetlać znaki alfanumeryczne i symbole graficzne. Wyświetlacz może współpracować z mikrokomputerem jednoukładowym lub mikroprocesorem z szyną danych cztero- lub ośmiobitową. Wyświetlacz wyposażony jest również w wewnętrzną pamięć RAM (80 bajtów) i ROM (która zawiera matryce 5×7 punktów lub 5×10 punktów dekodowanych znaków). Na rysunku 3 pokazano schemat blokowy wyświetlacza LCD, a w tabeli 4 pokazano opis styków wyświetlacza.
Bloki pokazane na rysunku 4 pełnią następujące funkcje:
***Rejestr instrukcji IR**:* rejestr ośmiobitowy przechowujący instrukcje sterujące, informację o adresach wewnętrznej pamięci danych RAM (DD RAM) oraz pamięci RAM generatora znaków (CG RAM). *Do tego rejestru można jedynie zapisywać dane*.
***Rejestr danych DR**:* rejestr ośmiobitowy chwilowo przechowujący dane zapisywane lub odczytywane do/z DD RAM lub CG RAM. Dane wpisywane do rejestru **DR** są automatycznie przepisywane do pamięci danych DD RAM lub pamięci znaków CG RAM przez operację wewnętrzną. Rejestr **DR** jest także wykorzystywany do przechowywania danej podczas operacji czytania danych z pamięci DD RAM lub CG RAM. Po zapisaniu adresu do rejestru **IR** dana jest przepisywana do rejestru **DR** z pamięci DD lub CG przez operację wewnętrzną.

Po odczycie przez MPU danej z rejestru DR, do rejestru DR przesyłana jest dana z komórki pamięci DD lub CG o adresie zwiększonym o 1. Przy pomocy sygnału RS dokonywany jest wybór między rejestrem **IR** i **DR**.



Rys. 4. Schemat blokowy wyświetlacza LCD

***Flaga zajętości (BF)**:* kiedy przyjmuje ona wartość "1", wyświetlacz znajduje się w trybie wykonywania operacji wewnętrznej i następna instrukcja nie będzie zaakceptowana. Flaga zajętości jest wystawiana jako bit DB7 (dla RS = "0", R/W = "1"). Następna instrukcja może być wpisana po stwierdzeniu, że BF = "0".
***Pamięć wyświetlanych danych (DD RAM)**:* Pamięć wyświetlanych danych przechowuje dane w postaci 8-mio bitowych kodów. Jej pojemność wynosi 80×8 bitów (80 znaków). Ta część pamięci, która nie jest wykorzystywana do wyświetlania może być użyta jako RAM ogólnego przeznaczenia. Zależność między adresami DD RAM i położeniem znaku na wyświetlaczu LCD pokazana jest poniżej. Adres DD RAM (ADD) jest ustawiany w liczniku adresów (AC) i ma postać binarną.
Pamięć znaków ROM (CG ROM): generator ten wytwarza wzory 5×7 lub 5×10 pikseli odpowiadające wyświetlanym 8-mio bitowym danym. Wzory znaków dla obydwu typów reprezentacji podano w tabelach przedstawiających zestawy znaków.
***Pamięć znaków RAM (CG RAM)**:* pamięć ta pozwala na zdefiniowanie własnego zestawu znaków, poprzez wpisanie odpowiednich wzorów 5×7 lub 5×10 pikseli.
***Blok sterowania wyświetlaczem LCD**:* blok ten zawiera 16 wzmacniaczy sterujących liniami wspólnymi i 40 wzmacniaczy sterujących segmentami. Po wybraniu przez program generatora znaków i liczby linii znakowych następuje automatyczna selekcja wzmacniaczy sterujących liniami wspólnymi. Matryce znaków są przesyłane szeregowo przez rejestr 40- bitowy i zatrzaskiwane po przesłaniu wszystkich znaków. Zatrzaśnięte dane sterują wzmacniaczem wyjściowym wytwarzającym odpowiedni kształt sygnału.

Tabela. 4: Opis wyprowadzeń wyświetlacza ciekłokrystalicznego

Nr styku	Nazwa	Poziom	Opis
1	VSS		Masa
2	VDD		Napięcie +5V
3	V0		Kontrast ekranu
4	RS	0/1	0 – kod instrukcji; 1 – dana
5	R/W	0/1	0 – wpisanie danej; 1 – czytanie danej
6	E	1->0	Impuls zapisu/odczytu
7	DB0	0/1	Linie danych
8	DB1	0/1	
9	DB2	0/1	
10	DB3	0/1	
11	DB4	0/1	
12	DB5	0/1	
13	DB6	0/1	
14	DB7	0/1	
15	VLED		Podświetlenie
16	NC		

***Blok sterowania kursorem**:* blok ten wytwarza kursor lub powoduje jego migotanie. Kursor pojawia się na pozycji wyznaczonej stanem licznika adresów DD RAM. Poniżej pokazano przykładowe położenie kursora dla stanu licznika 7 heksadecymalnie:

AC6	AC5	AC4	AC3	AC2	AC1	AC0
0	0	1	0	0	0	0

Praca w trybie jednowierszowym

1	2	3	4	5	6	7	8	9		40	
00	01	02	03	04	05	06	07	08	....	27	- adres DD RAM

- położenie znaku
- położenie kursora

Praca w trybie dwuwierszowym

1	2	3	4	5	6	7	8	9		40	
00	01	02	03	04	05	06	07	08	....	27	- adres DD RAM
40	41	41	43	44	45	46	47	48	....	67	- adres DD RAM

