Programowanie mikrokontrolerów

Obsługa klawiszy i LCD

Marcin Engel Marcin Peczarski

Instytut Informatyki Uniwersytetu Warszawskiego

21 października 2012

Porty mikrokontrolera

- Każde wyprowadzenie może być skonfigurowane jako wyjście lub jako wejście (po wyzerowaniu mikrokontrolera domyślnie jest wejściem).
- Mamy cztery porty: `PA`, `PB`, `PC`, `PD`.
- Każdy port ma 8 wyprowadzeń oznaczonych `PA0`, ..., `PA7`, `PB0`, ..., `PB7` itd.
- Kierunek działania każdego z wyprowadzeń osobno można dowolnie zmieniać w trakcie pracy mikrokontrolera.
- Do ustalenia kierunku pracy poszczególnych wyprowadzeń portów służą rejestry we-wy `DDRA`, `DDRB`, `DDRC`, `DDRD`, przy czym bit o wartości:
  - 1 oznacza wyjście,
  - 0 oznacza wejście.

Przykłady

- Skonfigurowanie całego portu PA jako wyjścia.

```
SER  R16
OUT  DDRA, R16
```

- Skonfigurowanie wyprowadzeń PB0, PB1, PB2, PB3 jako wyjść, a wyprowadzeń PB4, PB5, PB6, PB7 jako wejść.

```
LDI  R16, 0b00001111
OUT  DDRB, R16
```

- Zmiana wyprowadzenia PC5 na wyjście.

```
SBI  DDRC, PC5
```

- Zmiana wyprowadzenia PD6 na wejście.

```
CBI  DDRD, PD6
```

Porty mikrokontrolera, cd.

- Jeśli wyprowadzenie jest skonfigurowane jako wyjście, to jego stan zależy od wartości odpowiedniego bitu rejestru we-wy `PORTA`, `PORTB`, `PORTC` lub `PORTD`, przy czym bit o wartości:
  - 1 oznacza stan wysoki (napięcie zasilania),
  - 0 oznacza stan niski (napięcie $\approx$ 0 V).
- Z rejestrów `PORTx` można też czytać – przeczytany zostanie stan wyjścia.
- Jeśli wyprowadzenie jest skonfigurowane jako wejście, to jego stan możemy przeczytać z odpowiedniego bitu rejestru we-wy `PINA`, `PINB`, `PINC` lub `PIND`, przy czym bit o wartości:
  - 1 oznacza stan wysoki (napięcie zasilania),
  - 0 oznacza stan niski (napięcie $\approx$ 0 V).

Rezystory podciągające

- Jeśli wyprowadzenie jest skonfigurowane jako wejście, to rejestry we-wy `PORTA`, `PORTB`, `PORTC` lub `PORTD`, włączają rezystor podciągający, przy czym bit o wartości:
  - 1 oznacza rezystor włączony,
  - 0 oznacza brak rezystora.
- Większość wyprowadzeń ma także drugą funkcję, będziemy je poznawać sukcesywnie.

Przykłady

- Zmiana stanu wyprowadzeń PD2 i PD3 na wysoki (jeśli są skonfigurowane jako wyjścia) lub włączenie na nich rezystorów podciągających (jeśli są skonfigurowane jako wejścia).

```
IN  R17, PORTD
ORI R17, 0b00001100
OUT PORTD, R17
```

