Wyświetlacz alfanumeryczny LCD

Programy z użyciem LCD

Program – który pozwoli zwizualizować wartości zmiennych zapisanych w rejestrach

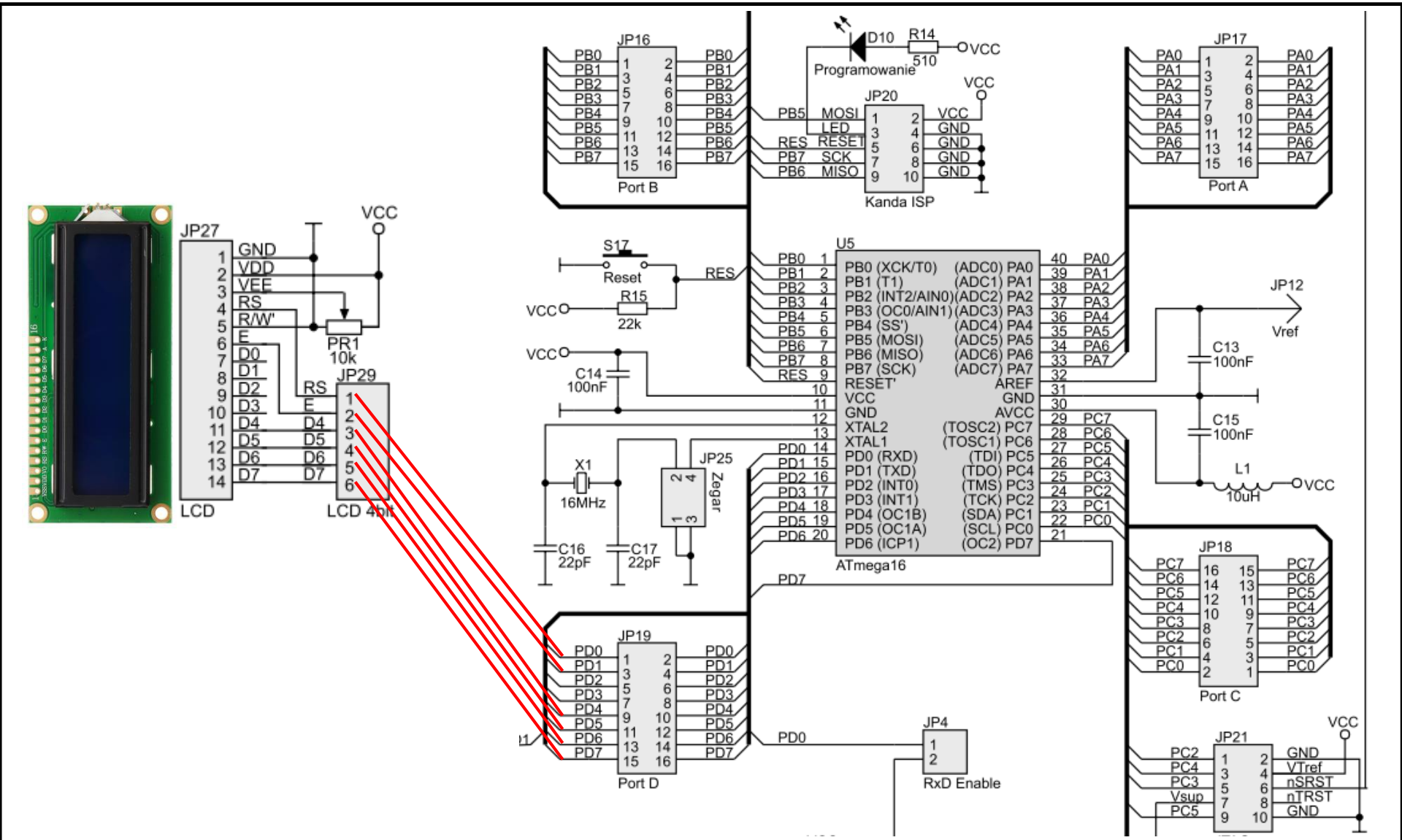
Celem ćwiczenia jest zapoznanie się z zagadnieniami związanymi z obsługą interfejsów równoległych.

Jako przykład przedstawiony zostanie sposób sterowania standardowym, alfanumerycznym wyświetlaczem LCD z wbudowanym sterownikiem HD44780 firmy Hitachi.

Przedstawione zostaną procedury, które ułatwią wykorzystywanie tego modułu do graficznego przedstawienia zmiennych i łańcuchów tekstowych.

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

Wyświetlacz LCD – schemat połączeń



Wyświetlacz LCD – schemat połączeń i konfiguracji ZL3AVR

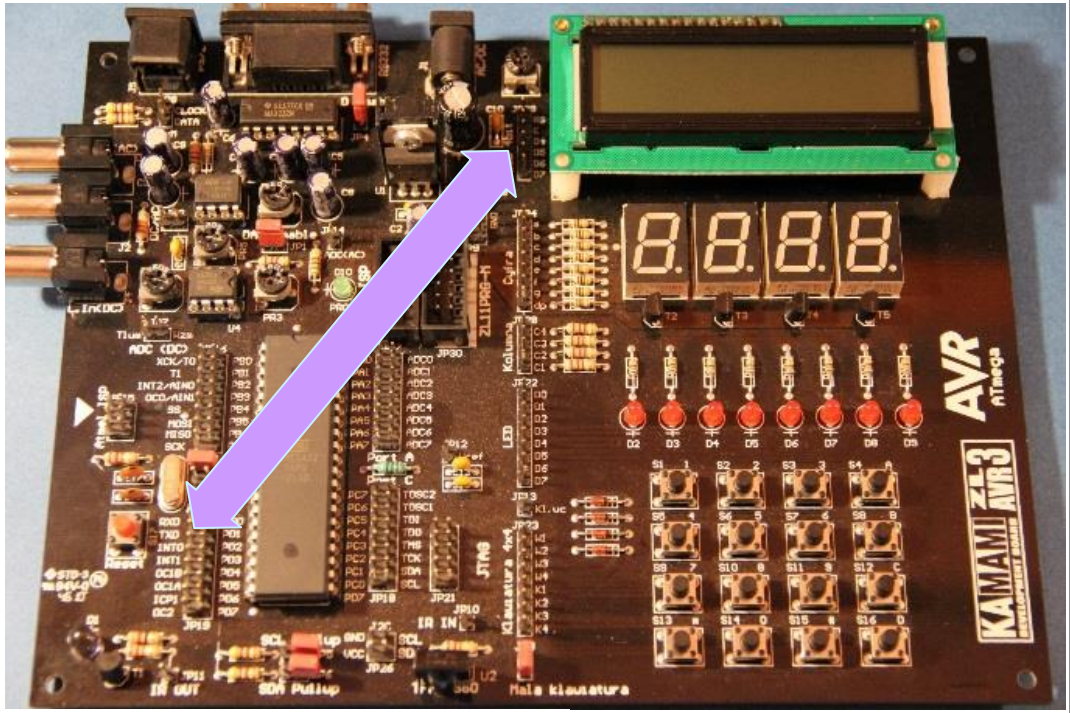
Obwód ćwiczeniowy:

W zestawie ZL3AVR należy:

a) połączyć wyprowadzenia wyświetlacza LCD z portem PD mikrokontrolera (JP29:RS,E,D4...D7 – JP19:PD0,PD1,PD4...PD7),

b) wyregulować kontrast wyświetlacza LCD (potencjometrem PR1),

e) zaprogramować bity konfiguracyjne mikrokontrolera następująco (wykorzystujemy wewnętrzny oscylator nastawiony na częstotliwość 1 MHz):



OCDEN	JTAGEN	SPIEN	CKOPT	EESAVE	BOOTSZ1	BOOTSZ0	BOOTRST
*	*	*			*	*	
BODLEVEL	BODEN	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0
*	*	*	*	*	*	*	

* – bit zaprogramowany (o zerowej wartości logicznej)

Wyświetlacz LCD

Interfejs modułów LCD

Przykładem urządzeń wewnątrzsystemowych o interfejsie równoległym jest większość wyświetlaczy ciekłokrystalicznych ze zintegrowanym sterownikiem. Moduły te są obecnie bardzo popularne zarówno w konstrukcjach amatorskich, jak i profesjonalnych ze względu na niewygórowaną cenę i prostotę obsługi. Wbudowane w wyświetlacze tego rodzaju sterowniki działają zwykle w zgodzie ze specyfikacją układu HD44780 firmy Hitachi (jest to niemalże standard wśród modułów wyświetlających). Dzięki temu większość „inteligentnych" wyświetlaczy alfanumerycznych obsługuje się podobnie.