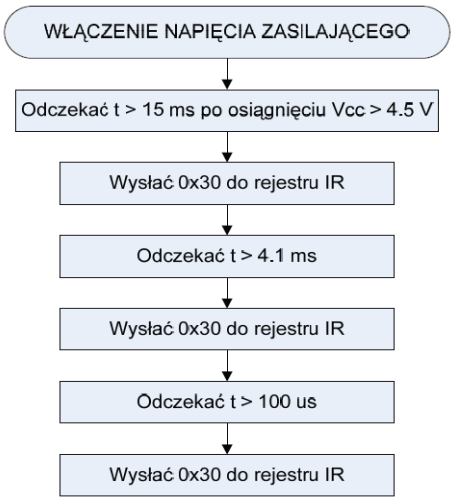
- położenie znaku

Zastosowany w zestawie uruchomieniowym wyświetlacz obsługuje 2 wiersze po 16 znaków.
Sterownik wyświetlacza zajmuje dwa słowa w pamięci RAM:
0x1F90 COMM_LCD Przy zapisie - adres rejestru instrukcji (**IR**), przy odczycie zwraca bajt, zawierający bit flagi zajętości (BF – bit 7) oraz siedmiobitowy adres pozycji znaku (bity 6..0)
0x1F91 DATA_LCD Adres rejestru danych (**DR**)

3. PROGRAMOWANIE WYŚWIETLACZA LCD

Inicjalizacja wyświetlacza odbywa się według następujących kroków:

1. Reset wyświetlacza po włączeniu zasilania (flaga zajętości jest ustawiana w stan BF='1' i może być testowana dopiero po wysłaniu pierwszego słowa operacyjnego). Wpisywanie sekwencji instrukcji wg powyższego schematu jest konieczne w przypadku szybkiego wysyłania instrukcji programujących (np. w momencie inicjalizacji systemu mikroprocesorowego).



2. Wysłanie słowa operacyjnego, ustawiającego parametry wyświetlacza.

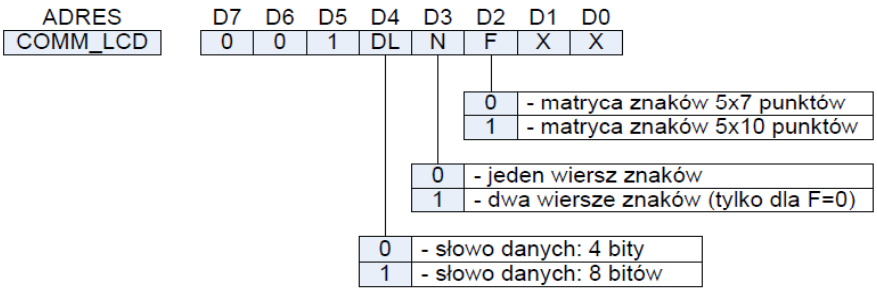
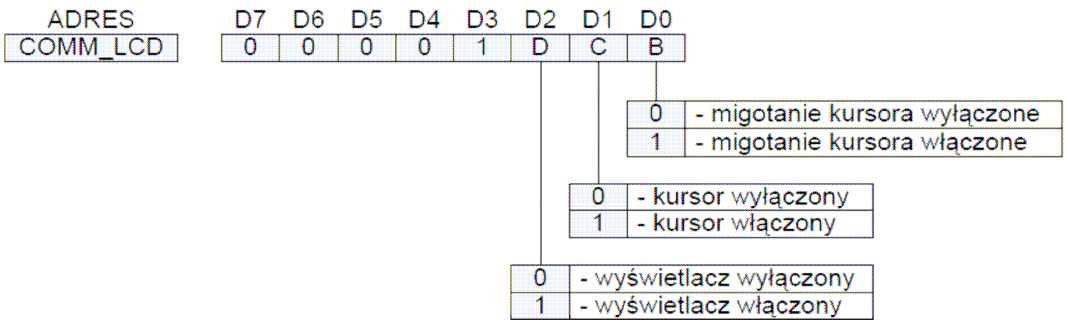


Tabela. 5: Zestaw instrukcji możliwych do przesłania do mikrokontrolera wyświetlacza LCD

Adres	Kod (DB7 – DB0)	Opis
COMM_LCD	0 0 0 0 0 0 0 1	Czyść LCD, kursor na pozycję spoczynkową.
COMM_LCD	0 0 0 0 0 0 1 X	Cofnij kursor na pozycję spoczynkową, zeruj licznik pozycji, zawartość DD RAM bez zmian.
COMM_LCD	0 0 0 0 0 1 I/D S 0 0 0 1 1 0 1 1	Tryb przesuwania kursora i obrazu: Znak pod kursor, kursor w lewo; Obraz w prawo, kursor bez zmian; Znak pod kursor, kursor w prawo; Obraz w lewo, kursor bez zmian.
COMM_LCD	0 0 0 0 1 D C B 1 0 0 0 1 1 0 1 1 0 1	Tryb wyświetlania kursora i obrazu: Brak kursora; Miga znak i podkreślenie; Znak nie miga, widoczne podkreślenie; Znak miga, widoczne podkreślenie; Wyświetlacz wyłączony Wyświetlacz włączony.
COMM_LCD	0 0 0 1 S/C R/L X X 0 0 0 1 1 0 1 1	Rozkaz przesunięcia kursora lub obrazu: Przesunięcie kursora w lewo; Przesunięcie kursora w prawo; Przesunięcie obrazu w lewo; Przesunięcie obrazu w prawo.
COMM_LCD	0 0 1 DL N F X X 0 0 0 1 1 0 1 1 0 1	Tryb wyświetlania kursora i obrazu: Jedna linia znaków, matryca 5x7; Jedna linia znaków, matryca 5x10; Dwie linie znaków, matryca 5x7; Kombinacja niedozwolona; Słowo danych 4-ro bitowe; Słowo danych 8-mio bitowe.
COMM_LCD	0 1 A A A A A A	Wpisanie adresu CG RAM do licznika adresów
COMM_LCD	1 A_N A A A A A A 0 1	Wpisanie adresu DD RAM do licznika adresów: A_N=0 dotyczy pierwszej linii znaków (00H – 27H); A_N=1 dotyczy drugiej linii znaków (40H – 67H).
COMM_LCD	D D D D D D D D	Wpisanie/odczyt danych do CG RAM lub DD RAM
COMM_LCD	BF A A A A A A A	Odczytanie flagi zajętości BF i zawartości licznika adresów