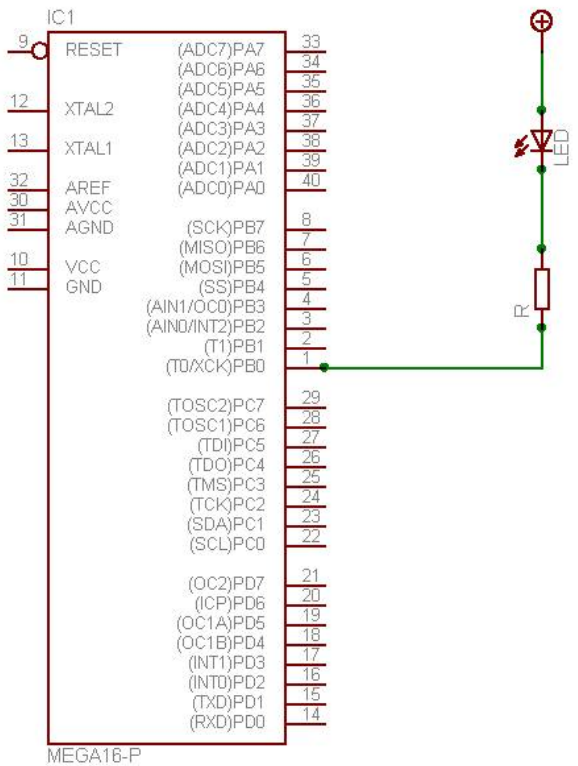
- Oczekiwanie na zmianę stanu wyprowadzenia PA2 z wysokiego na niski (PA2 musi być skonfigurowane jako wejście).

```
CZEKAJ:
     SBIC PINA, PA2
     RJMP CZEKAJ
```

- Oczekiwanie na zmianę stanu wyprowadzenia PA2 z niskiego na wysoki (PA2 musi być skonfigurowane jako wejście).

```
CZEKAJ:
     SBIS PINA, PA2
     RJMP CZEKAJ
```

Sposób podłączenia diody świecącej w zestawie



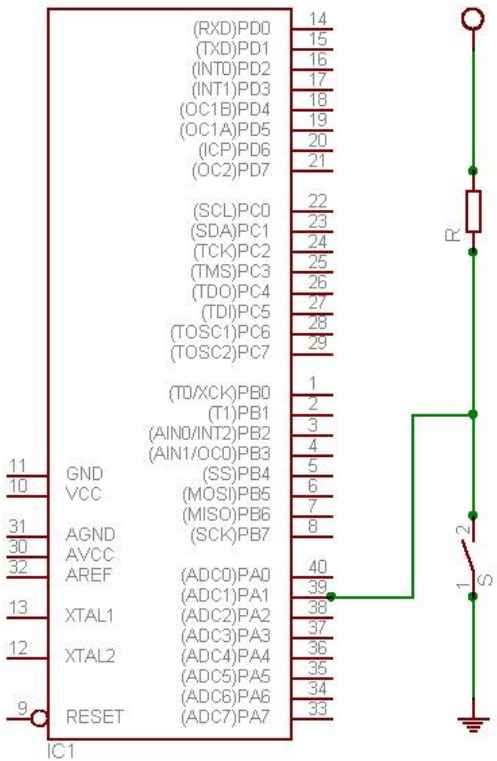
- Dioda świeci, gdy na wyprowadzeniu jest stan niski (0).

Diody świecące w VMLAB

- VMLAB umożliwia symulowanie 8 diod świecących oznaczonych od `D1` do `D8`.
- Dioda świeci, gdy na wyprowadzeniu jest stan niski (0).
- Przykładowa konfiguracja w pliku projektu, dioda `D4` jest podłączona przez rezystor `R4` o wartości 680 Ω do wyprowadzenia `PA3`.