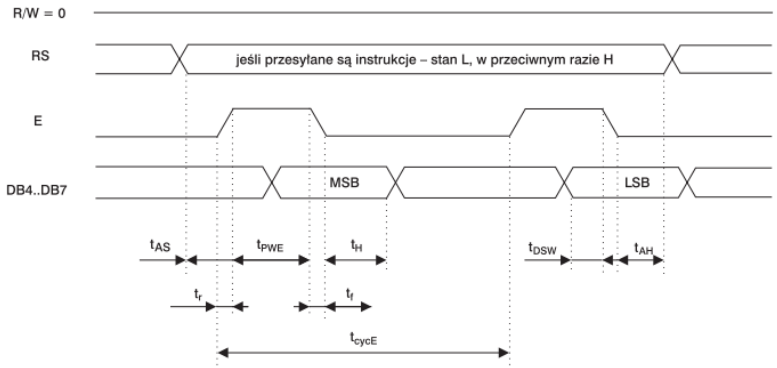
Wyświetlacze LCD charakteryzują się interfejsem 8-bitowym, tworzonym przez: 8 linii danych (DB0...DB7), linię potwierdzającą (E) i dwie linie specjalne (RS i R/W). Sterownik HD44780 jest jednak na tyle elastyczny, że umożliwia przełączenie interfejsu w tryb 4-bitowy (wykorzystywane są wówczas tylko starsze linie danych – DB4...DB7). Co więcej, zrezygnować można także z linii określającej kierunek transmisji – R/W, stale wymuszając na niej stan niski (uproszczenie to jest równoznaczne z rezygnacją z możliwości odczytu stanu wyświetlacza i zawartości jego pamięci). W praktyce stosuje się najczęściej oba uproszczenia jednocześnie – korzysta się z jednokierunkowego interfejsu 4-bitowego, tworzonego przez linie: DB4...DB7, E i RS. W dalszej części ćwiczenia będziemy zajmować się tylko tym przypadkiem, tylko te linie bowiem wyprowadzone zostały w zestawie ZL3AVR (R/W zwarto z GND).

Pin	Symbol	Opis
1	GND (VSS)	Potencjał odniesienia
2	VDD	Napięcie zasilające (zwykle 5 V)
3	VO (VEE)	Napięcie zasilające LCD, regulacja kontrastu (zwykle –3...+2 V)
4	RS	Sygnał określający znaczenie przesyłanej wartości (dane/$\overline{\text{instrukcja}}$)
5	R/$\overline{\text{W}}$	Kierunek transmisji (odczyt/$\overline{\text{zapis}}$)
6	E	Sygnał potwierdzający (potwierdzanie zboczem opadającym)
7...14	DB0...DB7	8 linii danych
15	A	elektroda „+" podświetlania (opcja)
16	K	elektroda „–" podświetlania (opcja)

Tab.1 Choć standard teoretycznie nie istnieje, większość wyświetlaczy LCD wyposażona jest w złącze, którego opis przedstawiono w tabeli

Sekwencja transmisji

Bez względu na wybraną szerokość interfejsu (4 lub 8 bitów) przesyłane do modułu wyświetlającego wartości mają zawsze postać 8-bitową. W trybie 4-bitowym transmisja jednej instrukcji lub danej (jednego bajtu) musi dokonywać się zatem w dwóch cyklach – w pierwszym przesyłany jest półbajt starszy, a w drugim – młodszy. Ważność danych wystawionych na liniach DB4...DB7 potwierdza się opadającym zboczem na linii E. W trakcie cyklu transmisyjnego stan linii RS określa znaczenie aktualnie przesyłanej wartości. Jeśli linia ta znajduje się w stanie niskim, przesłany bajt będzie interpretowany przez sterownik jako instrukcja. W przeciwnym razie wartość ta zostanie wzięta za daną do zapisania w pamięci wyświetlacza – DDRAM. Obsługę interfejsu należy przeprowadzać z zachowaniem wymaganych zależności czasowych. Na rysunku 1 przedstawiono sekwencję transmisji pojedynczego bajtu, natomiast w tabeli 2 zebrano typowe wartości czasów krytycznych. Na tym etapie należałoby dodatkowo zaznaczyć, że pomiędzy pełnymi sekwencjami (po transmisji całego bajtu) jest wymagana przerwa, trwająca minimum 37 µs lub dłużej.



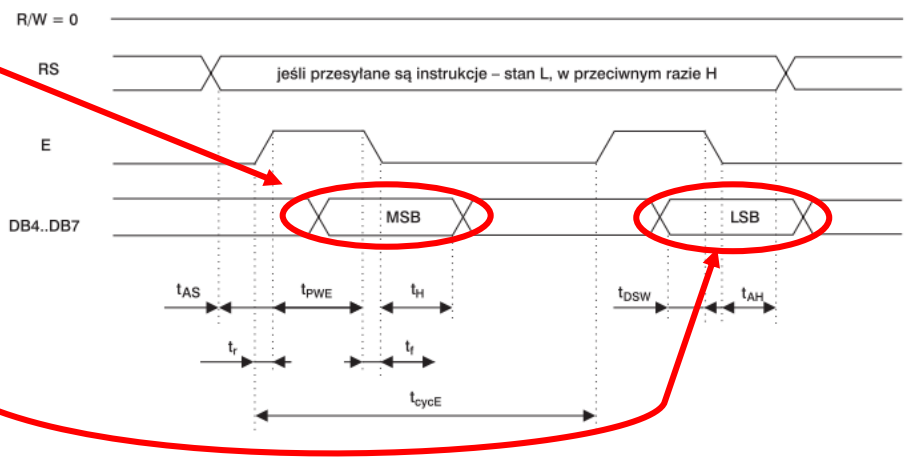
Rys.1 Sekwencja przesyłu pojedynczego bajtu do modułu LCD

Symbol	Opis	Min.	Maks.	Jednostka
t_{AS}	Ustalanie adresu	40	–	ns
t_{PWE}	Impuls E (stan wysoki)	230	–	
t_H	Przetrzymanie danych	10	–	
t_{DSW}	Ustalanie danych	80	–	
t_{AH}	Przetrzymanie adresu	10	–	
t_{cycE}	Cykl transmisji półbajtu	500	–	
t_r i t_f	Czasy trwania zboczy w sygnale E	–	20	

Tab. 2 Zależności czasowe przy transmisji do modułu LCD

Wyświetlacz LCD

Sekwencja transmisji



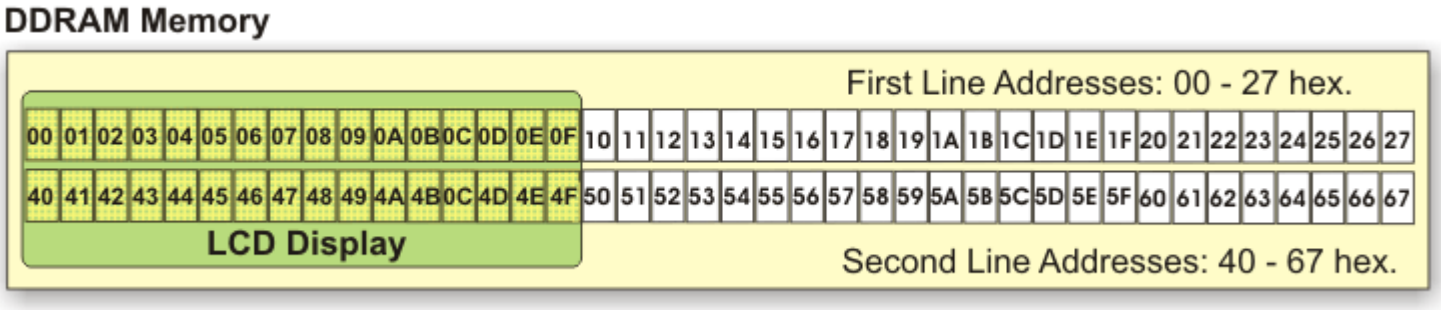
a = 01100001

Lower 4bit \ Higher 4bit	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
××××0000													
××××0001													
××××0010													
××××0011													
××××0100													
××××0101													
××××0110													
××××0111													
××××1000													
××××1001													
××××1010													
××××1011													
××××1100													
××××1101													
××××1110													
××××1111													

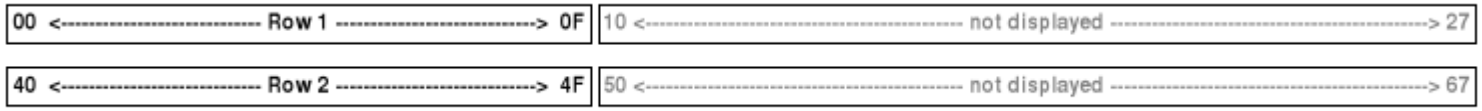
Tab. 3 Wygląd znaków zapisanych w generatorze znaków sterownika LCD HD44870 (wersja standardowa)