WSKAZÓWKA:
Dla trybu wyświetlania w dwóch liniach matryca znaków może się składać tylko z 5x7 punktów.
Przed wysłaniem każdego kolejnego słowa (danej lub instrukcji) należy sprawdzać flagę gotowości BF.

3. Wyświetlacz włączony/wyłączony.



4. Wyczyszczenie wyświetlacza.

ADRES	D7	D6	D5	D4	D3	D2	D1	D0
COMM_LCD	0	0	0	0	0	0	0	1

5. Ustawienie trybu pracy wyświetlacza.

ADRES	D7	D6	D5	D4	D3	D2	D1	D0
COMM_LCD	0	0	0	0	0	1	I/D	S

0	- wpis znaku od lewej strony
1	- wpis znaku od prawej strony

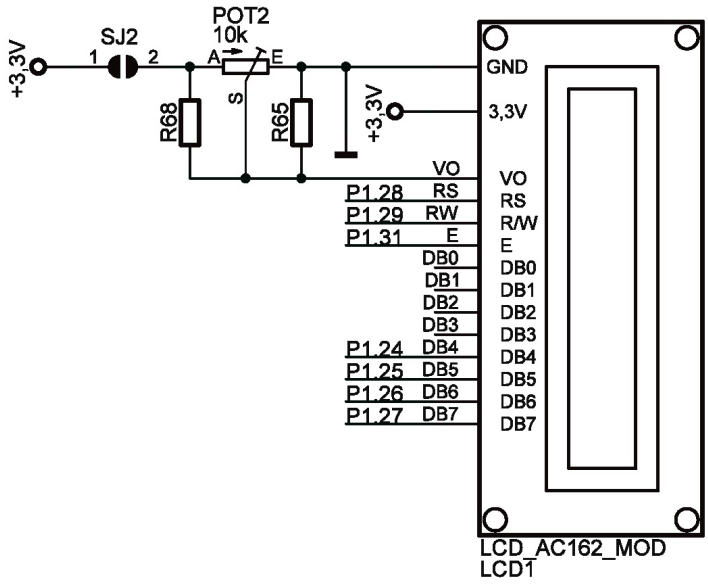
0	- dekrementacja
1	- inkrementacja

MPU ma bezpośredni dostęp do Rejestru Instrukcji (**IR**) oraz Rejestru Danych (**DR**). Wewnętrzne operacje w wyświetlaczu LCD określane są sygnałami generowanymi przez MPU:

- sygnał wyboru rejestrów RS
- sygnał czytaj/pisz R/W
- sygnały szyny danych DB7 - DB6.

Sygnały wysyłane do rejestru IR tworzą zestaw instrukcji który został podany w tabeli 5.

WSKAZÓWKA:
Zmiana trybu w trakcie pracy nie powoduje zmiany zawartości DDRAM i CGRAM.
Zmiana liczby wierszy wyświetlanych znaków musi być przeprowadzona poprzez realizację procedury inicjalizacji wyświetlacza.
W programie można zdefiniować funkcję, która będzie przesyłała do mikrokontrolera LCD 4 bity danych, a następnie dwie funkcje korzystające z poprzednio zasugerowanej, przesyłające rozkaz lub dane.
Ważne jest, aby zdefiniować funkcję sprawdzającą stan zajętości mikrokontrolera LCD (flaga BF). Przed przesłaniem danych lub rozkazu należy sprawdzić, czy mikrokontroler LCD nie jest zajęty przetwarzaniem poprzednio przesłanego rozkazy.



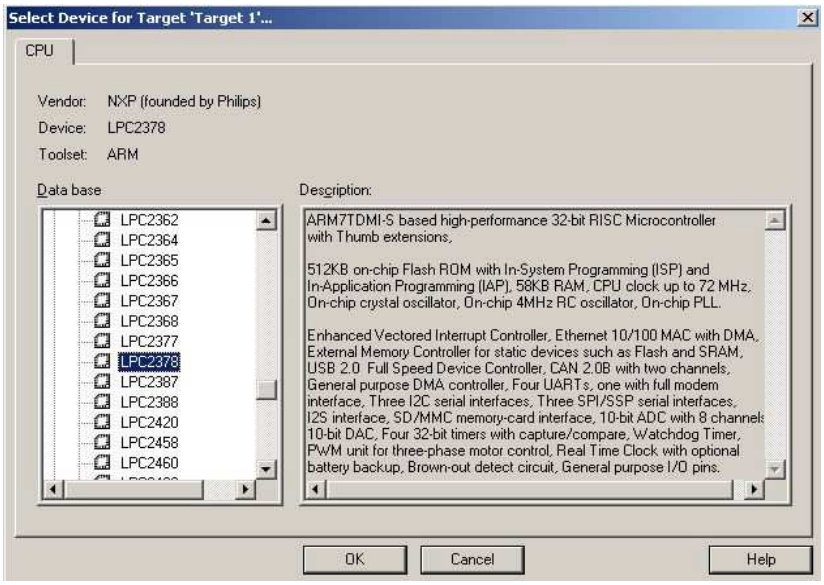
Rys. 5. Schemat podłączenia wyświetlacza LCD w zestawie uruchomieniowym MCB2300