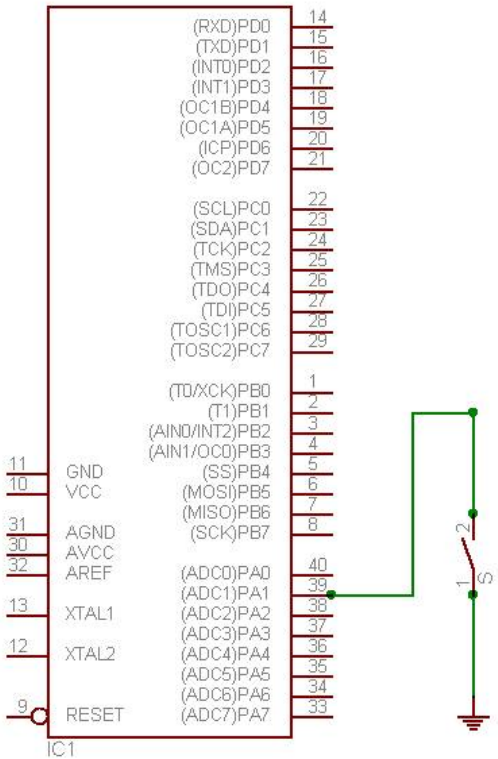
```
R4 PA3 N4 680
D4 VDD N4
```

Sposób podłączenia przycisku w zestawie



- Wyprowadzenie musi być ustawione jako wejście (odpowiedni bit w `DDRx` wyzerowany).
- Odczyt stanu przycisku odbywa się poprzez rejestr `PINx`.
- Stan wysoki oznacza przycisk otwarty.
- Stan niski – przycisk zamknięty.

Sposób podłączenia przycisku



- Wyprowadzenie musi być ustawione jako wejście (odpowiedni bit w `DDRx` wyzerowany).
- Wewnętrzny rezystor podciągający musi być włączony (odpowiedni bit w `PORTx` ustawiony).
- Odczyt stanu przycisku odbywa się poprzez rejestr `PINx`.
- Stan wysoki oznacza przycisk otwarty.
- Stan niski – przycisk zamknięty.

Przyciski w VMLAB

- VMLAB umożliwia symulowanie 16 przycisków oznaczonych od `K0` do `KF`.
- W panelu kontrolnym odpowiadają im przyciski oznaczone odpowiednio od „0” do „F”.
- Zakładamy, że wewnętrzne rezystory podciągające są włączone.
- Przycisk `K2` podłączony do wyprowadzenia `PA0`

  ```
  K2 PA0 GND
  ```

- Przycisk `KF` jako monostabilny podłączony do wyprowadzenia `PD3`, symulowany czas wciśnięcia 10 ms

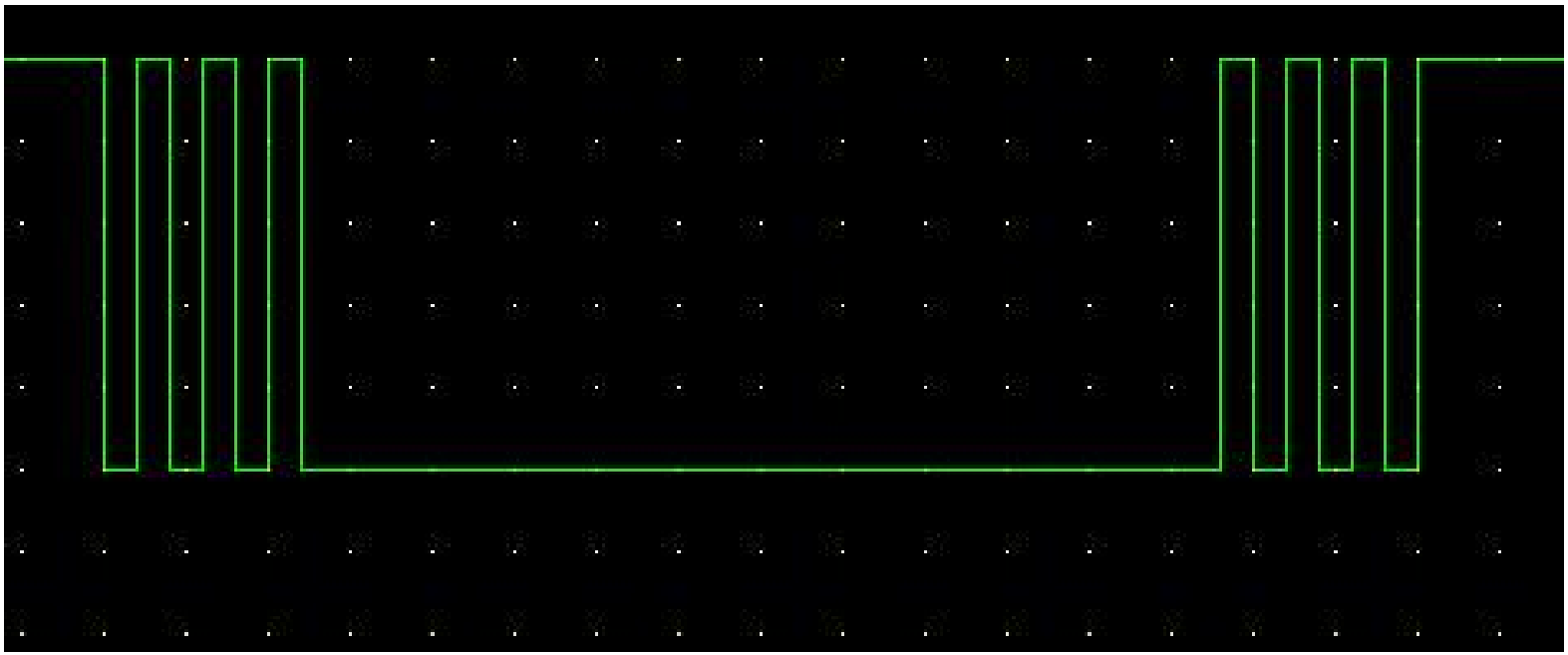
  ```
  KF PD3 GND MONOSTABLE(10m)
  ```