Sterownik HD44780

W rzeczywistości sterownik wyposażono w dwa rodzaje pamięci – DDRAM (ang. Display Data RAM) i CGRAM (ang. Character Generator RAM). W DDRAM przechowywane są aktualnie wyświetlane znaki w postaci zakodowanej. Sposób tego kodowania pokrywa się w pewnym zakresie z kodowaniem ASCII (w przybliżeniu przyjąć można, że kody ASCII w zakresie 7-bitowym są interpretowane poprawnie). Układy w pełni zgodne z HD44780 wyposażone są w pamięć DDRAM o objętości 80 bajtów, a każdej pozycji wyświetlacza odpowiada osobna komórka tej pamięci. W zależności od modułu (od liczby znaków w linii, liczby linii i wytwórcy) sposób przyporządkowania komórek pamięci poszczególnym pozycjom znakowym może być różny. Najczęściej spotykaną konfigurację dla wyświetlaczy 2x16 (2 linie po 16 znaków – wyświetlacz taki zastosowano w zestawie ZL3AVR) przedstawiono na rysunku 2. Widzimy, że część komórek DDRAM znajduje się poza zasięgiem wyświetlania, jednak ich wizualizacja jest możliwa z zastosowaniem instrukcji „przesuwu okna”.



Rys.2 Typowy sposób przyporządkowania komórek pamięci pozycjom znakowym wyświetlacza 2x16



Wyświetlacz LCD

Transmisja instrukcji

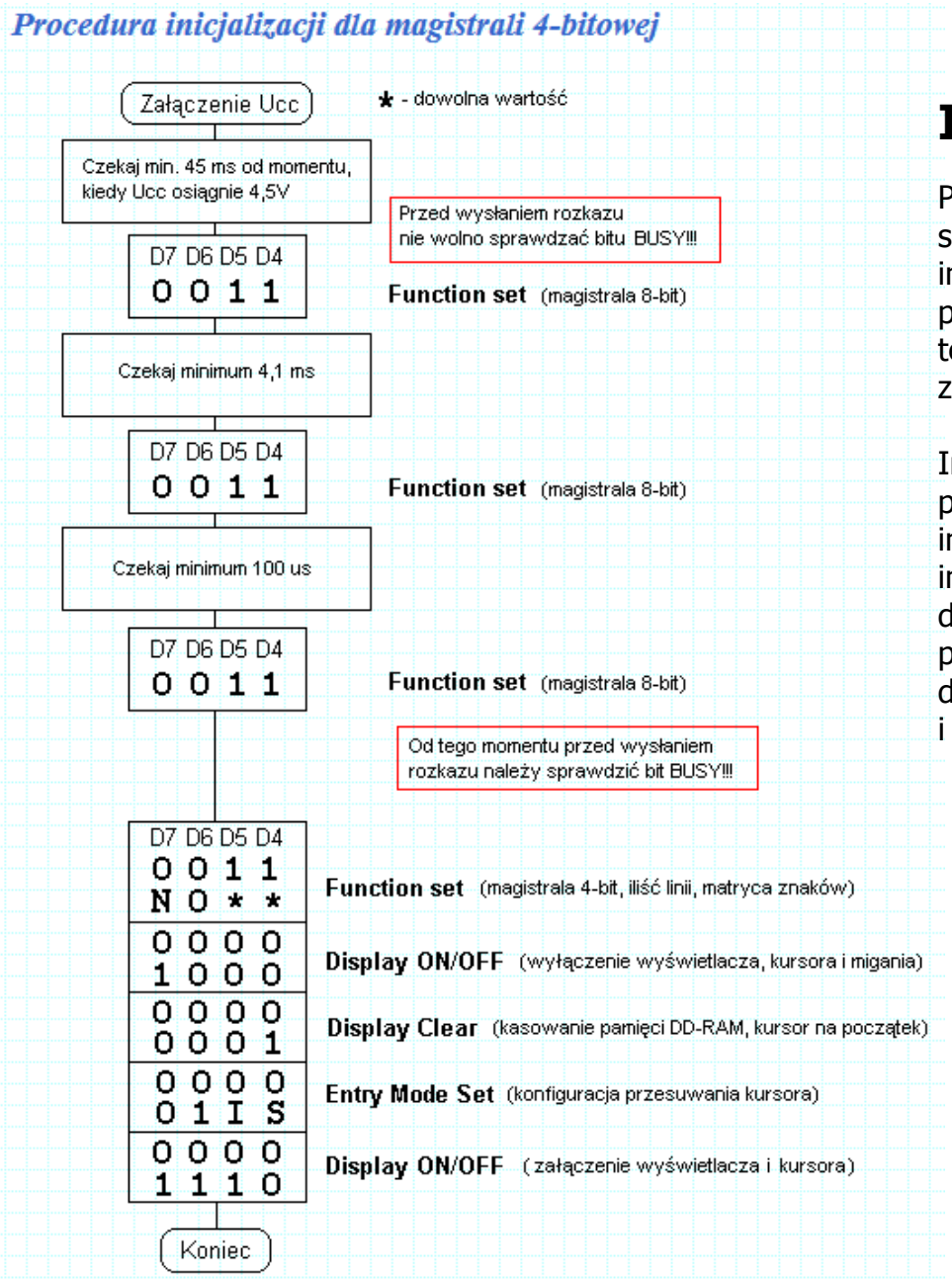
Już przy omawianiu interfejsu zaznaczono, że przesyłane wartości traktowane są jako instrukcje tylko wtedy, gdy w trakcie transmisji linia RS znajduje się w stanie niskim. Kody instrukcji obsługiwanych przez sterownik HD44780 przedstawiono w tabeli 4. Ponieważ stosujemy transmisję jednokierunkową (linia R/W w stanie niskim), pominięto tutaj instrukcje testujące stan i odczytujące zawartość pamięci wyświetlacza.

Instrukcja	Kod								Opis	Typowy, maks. czas wykonania
	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Czyść wyświetlacz								1	Wypełnia pamięć DDRAM znakami spacji i przesuwa kursor oraz okno do pozycji początkowej (adres DDRAM równy 0)	1,64 ms
Zawróć kursor							1	*	Przesuwa kursor i okno do pozycji początkowej (adres DDRAM równy 0)	1,64 ms
Ustaw tryb wprowadzania						1	I/D	S	Określa działanie podejmowane po zapisie komórki DDRAM lub CGRAM **I/D=1**: inkrementacja adresu (przy zapisie DDRAM kursor przesuwa się w prawo) **I/D=0**: dekrementacja adresu (przy zapisie DDRAM kursor przesuwa się w lewo) **S=1**: zamiast kursora przesuwa się wyświetlane okno (dotyczy wyłącznie operacji na pamięci DDRAM)	40 µs
Steruj wyświetlaczem					1	D	C	B	Włącza lub wyłącza niektóre funkcje wyświetlania **D=1**: wyświetlanie włączone **C=1**: kursor włączony **B=1**: miganie kursora włączone	40 µs
Przesuń okno lub kursor				1	S/C	R/L	*	*	Przesuwa kursor lub wyświetlane okno o jedną pozycję w wybranym kierunku **S/C=0**: przesuwanie kursora **S/C=1**: przesuwanie okna **R/L=0**: przesuwanie w lewo **R/L=1**: przesuwanie w prawo	40 µs
Zmień ustawienia wyświetlacza			1 DL N F * *						Określa interfejs i konfiguruje sterownik względem matrycy LCD **DL=0**: interfejs 4-bitowy **DL=1**: interfejs 8-bitowy **N=0**: wyświetlacz 1-liniowy **N=1**: wyświetlacz 2-liniowy **F=0**: matryca znakowa 5x8 punktów **F=1**: matryca znakowa 5x10 punktów	40 µs
Zmień adres CGRAM		1 Nowy adres CGRAM							Ustala aktualny adres pamięci CGRAM. Dane przesyłane po transmisji tej instrukcji (przy RS=1) będą zapisywane do pamięci CGRAM.	40 µs
Zmień adres DDRAM	1	Nowy adres DDRAM							Ustala aktualny adres pamięci DDRAM. Dane przesyłane po transmisji tej instrukcji (przy RS=1) będą zapisywane do pamięci DDRAM.	40 µs

* – wartość bez znaczenia

Tab. 4 Instrukcje sterownika HD44780

Wyświetlacz LCD



Inicjalizacja modułu

Po włączeniu zasilania sterownik HD44780 inicjalizuje się samoczynnie – włączany jest 8-bitowy tryb działania interfejsu, obsługa panelu LCD o jednej linii i matrycy 5×8 punktów, a ekran i kursor pozostają wygaszone. To jednak teoria – w praktyce sterownik powinno się zawsze zainicjalizować na drodze programowej.