Na rysunku 5 pokazano schemat podłączenia wyświetlacza LCD w zestawie uruchomieniowym MCB2300. Potencjometr POT2 służy do regulacji podświetlenia wyświetlacza. Linie sterujące RS, RW oraz E zostały

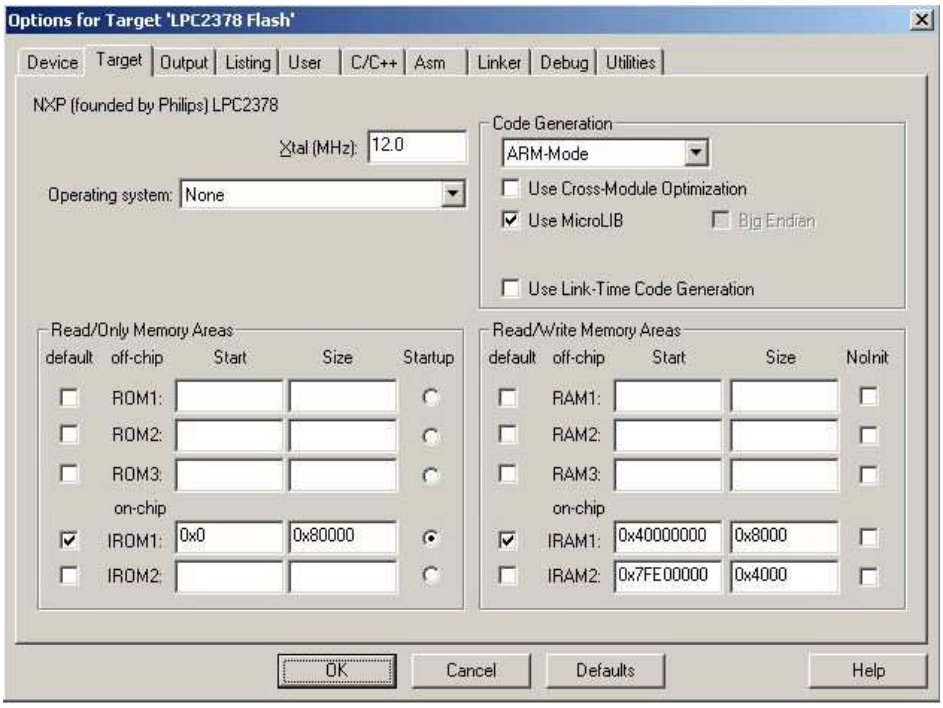
podłączone, odpowiednio, do pinów P1.28, P1.29 oraz P1.31 mikrokontrolera (PORT1). Linie danych DB4 – DB7 zostały odpowiednio podłączone do pinów P1.24 – P1.27 mikrokontrolera (PORT1).

4. KONFIGURACJA APLIKACJI KEIL µVISION4

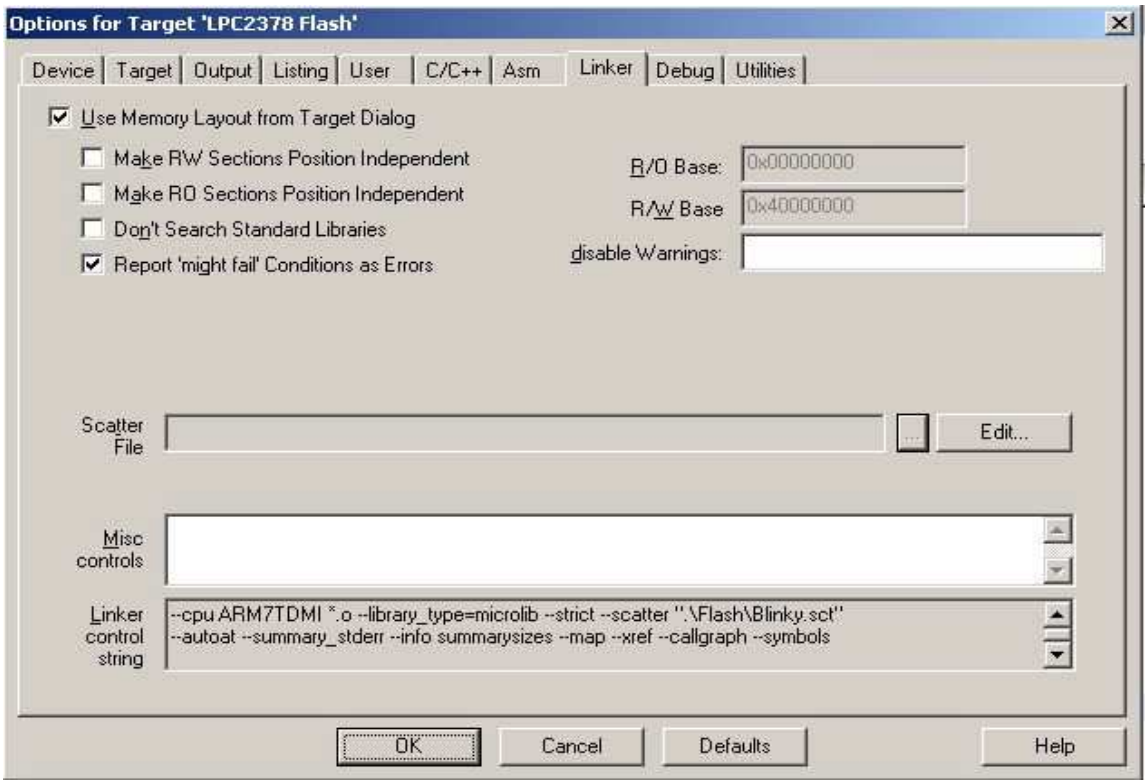
Aplikacja Keil uVision4 jest środowiskiem, w którym tworzymy i modyfikujemy projekt, oraz dokonujemy programowania mikrokontrolera. Pracę należy rozpocząć od uruchomienia aplikacji oraz zamknięcia otwartych projektów. Następnie z menu **Project** wybieramy **New µVision Project**. Program poprosi nas o wskazanie CPU użytego w płytce prototypowej. Wybieramy procesor z grupy **NXP**, model **LPC2378** (rysunek 6). Projekt zapisujemy na swoim dysku sieciowym.
Następnie należy skonfigurować program pod kątem współpracy z dostępnym programatorem. Wymaga to przejścia do menu **Flash → Options**, i podjęcia kilku kroków, pokazanych na rysunkach 7 – 10.