- Przycisk `K1` jako bistabilny podłączony do wyprowadzenia `PC0`

  ```
  K1 PC0 GND LATCHED
  ```

Zjawisko drgania styków

- W rzeczywistości wciśnięcie i puszczenie przycisku powoduje mikrodrgania.



- Stan stabilizuje się po pewnym czasie.

Symulacja tego zjawiska w VMLAB

- Przycisk K4 podłączony do wyprowadzenia PA1

```
P_left NRZ(2m) PA1
+ KEY_4 "0101010000000000000000000001010101"
+ RESET "1"
```

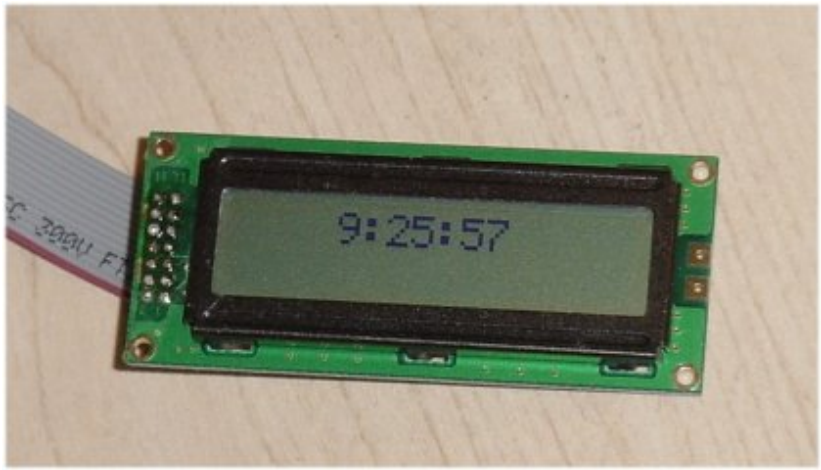
Algorytm obsługi przycisku bez powtarzania

```
if (klawisz wciśnięty) then
begin
  wait(T);
  if (klawisz wciśnięty) then
  begin
    obsłuż zdarzenie;
    while (klawisz wciśnięty)
      wait(T)
  end
end
```

Algorytm obsługi przycisku z powtarzaniem

```
if (klawisz wciśnięty) then
begin
  wait(T);
  while (klawisz wciśnięty) then
  begin
    obsłuż zdarzenie;
    licz := 0;
    while (klawisz wciśnięty and (licz < timeout))
    begin
      wait(T);
      licz := licz + 1
    end;
    zmodyfikuj(timeout)
  end
end
```

Alfanumeryczny wyświetlacz LCD



- Liczba wierszy: 1, 2 lub 4
- Liczba kolumn: 8, 16, 20, 24, 32 lub 40
- Umożliwia wyświetlanie znaków ze zbioru będącego rozszerzeniem ASCII.
- Posiada zintegrowany sterownik (zwykle zgodny z HD44780).
- Większe wyświetlacze mają dwa sterowniki!