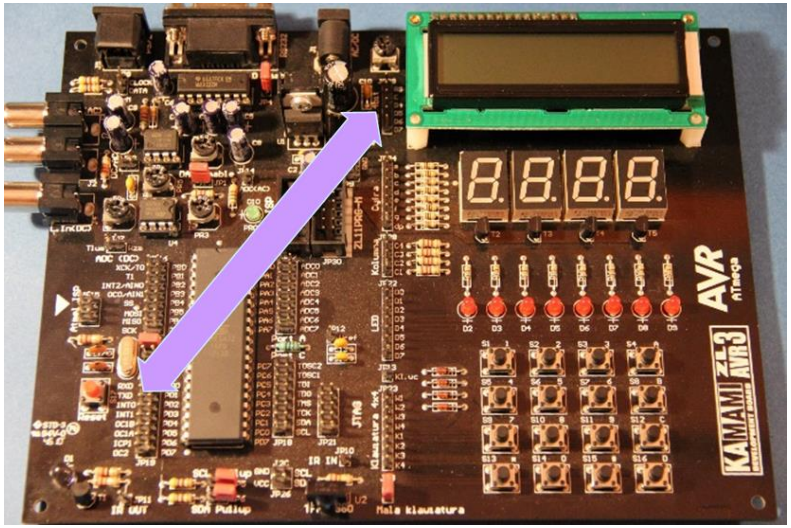
Inicjalizacja programowa polega na kilkukrotnym przesłaniu instrukcji konfigurującej wyświetlacz do pracy z interfejsem 8-bitowym. Wartość młodszej części kodu tej instrukcji jest przy tym nieważna – dzięki temu inicjalizacji dokonać możemy także poprzez interfejs 4-bitowy. Dopiero po kilkukrotnym przesłaniu wartości 0x3* przystąpić można do właściwej konfiguracji modułu. Sekwencja inicjalizująca i minimalne czasy opóźnień przedstawiono na rys. 3

Rys. 3 Sekwencja inicjalizująca sterownik HD44780

Inicjalizacja modułu

Posiadamy obecnie wystarczająco dużo informacji, aby móc zaimplementować obsługę najniższych warstw omawianego interfejsu w programie mikrokontrolera. Zanim jednak napiszemy odpowiednie procedury, weźmy pod uwagę konfigurację przyjętą we wstępie do ćwiczenia i zadeklarujmy kilka stałych:

```
.EQU K_LCD      = DDRD    ; rejestr kierunku portu wyświetlacza
.EQU O_LCD      = PORTD   ; rejestr wyjściowy portu wyświetlacza
.EQU LCD_RS     = 0       ; nr linii dla sygnału RS, podłączenie do portu PD0 mikrokontrolera
.EQU LCD_EN     = 1       ; nr linii dla sygnału EN, podłączenie do portu PD1 mikrokontrolera
.EQU LCD_DB4    = 4       ; nr linii dla sygnału DB4, podłączenie do portu PD4 mikrokontrolera
.EQU LCD_DB5    = 5       ; nr linii dla sygnału DB5, podłączenie do portu PD5 mikrokontrolera
.EQU LCD_DB6    = 6       ; nr linii dla sygnału DB6, podłączenie do portu PD6 mikrokontrolera
.EQU LCD_DB7    = 7       ; nr linii dla sygnału DB7, podłączenie do portu PD7 mikrokontrolera
```



Wyświetlacz LCD

Inicjalizacja modułu

Na podstawie podanej tabeli 4 możemy zdefiniować łatwe do zapamiętania etykiety, które zastąpią wagi bitów, wchodzących w skład instrukcji:

```
.EQU    LCD_CZYSC  = 0      ;  czyszczenie wyświetlacza i powrót kursora
.EQU    LCD_WROC   = 1      ;  powrót kursora do pozycji początkowej

.EQU    LCD_TRYB   = 2      ;  zmiana trybu zapisu pamięci
.EQU    LCD_T_INC  = 1      ;  adres inkrementowany przy każdym zapisie
.EQU    LCD_T_DEC  = 2      ;  adres dekrementowany przy każdym zapisie
.EQU    LCD_T_PRZES= 0      ;  okno przesuwane przy każdym zapisie

.EQU    LCD_STER   = 3      ;  sterowanie wyświetlaczem
.EQU    LCD_SW_ON  = 2      ;  wyświetlanie włączone
.EQU    LCD_SW_OFF= 3       ;  wyświetlanie wyłączone
.EQU    LCD_SK_ON  = 1      ;  kursor włączony
.EQU    LCD_SK_OFF = 3      ;  kursor wyłączony
.EQU    LCD_SM_ON  = 0      ;  miganie kursora włączone
.EQU    LCD_SM_OFF= 3       ;  miganie kursora wyłączone

.EQU    LCD_PRZES  = 4      ;  przesuwanie
.EQU    LCD_P_KUR  = 4      ;  przesuwanie kursora
.EQU    LCD_P_WYS  = 3      ;  przesuwanie okna
.EQU    LCD_LEWO   = 4      ;  przesuwanie w lewo
.EQU    LCD_PRAWO  = 2      ;  przesuwanie w prawo

.EQU    LCD_USTAW  = 5      ;  ustawienia interfejsu i wyświetlacza
.EQU    LCD_U_8BIT = 4      ;  interfejs 8-bitowy
.EQU    LCD_U_4BIT = 5      ;  interfejs 4-bitowy
.EQU    LCD_U_2LIN = 3      ;  wyświetlacz 2-liniowy
.EQU    LCD_U_1LIN = 5      ;  wyświetlacz 1-liniowy
.EQU    LCD_U_5x10 = 2      ;  znaki o rozmiarach 5x10 punktów
.EQU    LCD_U_5x8  = 5      ;  znaki o rozmiarach 5x8 punktów

.EQU    LCD_CGADR  = 6      ;  zmiana adresu CGRAM
.EQU    LCD_DDADR  = 7      ;  zmiana adresu DDRAM
```

Wyświetlacz LCD

Procedura Ini_LCD: Inicjalizacja modułu

Na podstawie podanej tabeli 4 oraz sekwencji z rys 3 należy napisać procedurę inicjalizacji wyświetlacza LCD:
Procedura ta będzie korzystać z dodatkowych procedur:
Czekaj_us
Czekaj_ms
Instrukcja_LCD
Czysc_LCD

```
;*************************************************************************************
Ini_LCD:
        push    R16
        push    R17

        sbi     K_LCD, LCD_RS       ; porty wyświetlacza pracują jako wyjścia
        sbi     K_LCD, LCD_EN
        sbi     K_LCD, LCD_DB4
        sbi     K_LCD, LCD_DB5
        sbi     K_LCD, LCD_DB6
        sbi     K_LCD, LCD_DB7

        cbi     O_LCD, LCD_RS       ; przesyłane wartości będą instrukcjami
        cbi     O_LCD, LCD_EN

        ldi     R16, 45
        ldi     R17, 1
        rcall   Czekaj_ms           ; oczekiwanie przez 45 ms na wypadek, gdyby
                                    ;  zasilanie dopiero zostało włączone
        sbi     O_LCD, LCD_EN
        sbi     O_LCD, LCD_DB4
        sbi     O_LCD, LCD_DB5
        cbi     O_LCD, LCD_DB6
        cbi     O_LCD, LCD_DB7
        nop
        nop
        cbi     O_LCD, LCD_EN       ; ustawienie DB7..4=0011 i potwierdzenie
                                    ;  zboczem opadającym na linii EN
        ldi     R16, 5
        ldi     R17, 1
        rcall   Czekaj_ms           ; oczekiwanie przez 5 ms
```

Proceduta Inicjalizacja modułu Ini_LCD:

Wyświetlacz LCD

```
;*****************************************************************************************
                ldi             R17, 2
Ini_LCD_1:
                sbi             O_LCD, LCD_EN
                ldi             R16, 1
                rcall           Czekaj_us
                cbi             O_LCD, LCD_EN
                ldi             R16, 20
                rcall           Czekaj_us
                dec             R17
                brne            Ini_LCD_1               ;  dwukrotne powtórzenie transmisji tej samej
                                                        ;  wartości (DB7..4=0011) z opóźnieniem 200 us
                                                        ;  (programowa inicjalizacja wyświetlacza)

                sbi             O_LCD, LCD_EN
                cbi             O_LCD, LCD_DB4
                ldi             R16, 1
                rcall           Czekaj_us
                cbi             O_LCD, LCD_EN
                ldi             R16, 10
                rcall           Czekaj_us               ;  transmisja wartości DB7..4=0010
                                                        ;  (przejście w tryb interfejsu 4-bitowego)

                ldi             R16, (1<<LCD_USTAW)|(1<<LCD_U_4BIT)|(1<<LCD_U_2LIN)|(1<<LCD_U_5x8)
                rcall           Instrukcja_LCD          ; interfejs 4-bitowy, 2 linie, 5x8 pkt

                ldi             R16, (1<<LCD_TRYB)|(1<<LCD_T_INC)
                rcall           Instrukcja_LCD          ; tryb zwiększania adresu przy zapisie

                ldi             R16, (1<<LCD_STER)|(1<<LCD_SW_ON)|(1<<LCD_SK_ON)|(1<<LCD_SM_ON)
                rcall           Instrukcja_LCD          ; wyświetlanie włączone, kursor aktywny i miga

                sbi             O_LCD, LCD_RS           ; przesyłane wartości będą danymi

                rcall           Czysc_LCD

                pop             R17
                pop             R16
                ret
```

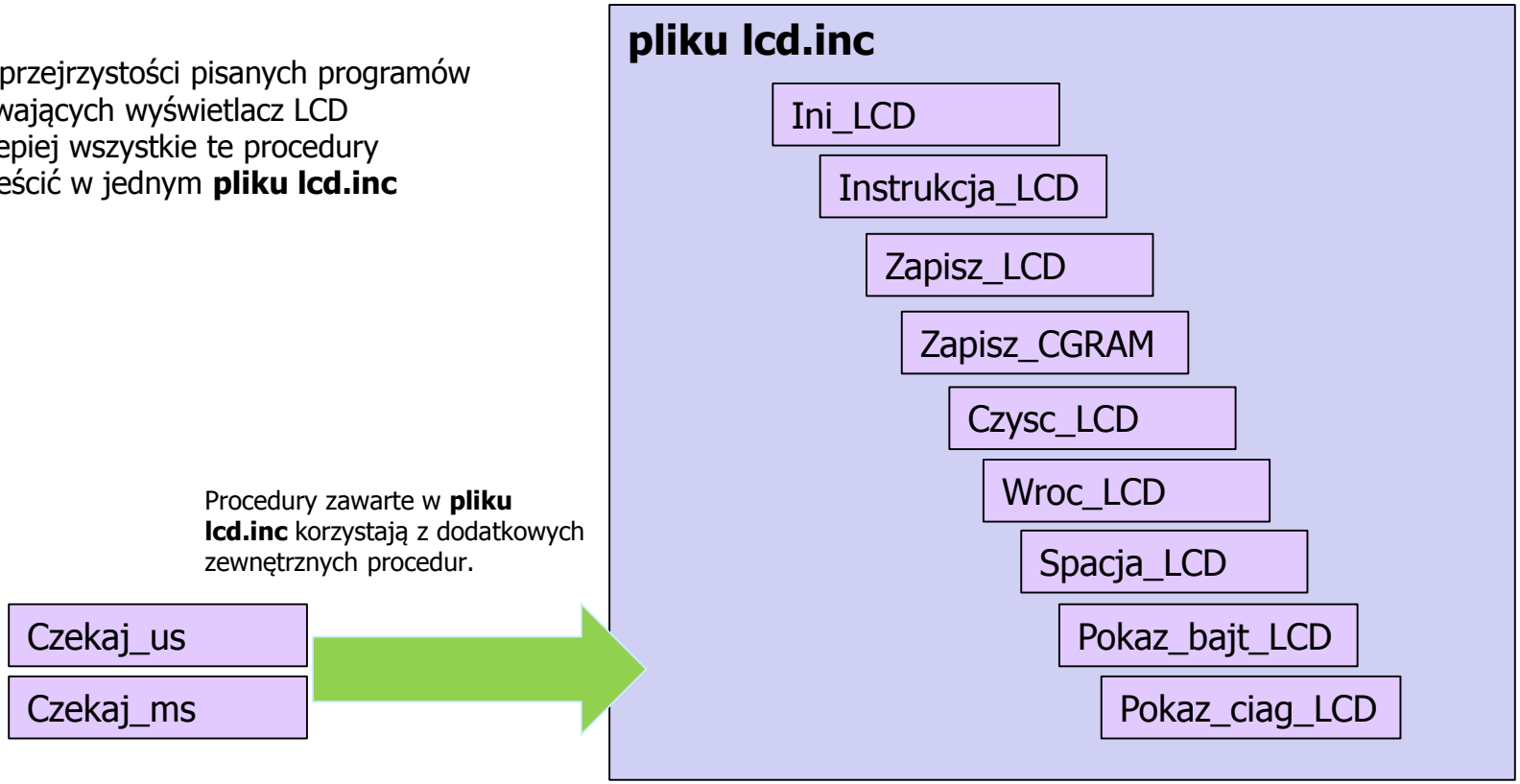
Wyświetlacz LCD

Procedura Ini_LCD:

Aby procedura inicjalizacyjna Ini_LCD była bardziej uniwersalna i miała więcej funkcjonalności można dodać do niej dodatkowe procedury:

Instrukcja_LCD	- przesyłanie instrukcji do wyświetlacza
Zapisz_LCD	- przesyłanie danych do wyświetlacza
Zapisz_CGRAM	- kopiowanie tablicy do pamięci CGRAM(zdefiniowane polskie znaki)
Czysc_LCD	- czyszczenie wyświetlacza
Wroc_LCD	- powrót kursora do pozycji początkowej
Spacja_LCD	- generacja odstępu
Pokaz_bajt_LCD	- wyświetlanie wartości bajtu
Pokaz_ciag_LCD	- wyświetlanie stałego łańcucha znaków

Dla przejrzystości pisanych programów używających wyświetlacz LCD najlepiej wszystkie te procedury umieścić w jednym **pliku lcd.inc**



PROCEDURA PRZESYŁAJĄCA INSTRUKCJĘ DO WYŚWIETLACZA

Instrukcja_LCD

```
; Parametry:
; R16          - kod instrukcji
;
; Przykład:
;                          ; włączanie kursora
;               ldi        R16, (1<<LCD_STER)|(1<<LCD_SW_ON)|(1<<LCD_SK_ON)|(1<<LCD_SM_OFF)
;               rcall      Instrukcja_LCD
;
*****************************************************************************************
               Instrukcja_LCD:
               cbi        O_LCD, LCD_RS          ; przesyłane wartości będą instrukcjami
               rcall      Zapisz_LCD             ; prześlij instrukcję zawartą w R16
               sbi        O_LCD, LCD_RS          ; przesyłane wartości będą danymi

               ret
```

PROCEDURA PRZESYLAJĄCA BAJT

Zapisz_LCD

```
; Parametry:
; R16         - dane do wysłania (np. kod wyświetlanego znaku);
*********************************************************************************************
Zapisz_LCD:
            sbi         O_LCD, LCD_EN           ; linia EN w stanie wysokim
            cbi         O_LCD, LCD_DB4
            cbi         O_LCD, LCD_DB5
            cbi         O_LCD, LCD_DB6
            cbi         O_LCD, LCD_DB7          ; linie DB7..4 w stanie niskim
            sbrc        R16, 7                  ; jeśli bit 7 rejestru R16 jest 0 to pomiń następną instrukcję
            sbi         O_LCD, LCD_DB7
            sbrc        R16, 6                  ; jeśli bit 6 rejestru R16 jest 0 to pomiń następną instrukcję
            sbi         O_LCD, LCD_DB6
            sbrc        R16, 5                  ; jeśli bit 5 rejestru R16 jest 0 to pomiń następną instrukcję
            sbi         O_LCD, LCD_DB5
            sbrc        R16, 4                  ; jeśli bit 4 rejestru R16 jest 0 to pomiń następną instrukcję
            sbi         O_LCD, LCD_DB4          ; ustawianie tylko tych linii DB7..4,
            nop                                 ; którym odpowiadają wysokie wartości
                                                ; bitów 7..4 z rejestru R16
            cbi         O_LCD, LCD_EN           ; linia EN w stanie niskim - potwierdzenie
                                                ; (przesłanie starszego półbajtu)
            nop
            nop                                 ; opóźnienie o dwa takty zegarowe

            sbi         O_LCD, LCD_EN
            cbi         O_LCD, LCD_DB4
            cbi         O_LCD, LCD_DB5
            cbi         O_LCD, LCD_DB6
            cbi         O_LCD, LCD_DB7
            sbrc        R16, 3
            sbi         O_LCD, LCD_DB7
            sbrc        R16, 2
            sbi         O_LCD, LCD_DB6
```