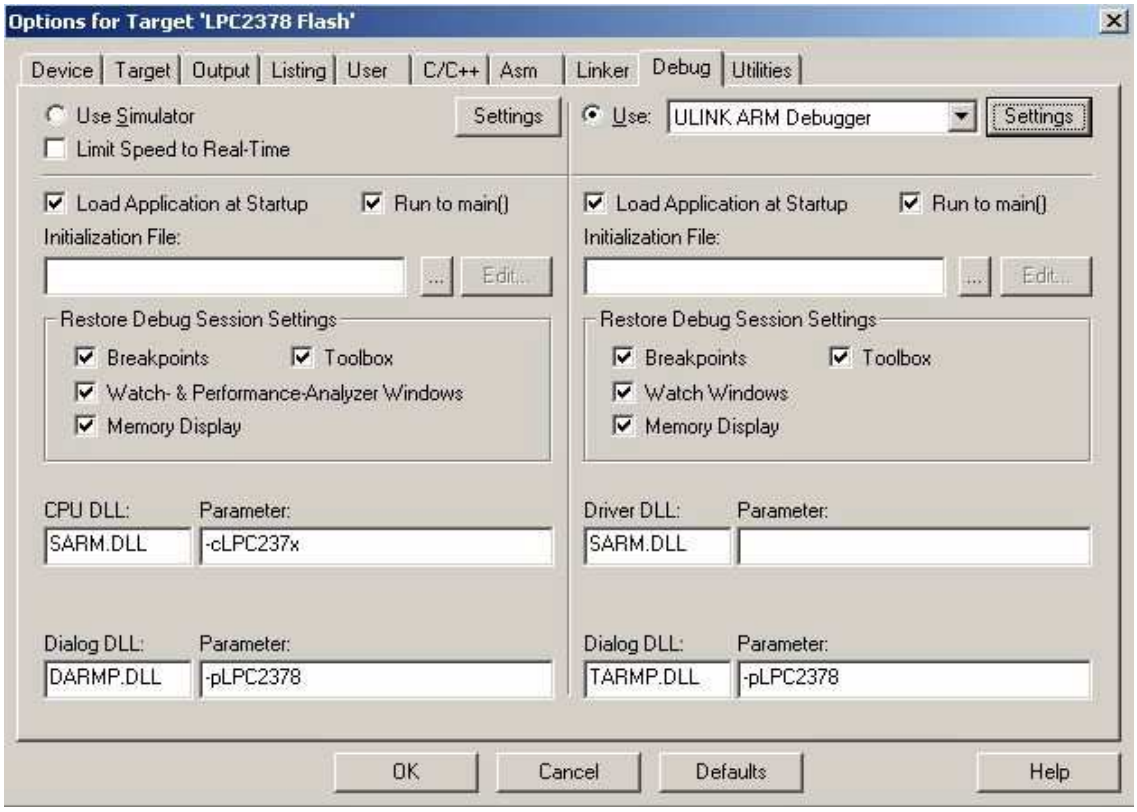
Rys. 6. Okno wyboru procesora



Rys. 7. Okno określania parametrów płytki prototypowej i sposobu generowania kodu