Interfejs wyświetlacza

- Równoległy
- 14 lub 16 wyprowadzeń, typowo:
  - masa i zasilanie – odpowiednio wyprowadzenia 1 i 2
  - napięcie regulacji kontrastu – wyprowadzenie 3
  - linia RS (*Register Select*) – wyprowadzenie 4
    - 0 – przesyłanie instrukcji
    - 1 – przesyłanie danych
  - linia RW (*Read/Write*), kierunek transmisji – wyprowadzenie 5
    - 0 – zapis
    - 1 – odczyt
  - linia E (*Enable signal*), wyzwalanie zboczem opadającym – wyprowadzenie 6
  - linie danych D0 do D7 – odpowiednio wyprowadzenia 7–14
  - opcjonalnie dwie linie zasilające podświetlanie

Jakie polecenia wykonuje wyświetlacz?

- Czyszczenie zawartości ekranu
- Przeniesienie kursora na wskazaną pozycję
- Przesunięcie kursora w lewo lub w prawo
- Przesunięcie całego ekranu (okna) w lewo lub w prawo
- Zapis znaku na aktualnej pozycji kursora (do pamięci ekranu – DDRAM)
- Odczyt znaku z aktualnej pozycji pamięci ekranu
- Zapis/odczyt znaku do/z pamięci generatora znakowego (CGRAM)
- Polecenia konfiguracyjne, ustalające m.in.:
  - Liczbę wierszy, rozmiar znaków
  - Akcje po zapisaniu znaku (przesunięcie kursora lub okna)
  - Widoczność i migotanie kursora
  - Włączenie i wyłączenie wyświetlania

Pamięć wyświetlacza

- Wyświetlacz wyświetla znaki znajdujące się w określonych komórkach pamięci.
- Znaki pochodzą z rozszerzonego zbioru ASCII.
- Organizacja pamięci wyświetlacza 2×16:

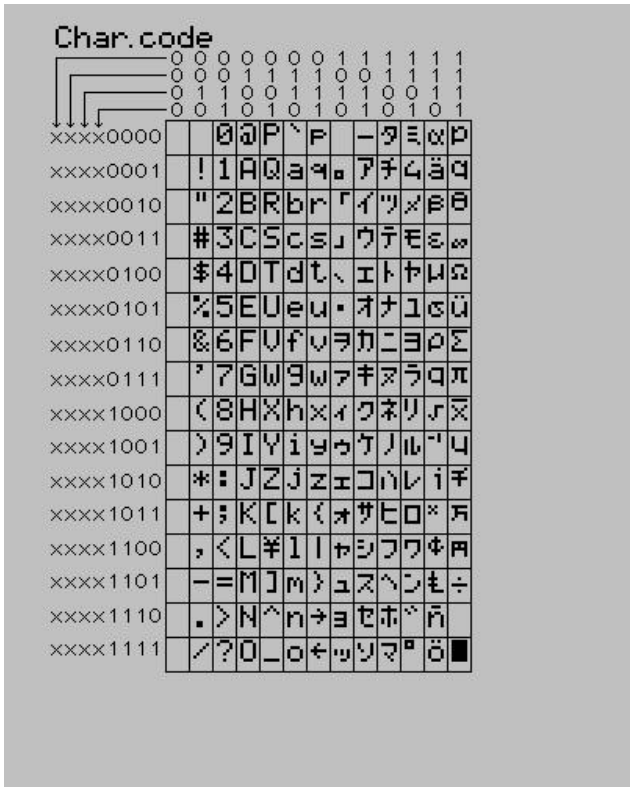
0	1	2	3	4	5	6	7	8	9	0A	0B	0C	0D	0E	0F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

- Organizacja pamięci wyświetlacza 4×16:

0	1	2	3	4	5	6	7	8	9	0A	0B	0C	0D	0E	0F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23
54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63