PROCEDURA PRZESYLAJĄCA BAJT

Zapisz_LCD

```
; Parametry:
; R16         - dane do wysłania (np. kod wyświetlanego znaku);
*************************************************************************************************
                sbrc        R16, 1
                sbi         O_LCD, LCD_DB5
                sbrc        R16, 0
                sbi         O_LCD, LCD_DB4          ;  ustawianie tylko tych linii DB7..4,
                nop                                 ;  którym odpowiadają wysokie wartości
                                                    ;  bitów 3..0 z rejestru R16
                cbi         O_LCD, LCD_EN           ;  linia EN w stanie niskim - potwierdzenie
                                                    ;  (przesłanie młodszego półbajtu)

                ldi         R16, 5
                rcall       Czekaj_us               ; opóźnienie o 50 us

                ret
```

PROCEDURA KOPIUJĄCA TABLICĘ STAŁYCH DO PAMIĘCI CGRAM

Zapisz_CGRAM

```
; Parametry:
; Z             - adres pierwszej komórki tablicy w pamięci programu
;
*****************************************************************************************
Zapisz_CGRAM:
                push        R16
                push        R17

                ldi         R16, 0|(1<<LCD_CGADR)
                rcall       Instrukcja_LCD          ;  wejście w tryb adresowania pamięci
                                                    ;  CGRAM i wymuszenie adresu zerowego

                ldi         R17, 64                 ;  inicjalizacja licznika pętli (wyk. 64x)
Zapisz_CGRAM_1:
                lpm         R16, Z+
                rcall       Zapisz_LCD              ;  zapisywanie w pamięci CGRAM kolejnych
                                                    ;  wartości tablicy pamięci programu,
                                                    ;  wskazywanej przez rejestr indeksowy Z
                dec         R17
                brne        Zapisz_CGRAM_1          ; koniec pętli

                ldi         R16, 0|(1<<LCD_DDADR)
                rcall       Instrukcja_LCD          ;  wejście w tryb adresowania pamięci DDRAM
                                                    ;  (ustawienie kursora na pozycji (1,1))

                pop         R17
                pop         R16
                ret
```

PROCEDURA KOPIUJĄCA TABLICĘ STAŁYCH DO PAMIĘCI CGRAM

Zapisz_CGRAM

```
; Parametry:
; Z             - adres pierwszej komórki tablicy w pamięci programu
;
*****************************************************************************************

                                        ; kształty polskich znaków diakrytycznych
Polskie_znaki:
.db 0b00000000,0b00000000,0b00001110,0b00000001,0b00001111,0b00010001,0b00001111,0b00000011 ; "ą"
.db 0b00000010,0b00000100,0b00001110,0b00010000,0b00010000,0b00010001,0b00001110,0b00000000 ; "ć"
.db 0b00000000,0b00000000,0b00001110,0b00010001,0b00011111,0b00010000,0b00001110,0b00000011 ; "ę"
.db 0b00001100,0b00000100,0b00000110,0b00001100,0b00000100,0b00000100,0b00001110,0b00000000 ; "ł"
.db 0b00000010,0b00000100,0b00010110,0b00011001,0b00010001,0b00010001,0b00010001,0b00000000 ; "ń"
.db 0b00000010,0b00000100,0b00001110,0b00010001,0b00010001,0b00010001,0b00001110,0b00000000 ; "ó"
.db 0b00000010,0b00000100,0b00001110,0b00010000,0b00001110,0b00000001,0b00011110,0b00000000 ; "ś"
.db 0b00000100,0b00000000,0b00011111,0b00000010,0b00000100,0b00001000,0b00011111,0b00000000 ; "ż"

                                        ; etykiety symboliczne kodów polskich znaków po załadowaniu
                                        ;   pamięci CGRAM wartościami z powyższej tablicy
.EQU POL_A  = 0         ;   "ą"
.EQU POL_C  = 1         ;   "ć"
.EQU POL_E  = 2         ;   "ę"
.EQU POL_L  = 3         ;   "ł"
.EQU POL_N  = 4         ;   "ń"
.EQU POL_O  = 5         ;   "ó"
.EQU POL_S  = 6         ;   "ś"
.EQU POL_Z  = 7         ;   "ż"
```

PROCEDURA CZYSZCZĄCA WYŚWIETLACZ LCD

Czysc_LCD

```
*****************************************************************************************
Czysc_LCD:
        push    R16

        ldi     R16, (1<<LCD_CZYSC)
        rcall   Instrukcja_LCD          ; kasowanie zawartości ekrany wyświetlacza LCD i
                                        ;   powrót kursora do pozycji
                                        ;   początkowej
        ldi     R16, 180
        rcall   Czekaj_us               ; opóźnienie o 1,8 ms

        pop     R16
        ret
```

PROCEDURA POWROTU KURSORA

Wroc_LCD

```
;*****************************************************************************************************
Wroc_LCD:
        push    R16

        ldi     R16, (1<<LCD_WROC)
        rcall   Instrukcja_LCD          ;   powrót kursora do pozycji
                                        ;   początkowej
        ldi     R16, 180
        rcall   Czekaj_us               ; opóźnienie o 1,8 ms

        pop     R16
        ret
```

PROCEDURA WPROWADZAJĄCA ODSTĘP

Spacja_LCD

```
; Parametry:
; R16          - liczba spacji do wyświetlenia
;
*****************************************************************************************

Spacja_LCD:
                push        R17

                mov         R17, R16            ; kopiuj liczbę wymaganych spacji do R17
                                                ; (inicjalizacja licznika pętli)
Spacja_LCD_1:
                ldi         R16, ' '            ; R16 zawiera kod ASCII pustego pola (spacji)
                rcall       Zapisz_LCD          ; wyświetlenie pustego pola (znaku z R16)
                dec         R17                 ; zmniejszenie licznika pętli
                brne        Spacja_LCD_1        ; iteracyjne zapętlenie

                pop         R17
                ret
```

PROCEDURA WYŚWIETLAJĄCA WARTOŚĆ BAJTU

Pokaz_bajt_LCD

```
; Parametry:
; R16          - wartość do wyświetlenia

*****************************************************************************************
Pokaz_bajt_LCD:
                push        R16
                push        R17

                mov         R17, R16            ; kopiuj wartość wyświetlaną do R17
                clt                             ; zeruj znacznik T, który zostanie ustawiony,
                                                ; gdy natrafimy na pierwszą niezerową cyfrę

                clr         R16                 ; pętla obliczająca cyfrę setek
Pok_b_LCD_1:
                subi        R17, 100            ; odejmij stała 100 od rejestru R17 bez przeniesienia
                brlo        Pok_b_LCD_2         ; skocz jeśli ujemne (bez znaku) C=1 - skok
                inc         R16
                rjmp        Pok_b_LCD_1
Pok_b_LCD_2:
                subi        R17, -100
                tst         R16                 ; sprawdź czy wartość zerowa czy ujemna Z=1 skok
                breq        Pok_b_LCD_3         ; jeśli liczba setek jest różna od 0, to..
                set                             ; ustaw znacznik T (dalsze cyfry muszą być
                                                ; wyświetlane, nawet jeśli są zerowe)
                ori         R16, 0b00110000     ; przekonwertuj ją na kod ASCII      ori-bitowa suma OR
                rcall       Zapisz_LCD          ; i wyświetl na LCD                  (dodane jest 48 do R16)

                clr         R16                 ; pętla obliczająca cyfrę dziesiątek
Pok_b_LCD_3:
                subi        R17, 10
                brlo        Pok_b_LCD_4
                inc         R16
                rjmp        Pok_b_LCD_3
```

PROCEDURA WYŚWIETLAJĄCA WARTOŚĆ BAJTU

Pokaz_bajt_LCD

```
; Parametry:
;R16          - wartość do wyświetlenia

*****************************************************************************************
Pok_b_LCD_4:
            subi        R17, -10
            tst         R16
            brne        Pok_b_LCD_5         ; jeśli liczba dziesiątek jest równa 0, to..
            brtc        Pok_b_LCD_6         ; jeśli znacznik T nie był ustawiony, to..
Pok_b_LCD_5:
            set                             ; ustaw znacznik T (dalsze cyfry muszą być
                                            ; wyświetlane, nawet jeśli są zerowe)
            ori         R16, 0b00110000     ; przekonwertuj ją na kod ASCII,
            rcall       Zapisz_LCD          ;  i wyświetl na LCD

Pok_b_LCD_6:
            mov         R16, R17            ; pozostałość jest cyfrą jednostek
            ori         R16, 0b00110000     ; przekonwertuj ją na kod ASCII (+48)
            rcall       Zapisz_LCD          ; i wyświetl na LCD

            pop         R17
            pop         R16
            ret
```

ori-bitowa suma OR
(dodane jest 48 do R16)