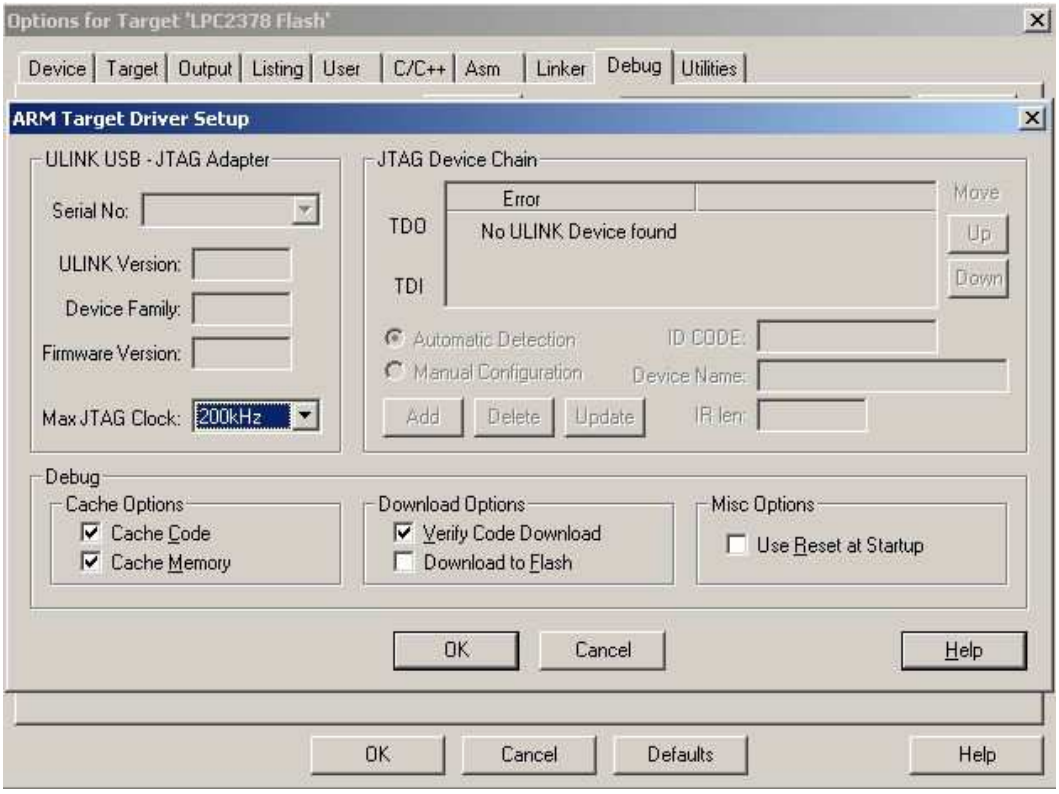


Rys. 8. Okno konfiguracji linkera



Rys. 9. Okno konfiguracji debugera

Następnie wybieramy przycisk **Settings** dla ULINK ARM Debugger, w celu modyfikacji ustawień zegara programatora.

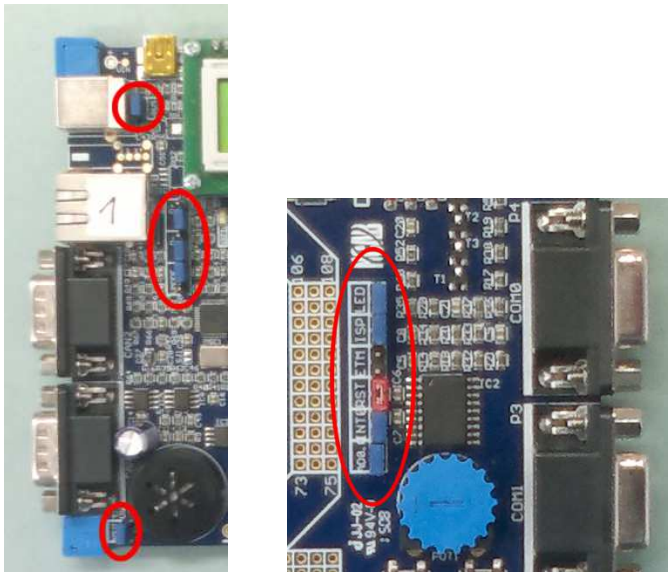


Rys. 10. Okno konfiguracji zegara dla adaptera ULINK USB-JTAG

W zakładce **Utilities** ustawiamy **Use Target for Flash Programming**.

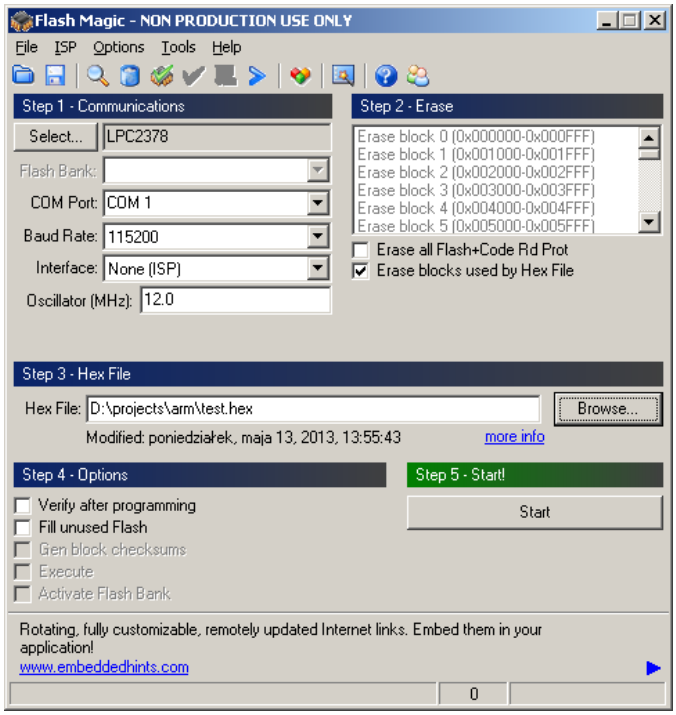
W przypadku, gdy programowanie płytki prototypowej odbywa się za pomoca portu szeregowego, należy dodatkowo wykonać następujące czynności:

6. Sprawdzić i ew. skorygować ustawienie zworek na płycie uruchomieniowej umożliwiającej zaprogramowanie układu za pomocą portu RS232 (rys. 1)



Rys. 11. Ustawienie zworek na płycie MCB2300

7. Podłączyć jeden koniec kabla RS232 do komputera
8. Podłączyć drugi koniec kabla do złącza COM0 zestawu uruchomieniowego
9. Podłączyć jeden koniec kabla do portu USB komputera