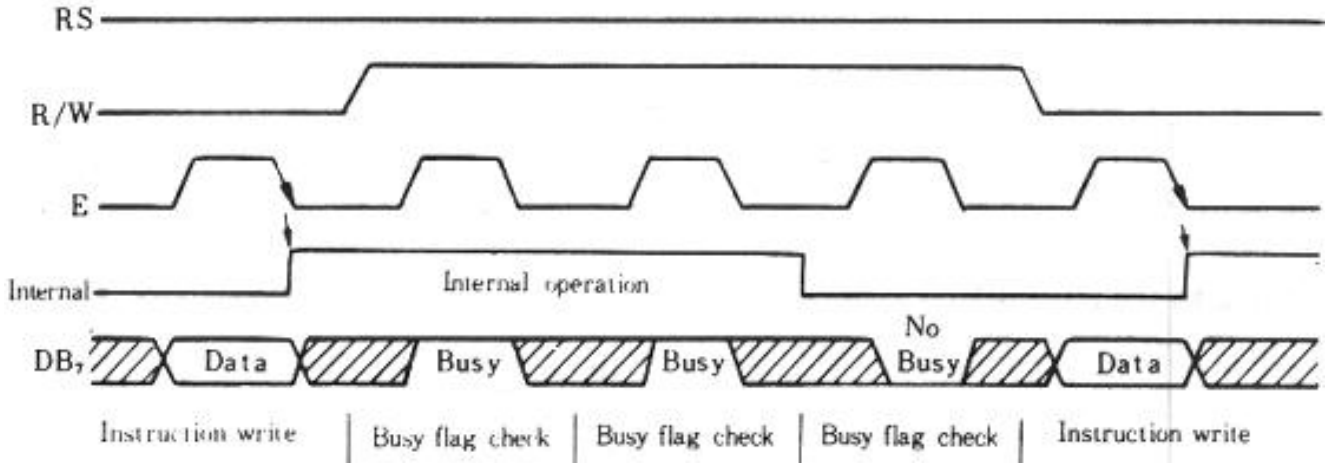
Zestaw znaków

- Standardowy zestaw znaków umieszczony w pamięci CGRAM:



- Można przeprogramować matrycę dowolnych 4 lub 8 znaków.