PROCEDURA WYŚWIETLAJĄCA STAŁY ŁAŃCUCH ZNAKÓW

Pokaz_ciag_LCD

```
; Parametry:
;Z              - adres pierwszego znaku łańcucha w pamięci programu

*****************************************************************************************
.EQU ZNAK_KONCA         = 255                   ; etykieta symboliczna znaku końca łańcucha

Pokaz_ciag_LCD:
                push    R16
                push    R17

Pok_c_LCD_1:                                    ; pętla wyświetlająca kolejne znaki łańcucha
                lpm     R16, Z+                 ; odczytaj znak z pamięci programu, spod adresu
                                                ; wskazywanego przez rejestr indeksowy Z
                cpi     R16, ZNAK_KONCA
                breq    Pok_c_LCD_2             ; jeśli znak jest znakiem końca, to zakończ,
                rcall   Zapisz_LCD              ; w przeciwnym razie wyświetl go na LCD
                rjmp    Pok_c_LCD_1             ; i przejdź do odczytywania kolejnego znaku

Pok_c_LCD_2:
                pop     R17
                pop     R16
                ret
```

MIKROSEKUNDOWA PROCEDURA OPÓŹNIAJĄCA

Czekaj_us

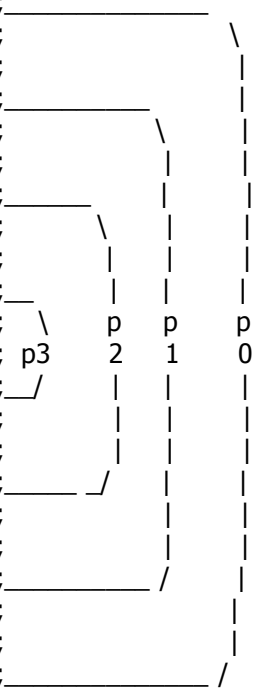
```
Parametry:
;               R16 - wartość opóźnienia w dziesiątkach us (dla zera - 2560 us)
; Wymagane stałe:
;               SYS_FREQ - częstotliwość pracy w MHz
*****************************************************************************************
Czekaj_us:
                push        R17
                push        R18

                mov         R18, R16
                ldi         R17, SYS_FREQ

Czekaj_us_0:
                mov         R16, R18            ;______
                                                ;      \
Czekaj_us_1:                                    ;       |
                nop                             ;__     |
                nop                             ;  \    |
                nop                             ;   |   |
                nop                             ;   |   |
                nop                             ;   p   p
                nop                             ;   1   0
                nop                             ;   |   |
                dec         R16                 ;   |   |
                brne        Czekaj_us_1         ;__/    |
                                                ;       |
                dec         R17                 ;       |
                brne        Czekaj_us_0         ;______ /

                pop         R18
                pop         R17
                ret
```

MIKROSEKUNDOWA PROCEDURA OPÓŹNIAJĄCA

Czekaj_ms

Parametry:
; R16 - wartość opóźnienia w ms (dla zera - 256 ms)
; R17 - mnożnik opóźnienia (dla zera - 256x)
; Wymagane stałe:
; SYS_FREQ - częstotliwość pracy w MHz
;
****

```
Czekaj_ms:
        push    R18
        push    R19
        push    R20
        push    R21

        mov     R20, R16
        mov     R21, R17
        ldi     R18, SYS_FREQ

Czekaj_ms_0:
        mov     R17, R21
Czekaj_ms_1:
        mov     R16, R20
Czekaj_ms_2:
        ldi     R19, 249
        nop
Czekaj_ms_3:
        nop
        dec     R19
        brne    Czekaj_ms_3

        dec     R16
        brne    Czekaj_ms_2

        dec     R17
        brne    Czekaj_ms_1

        dec     R18
        brne    Czekaj_ms_0

        pop     R21
        pop     R20
        pop     R19
        pop     R18
        ret
```



Programy z użyciem LCD

Program – który pozwoli zwizualizować wartości zmiennych zapisanych w rejestrach roboczych

Mając tak przygotowane procedury obsługi wyświetlacza LCD możemy wizualizować wyniki działań arytmetycznych.

(np. wynik dodawania będzie można wysłać na wyświetlacz LCD)

Zadanie - wynik dodawania wysyłamy na wyświetlacz LCD

```
; PODPROGRAM OBSŁUGI WYŚWIETLACZA LCD W TRYBIE 4-BITOWYM
;
; Etykiety procedur, które można wykorzystywać w programie:
;
;               Ini_LCD             - inicjalizacja wyświetlacza
;               Instrukcja_LCD      - przesyłanie instrukcji do wyświetlacza
;               Zapisz_LCD          - przesyłanie danych do wyświetlacza
;               Zapisz_CGRAM        - kopiowanie tablicy do pamięci CGRAM
;               Czysc_LCD           - czyszczenie wyświetlacza
;               Wroc_LCD            - powrót kursora do pozycji początkowej
;               Spacja_LCD          - generacja odstępu
;               Pokaz_bajt_LCD      - wyświetlanie wartości bajtu
;               Pokaz_ciag_LCD      - wyświetlanie stałego łańcucha znaków
;
; SEGMENT DEKLARACJI

.INCLUDE "m32def.inc"       ; plik nagłówkowy dla mikrokontrolera ATmega32
.EQU SYS_FREQ = 1           ; częstotliwość pracy w MHz, ustawiamy częstotliwość taktowania mikrokontrolera 1MHz
.EQU K_LCD = DDRD           ; rejestr kierunku portu wyświetlacza
.EQU O_LCD = PORTD          ; rejestr wyjściowy portu wyświetlacza
.EQU LCD_RS   = 0           ; nr linii dla sygnału RS podłączenie do portu PD0 mikrokontrolera
.EQU LCD_EN   = 1           ; nr linii dla sygnału EN podłączenie do portu PD1 mikrokontrolera
.EQU LCD_DB4  = 4           ; nr linii dla sygnału DB4 podłączenie do portu PD4 mikrokontrolera
.EQU LCD_DB5  = 5           ; nr linii dla sygnału DB5 podłączenie do portu PD5 mikrokontrolera
.EQU LCD_DB6  = 6           ; nr linii dla sygnału DB6 podłączenie do portu PD6 mikrokontrolera
.EQU LCD_DB7  = 7           ; nr linii dla sygnału DB7 podłączenie do portu PD7 mikrokontrolera
```

deklaracja wyprowadzeń mikrokontrolera podłączonych do wyświetlacza LCD

Zadanie - wynik dodawania wysyłamy na wyświetlacz LCD

```
.CSEG                       ; SEGMENT PROGRAMU
.ORG        0               ; wektor zerowania
rjmp        Reset           ; skok do procedury inicjalizującej

Reset:                                          ; podprogram inicjalizujący
            ldi             R17, high(RAMEND)
            ldi             R16, low(RAMEND)
            out             SPH, R17
            out             SPL, R16            ; inicjalizacja wskaźnika stosu

            rcall           Ini_LCD             ; inicjalizacja wyświetlacza LCD
            rcall           Czysc_LCD           ; kasowanie zawartości ekrany wyświetlacza LCD

;******************  PROGRAM DODAWANIE  ***********************************

; R16-dana do wysłania (wysyłane są kody ASCII liczb znajdującej się w R16)

ldi R16, 15                 ;ładuj rejestr R16 wartością 15
ldi R17, 35                 ;ładuj rejestr R17 wartością 35
add R16,R17                 ;Wynik dodawania zapisywany jest w rejestrze R16

rcall Pokaz_bajt_LCD        ; wartość rejestru R16 będzie wysłana do wyświetlacza LCD
                            ; będzie ona najpierw przekonwertowana na kod ASCII danej liczby

Petla:                      ; pętla główna programu lub zapętlenie
Rjmp Petla                  ; zapętl program


;****************** DOŁĄCZONE PROCEDURY  ***********************************

.INCLUDE "czekaj_ms.inc"    ;oczekiwanie milisekundowe
.INCLUDE "czekaj_us.inc"    ;oczekiwanie mikrosekundowe
.INCLUDE "lcd.inc"          ;obsługa wyświetlacza LCD
.exit
```

Zadanie - wynik mnożenia wysyłamy na wyświetlacz LCD

?????