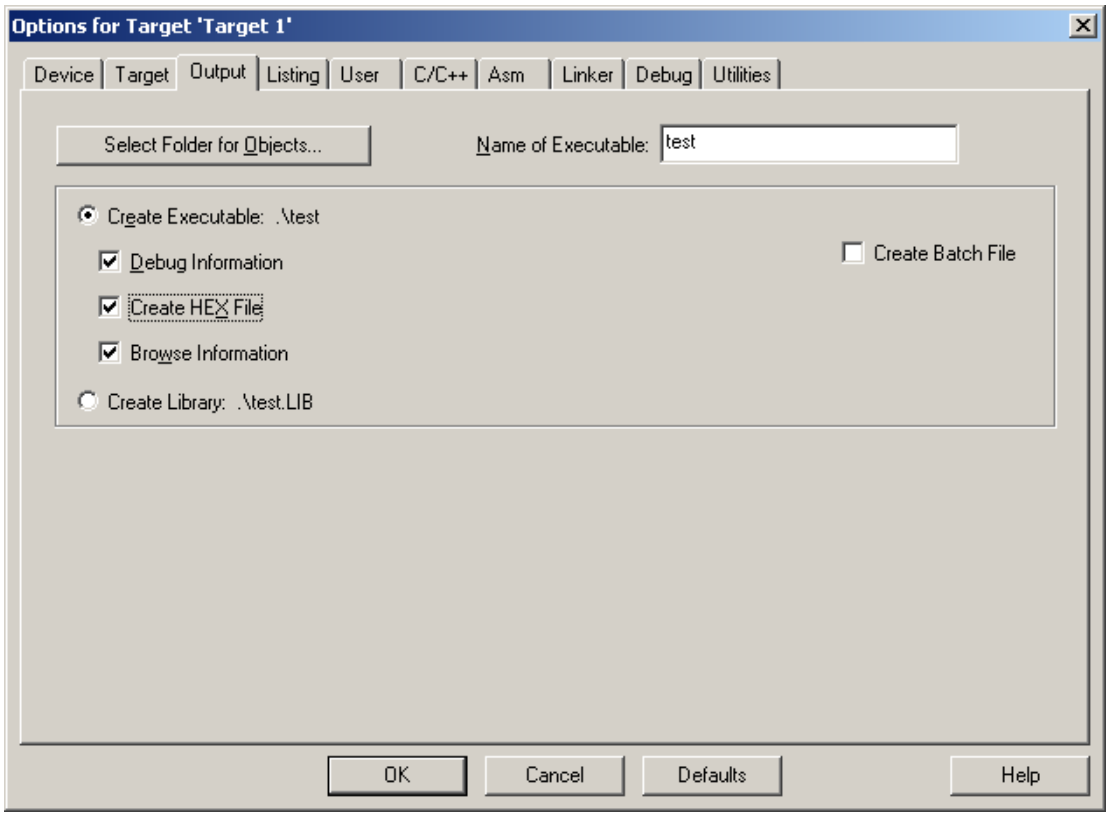


Rys. 12. Okno programu Flash Magic

10. Podłączyć drugi koniec kabla do złącza USB zestawu uruchomieniowego
11. Uruchomić program Flash Magic
12. Skonfigurować zgodnie z rys. 12, zwracając szczególną uwagę na wybór mikrokontrolera (LPC2378 oraz ustawienia portu szeregowego)
13. Wczytać w programie Flash Magic program wynikowy w formacie *.HEX
14. Zaprogramować układ za pomocą przycisku Start w oknie programu Flash Magic(rys. 12)

UWAGA
W środowisku Keil uVision przed kompilacją należy ustawić opcję „Create HEX file” (rys. 13) – opcja menu Flash | Configure Flash Tools, zakładka Output.

Po zakończeniu konfiguracji programu, możemy przystąpić do pisania projektu. Z menu kontekstowego, dostępnego po naciśnięciu **Target 1**, wybieramy **Add Group** i tworzymy grupę, w której umieścimy kod programu. Wybieramy ikonę **New**, a następnie zapisujemy kod jako *nazwa.c*. Ostatecznie dołączamy plik z kodem do projektu, wybierając z menu kontekstowego dostępnego do wybraniu poprzednio utworzonej grupy, opcję **Add Files to Group** ….
Kompilacji programu dokonujemy poprzez wciśnięcie **F7**. **Zaprogramowania mikrokontrolera** dokonujemy poprzez wybór przycisku **Load**, znajdującego się w dolnym rzędzie paska narzędziowego.



Rys. 13. Opcje generowania pliku wynikowego w programie Keil uVision

5. NIEZBĘDNE DEKLARACJE

Program wymaga dołączenia bibliotek:

```
#include <LPC23xx.H>         /* definicje dla LPC23xx*/
#include <stdio.h>
```

Ponieważ standardowo diody zestawu są zapalone, sugerowane jest ich zgaszenie w następujący sposób: wprowadzamy niezbędne definicje dla każdej z diod

```
#define _BV(x) (1<<(x))
#define LED_0 _BV(0)
#define LED_1 _BV(1)
#define LED_2 _BV(2)
#define LED_3 _BV(3)
#define LED_4 _BV(4)
#define LED_5 _BV(5)
#define LED_6 _BV(6)
#define LED_7 _BV(7)
#define LED_MASKA (LED_0|LED_1|LED_2|LED_3|LED_4|LED_5|LED_6|LED_7)
```

Następnie zdefiniować funkcję inicjującą diody:

```
void LED_init(void)
{
PINSEL10=0;
FIO2DIR|=LED_MASKA;
FIO2MASK=0x0000000;
```

```
}
```

PINSEL10 odpowiedzialny jest tylko i wyłącznie za włączenie i wyłączenie interfejsu ETM (Embedded Trace Module – śledzi pracę procesora ARM na poziomie rdzenia). Wyłączenie interfejsu ETM umożliwia obsługę diód LED.
FIO2DIR ustawia odpowiednie piny portu 2 jako piny wyjściowe.
FIO2MASK pisze, czyta ustawia i czyści port. (poprzez FIO2PIN, FIO2SET i FIO2CLR).
Ostatecznie, w części głównej programu wystarczy wywołać powyższą funkcję, co zgasi diody.