8-bitowy protokół przesłania danych do LCD

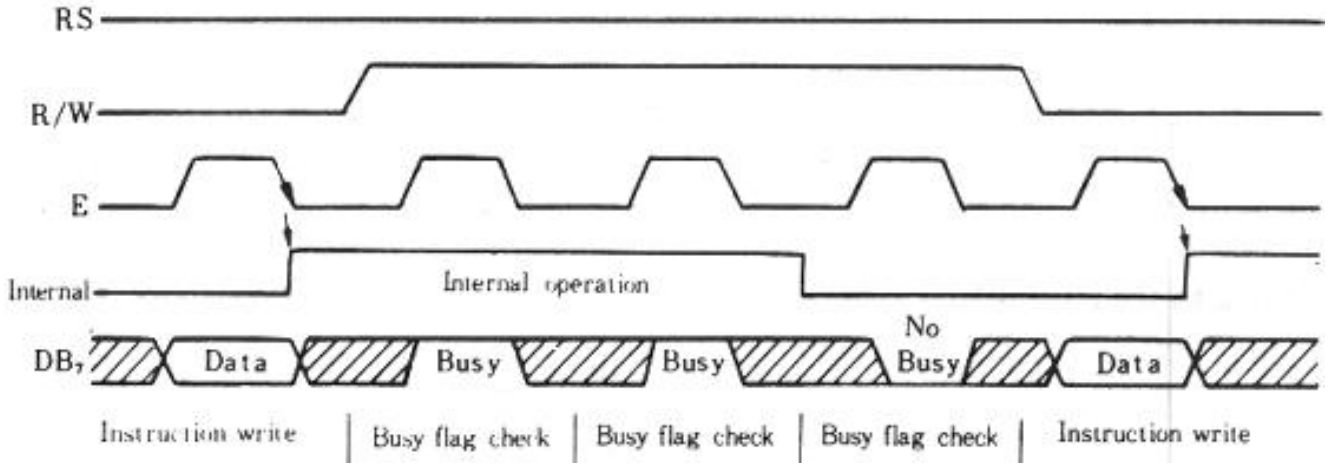


- Ustaw linię RW w stan 0.
- Ustaw rodzaj transmisji na linii RS:
  - 0 – przesyłanie instrukcji,
  - 1 – przesyłanie danych.
- Odczekaj min. 40 ns.
- Ustaw linię E w stan 1.
- Ustaw na liniach D0 do D7 odpowiednio bity 0 do 7.
- Odczekaj min. 80 ns.
- Ustaw linię E w stan 0, czas trwania stanu wysokiego min. 230 ns.
- Odczekaj min. 10 ns.

Uwagi

- Przy taktowaniu mikrokontrolera zegarem 1 MHz jeden takt trwa 1 μs, zatem odstęp czasu między wykonaniem kolejnych instrukcji jest wystarczający do uzyskania wymaganych opóźnień.
- Przy taktowaniu mikrokontrolera zegarem 16 MHz jeden takt trwa 62,5 ns, zatem dla zapewnienia wymaganych opóźnień może być konieczne wstawienia przed wyzerowaniem linii E jednej do trzech instrukcji `nop`.
- Wykonanie rozkazów chwilę trwa (od 40 μs do 1,64 ms)
- W tym czasie LCD nie będzie przyjmował kolejnych poleceń.
- Przed wysłaniem kolejnego rozkazu trzeba więc poczekać ...
- ...lub skorzystać z rozkazu odczytu danych.

Odczyt danych



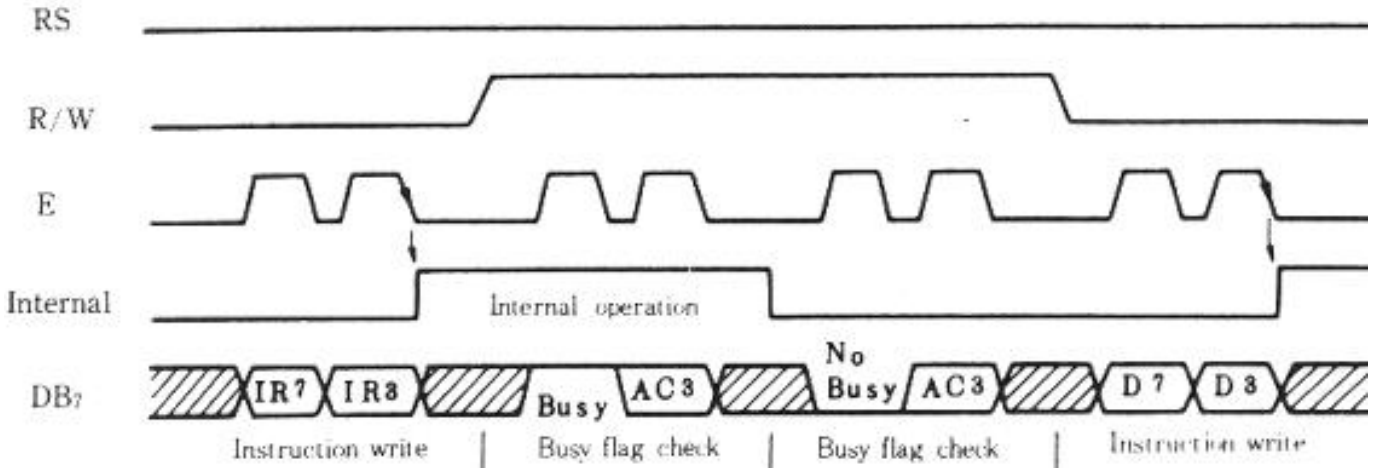
- Ustaw rodzaj transmisji na linii RS:
  - 0 – odczyt stanu i aktualnego adresu,
  - 1 – odczyt danych z pamięci DDRAM lub CGRAM.
- Ustaw linię RW w stan 1.
- Ustaw linię E w stan 1.
- Odczytaj dane, jeśli odczytujemy stan,
  - na linii D7 pojawi się 1, jeśli LCD był zajęty (nie był gotowy do kolejnej transmisji),
  - pozostałe bity zawierają aktualny adres,
- w p.p. na liniach danych pojawiają się odczytywane dane.
- Ustaw linię E w stan 0.