Zadanie się komplikuje powyższa procedura Pokaz_bajt_LCD: **wyświetla tylko zawartość rejestru R16 czyli jeden baty. W wyniku mnożenia na przykład rejestru R16*R17 wynik zapisany jest w postaci słowa 16-bitowego umieszczonego w rejestrach R0,R1**

Musimy stworzyć nową procedurę Pokaz_word16bit_LCD: **która wyświetli na wyświetlaczu LCD wartość 16-bitową**

Dokładamy tą procedurę do pliku lcd.inc **i zapiszmy go jako nowy** lcd16b.inc **bogatszy o wyświetlanie wartości 16 bitową**

PROCEDURA WYŚWIETLAJĄCA WARTOŚĆ 16 BITOWĄ

Pokaz_word16bit_LCD:

```
;*************************************************************
;                   PROCEDURA WYŚWIETLAJĄCA WARTOŚĆ word 16bit
; Parametry:
;           R16 - wartość przekazywana do procedury Zapisz_LCD tą wartość wyświetla wyświetlacz LCD
;           R0, R1 słowo 16 bitowe do wyświetlenia na LCD
Pokaz_word16bit_LCD:
            push    R16
            push    R0
            push    R1
            push    R18
            push    R19
mov R19, R1                 ; wynik mnożenia zapisany jest 16bitowo w rejestrze R0 (lowbits) R1 (highbits)
mov R18, R0                 ; rejestry R0 R1 kopiujemy do R18 i R19 bo na nich nie działa jednostka ALU

            clt                         ; zeruj znacznik T, który zostanie ustawiony,
                                        ; gdy natrafimy na pierwszą niezerową cyfrę

            clr     R16                 ; zerujemy rejestr R16=0, R16 jest teraz zmienna iteracyjną pętli
Pok_16b_LCD_1:              ; pętla obliczająca cyfrę tysięcy
            subi R18, low(1000)         ; odejmij młodszą cześć stałej 1000 od rejestru R18 bez przeniesienia
            sbci R19, high(1000)        ; odejmij starszą cześć stałej 1000 od rejestru R19 z przeniesieniem
            brlo    Pok_16b_LCD_2       ; skocz jeśli ujemne (bez znaku) C=1 - skok
            inc     R16                 ; zmienna pokazująca ilość iteracji pętli
                                        ; mówi nam ile jest tysięcy w wartość 16bitowej trzymanej w R0,R1
            rjmp    Pok_16b_LCD_1
Pok_16b_LCD_2:
            subi R18, low(-1000)        ; jeśli odejmowaliśmy 1000 i po odjęci dostaliśmy liczbę ujemną
            sbci R19, high(-1000)       ; musimy wrócić do wartości dodatnie ale mniejszej od 1000
            tst     R16
            breq    Pok_16b_LCD_3       ; jeśli liczba tysięcy jest różna od 0, to..
            set                         ; ustaw znacznik T (dalsze cyfry muszą być
                                        ; wyświetlane, nawet jeśli są zerowe)
            ori     R16, 0b00110000     ; przekonwertuj ją na kod ASCII (+48)
            rcall   Zapisz_LCD          ; i wyświetl na LCD
```

ori-bitowa suma OR (dodane jest 48 do R16)

PROCEDURA WYŚWIETLAJĄCA WARTOŚĆ 16 BITOWĄ

Pokaz_word16bit_LCD:

```
clr         R16                     ; pętla obliczająca cyfrę setek
Pok_16b_LCD_3:
            subi R18, low(100)              ; odejmij młodszą cześć stałej 100 od rejestru R18 bez przeniesienia
            sbci R19, high(100)             ; odejmij starszą cześć stałej 100 od rejestru R19 z przeniesieniem
            brlo        Pok_16b_LCD_4       ; skocz jeśli ujemne (bez znaku) C=1 - skok
            inc         R16                 ; mówi nam ile jest setek w wartości 16bitowej trzymanej w R0,R1
            rjmp        Pok_16b_LCD_3
Pok_16b_LCD_4:
            subi R18, low(-100)             ;jeśli odejmowaliśmy 1000 i po odjęci dostaliśmy liczbę ujemną
            sbci R19, high(-100)            ;musimy wrócić do wartości dodatnie ale mniejszej od 100
            tst         R16
            brne        Pok_16b_LCD_5       ; jeśli liczba setek jest równa 0, to..
            brtc        Pok_16b_LCD_6       ; jeśli znacznik T nie był ustawiony, to..
Pok_16b_LCD_5:
            set                             ; ustaw znacznik T (dalsze cyfry muszą być
                                            ; wyświetlane, nawet jeśli są zerowe)
            ori         R16, 0b00110000     ; przekonwertuj ją na kod ASCII (+48)
            rcall       Zapisz_LCD          ; i wyświetl na LCD

clr         R16                     ; pętla obliczająca cyfrę dziesiątek
Pok_16b_LCD_6:
            subi R18, low(10)               ; odejmij młodszą cześć stałej 10 od rejestru R18 bez przeniesienia
            sbci R19, high(10)              ; odejmij starszą cześć stałej 10 od rejestru R19 z przeniesieniem
            brlo        Pok_16b_LCD_7       ; skocz jeśli ujemne (bez znaku) C=1 - skok
            inc         R16                 ;mówi nam ile jest dziesiątek w wartości 16bitowej trzymanej w R0,R1
            rjmp        Pok_16b_LCD_6
Pok_16b_LCD_7:
            subi R18, low(-10)              ; jeśli odejmowaliśmy 10 i po odjęci dostaliśmy liczbę ujemną
            sbci R19, high(-10)             ; musimy wrócić do wartości dodatnie ale mniejszej od 10
            tst         R16
            brne        Pok_16b_LCD_8       ; jeśli liczba setek jest równa 0, to..
            brtc        Pok_16b_LCD_9       ; jeśli znacznik T nie był ustawiony, to..
Pok_16b_LCD_8:
```

ori-bitowa suma OR
(dodane jest 48 do R16)

PROCEDURA WYŚWIETLAJĄCA WARTOŚĆ 16 BITOWĄ

Pokaz_word16bit_LCD:

```
            set                         ; ustaw znacznik T (dalsze cyfry muszą być
                                        ; wyświetlane, nawet jeśli są zerowe)
            ori     R16, 0b00110000     ; przekonwertuj ją na kod ASCII (+48)
            rcall   Zapisz_LCD          ; i wyświetl na LCD



Pok_16b_LCD_9:
            mov     R16, R18            ; pozostałość jest cyfrą jednostek      ori-bitowa suma OR
            ori     R16, 0b00110000     ; przekonwertuj ją na kod ASCII (+48)   (dodane jest 48 do R16)
            rcall   Zapisz_LCD          ; i wyświetl na LCD


            pop     R16
            pop     R0
            pop     R1
            pop     R18
            pop     R19
            ret

;*****************************************************************
```

Zadanie - wynik mnożenia wysyłamy na wyświetlacz LCD

```
.CSEG                       ; SEGMENT PROGRAMU
.ORG        0               ; wektor zerowania
rjmp        Reset           ; skok do procedury inicjalizującej

Reset:                                          ; podprogram inicjalizujący
            ldi             R17, high(RAMEND)
            ldi             R16, low(RAMEND)
            out             SPH, R17
            out             SPL, R16            ; inicjalizacja wskaźnika stosu

            rcall           Ini_LCD             ; inicjalizacja wyświetlacza LCD
            rcall           Czysc_LCD           ; kasowanie zawartości ekrany wyświetlacza LCD

;******************  PROGRAM MNOŻENIE  ***********************************

; R0, R1- dane do wysłania (wysyłane są kody ASCII liczb 16 bitowej znajdującej się w R0 i R1)

ldi R16, 12                 ;ładuj rejestr R16 wartością 12
ldi R17, 135                ;ładuj rejestr R17 wartością 135
mul R16, R17                ;Wynik mnożenia zapisywany jest w rejestrze R0 i R1

rcall Pokaz_word16bit_LCD   ; wartość rejestru R16 będzie wysłana do wyświetlacza LCD
                            ; będzie ona najpierw przekonwertowana na kod ASCII danej liczby

Petla:                      ; pętla główna programu lub zapętlenie
Rjmp Petla                  ; zapętl program


;****************** DOŁĄCZONE PROCEDURY  ***********************************

.INCLUDE "czekaj_ms.inc"    ;oczekiwanie milisekundowe
.INCLUDE "czekaj_us.inc"    ;oczekiwanie mikrosekundowe
.INCLUDE "lcd16b.inc"       ;obsługa wyświetlacza LCD
.exit
```

12*135= 1620
00000110 001010100
R0 = 84 = 01010100
R1 = 6 = 000001100