Proponuje się zdefiniować zmienne mówiące o tym, ile linii i ile znaków obsługuje wyświetlacz:

```
/*-----------Definicje rozmiarow dla tekstowego LCD---------------------*/

#define LineLen     16        /* Szerokosc (w znakach)                  */
#define NumLines     2        /* Wysokosc (w liniach)                   */
```

Należy również przypisać poszczególnym wyprowadzeniom wyświetlacza maski, które będą pozwalały operować na określonych portach mikrokontrolera, zgodnie z rysunkiem 5:

```
/*----------------Definicje dotyczace wersji boardu---------------------*/

#define PIN_E          0xC0000000
#define PIN_RW         0x20000000
#define PIN_RS         0x10000000
#define PINS_CTRL      0xF0000000
#define PINS_DATA      0x0F000000
```

W ostatnim kroku należy wprowadzić definicje, które pozwolą ustawić charakterystyczne wartości na poszczególnych wyprowadzeniach wyświetlacza LCD. Poszczególne linie wyświetlacza LCD podłączone są do linii portu 1 mikrokontrolera LPC2378. Z tego powodu funkcje sterujące ustawieniami pinów (zgodnie z tabelą 3) mają w swych nazwach cyfrę 1. Chcąc ustawić na tym porcie bit 31, aby ustawić pin P1.31 (sygnał E LCD), należy przypisać pinowi E odpowiednią wartość szesnastkową. Aby ułatwić sobie pracę, wybrano wartość C0000000, która po zamianie na wartość binarną, daje nam 1 na 30 i 31 bicie. Bit 30 tego portu nas nie interesuje, ale w ten sposób mamy ustawiony na 1 bit 31 (sygnał E). Sygnał R/W podłączony jest do pinu P1.29, czyli przypisując temu pinowi wartość 20000000 szesnastkowo, po przeliczeniu na wartość binarną, da nam to 1 na pozycji 29, i resztę 0, czyli wybierzemy pin P1.29, i tak dalej. **Przeanalizuj dokumentację** kontrolera i zastanów się, dlaczego poniższe wpisy mają taką postać:

```
/* pin E  ustawienie 0 lub 1 impuls zapisu/odczytu                       */
#define LCD_E(x)   ((x) ? (IOSET1 = PIN_E)  : (IOCLR1 = PIN_E) );

/* pin RW ustawianie na 0 - pisz lub 1 - czytaj                          */
#define LCD_RW(x)  ((x) ? (IOSET1 = PIN_RW) : (IOCLR1 = PIN_RW));

/* pin RS ustawianie na 0 - kod instrukcji lub 1 - dane                  */
#define LCD_RS(x)  ((x) ? (IOSET1 = PIN_RS) : (IOCLR1 = PIN_RS));

/* Odczyt pinów DATA                                                     */
#define LCD_DATA_IN  ((IOPIN1 >> 24) & 0xF)

/* Zapis wartosci do pinów DATA                                          */
#define LCD_DATA_OUT(x)  IOCLR1 = PINS_DATA; IOSET1 = (x & 0xF) << 24;

/* Ustawienie pinów w tryb wyjscia                                       */
#define LCD_ALL_DIR_OUT  IODIR1  |=  PINS_CTRL | PINS_DATA;

/* Ustawienie pinów DATA w tryb wejscia                                  */
#define LCD_DATA_DIR_IN  IODIR1 &= ~PINS_DATA;
```

```
/* Ustawienie pinów DATA w tryb wyjscia                                    */
#define LCD_DATA_DIR_OUT  IODIR1 |=  PINS_DATA;
```

Realizując zadania, warto zdefiniować funkcje pomocnicze: przesyłającą 4 bity danych (uwaga na ustawienie pinów w LPC2378!), przesyłającą rozkaz oraz przesyłającą znak do wyświetlenia. Pisząc funkcje należy pamiętać o odpowiednim ustawieniu sygnałów sterujących pracą wyświetlacza LCD (RS, R/W, E). Należy też zwrócić uwagę na fakt, że mikrokontroler w wyświetlaczu w trakcie realizacji rozkazu pomija wszystkie nadchodzące nowe rozkazu do momentu, gdy zakończy przetwarzanie aktualnego rozkazu.

6. ZADANIA

1. Napisz program, w którym dokonasz inicjalizacji wyświetlacza LCD, a następnie wypiszesz w obu liniach wyświetlacza tekst.
2. Zmodyfikuj program z zadania 1 tak, aby napis pojawiał się z prawej strony na wyświetlaczu (litera po literze), przewijał się przez wyświetlacz, i znikał po lewej stronie wyświetlacza (znak po znaku).
3. Zmodyfikuj program z zadania 2 w taki sposób, aby uzyskać efekt napisu odbijającego się od lewej i prawej strony ekranu (efekt ping-pong).

7. LITERATURA

[1] NXP: Dokumentacja techniczna mikrokontrolerów z rodziny LPC23xx, http://ics.nxp.com/support/ documents/microcontrollers/pdf/user.manual.lpc23xx.pdf (dostęp: kwiecień 2011).
[2] NXP, Serwis poświęcony mikrokontrolerom z rodziny LPC23xx, zawierający przykłady wykorzystania i programowania, http://ics.nxp.com/support/documents/microcontrollers/?scope=LPC2300 (dostęp: kwiecień 2011).
[3] Keil, Dokumentacja zestawu uruchomieniowego MCB2300 wraz z przykładami, http://www.keil. com/ support/man/docs/mcb2300 (dostęp: kwiecień 2011).
[4] Ampire Co. LTD., Dokumentacja dotycząca wyświetlaczy LCD z rodziny AC162B, http://www.ampire. com.tw /Spec-AC/AC-162B.pdf (dostęp: kwiecień 2011).