Interfejs uproszczony

Często stosuje się interfejs uproszczony:

- dane się tylko zapisuje – linia RW zawsze ustawiona na 0. (zwarta do masy)
- Interfejs 4-bitowy:
  - do transmisji używamy linii D4 do D7,
  - linie D0 do D3 zawsze ustawione na 0 (zwarte do masy),
  - po każdym rozkazie trzeba poczekać.

Protokół transmisji jednego bajtu – tryb 4-bitowy



- Ustaw rodzaj transmisji na linii RS (0 – instrukcja, 1 – dane).
- Ustaw linię E w stan 1.
- Ustaw na liniach D4 do D7 odpowiednio bity 4 do 7 transmitowanego bajtu.
- Ustaw linię E w stan 0.
- Ustaw linię E w stan 1.
- Ustaw na liniach D4 do D7 odpowiednio bity 0 do 3 transmitowanego bajtu.
- Ustaw linię E w stan 0.
- Wymagania czasowe jak dla trybu 8-bitowego.
- Minimalny odstęp między transmisją półbajtów wynosi 500 ns.

Inicjacja wyświetlacza

- Po włączeniu zasilania jesteśmy w trybie 8-bitowym.
- Odczekaj po włączeniu zasilania 40 ms.
- Wyślij instrukcję $(30)_{16}$ – młodszy półbajt jest ustawiony na stałe sprzętowo.
- Odczekaj 4,1 ms.
- Wyślij instrukcję $(30)_{16}$.
- Odczekaj 100 μs.
- Wyślij instrukcję $(30)_{16}$.
- Odczekaj 100 μs.
- Włącz interfejs 4-bitowy – wyślij instrukcję $(20)_{16}$.
- Odczekaj 40 μs.
- Dalej konfiguruj wyświetlacz, wysyłając instrukcje i dane w trybie 4-bitowym.

Rozkazy konfigurujące

- *Function set* – ustawia tryb (4-bitowy lub 8-bitowy), liczbę wierszy, rozmiar znaków.
  Może (i powinien!) być wykonany tylko jako pierwszy rozkaz.
- *Clear display* – czyści ekran oraz przesuwa okno do standardowej pozycji.
  Efekt uboczny: ustawia przesuwanie kursora w prawo po wypisaniu kolejnego znaku.
- *Return home* – przesuwa kursor i okno do początkowej pozycji.
- *Entry mode set* – konfiguruje zachowanie kursora i okna po wypisaniu kolejnego znaku (przesunięcie kursora w lewo lub w prawo, przesuwanie okna).
- *Display control* – włącza/wyłącza wyświetlacz i/lub kursor.

Tabela kodów operacji

- Tabelę kodów oraz szczegóły można znaleźć w tabeli 6 dokumentacji technicznej kontrolera HD44780.
- Linki są na stronie przedmiotu.

Tabela kodów operacji

Operacja	D7	D6	D5	D4	D3	D2	D1	D0	Czas
Clear display	0	0	0	0	0	0	0	1	1,64 ms
Return home	0	0	0	0	0	0	1	X	1,64 ms
Entry mode set	0	0	0	0	0	1	I/D	S	40 μs
Display control	0	0	0	0	1	D	C	B	40 μs
Cursor display shift	0	0	0	1	S/C	R/L	X	X	40 μs
Function set	0	0	1	DL	N	F	X	X	40 μs
Set DDRAM address	1	adres							40 μs

Kody operacji

- X – dowolna wartość bitu
- I/D = 1 – inkrementacja adresu po zapisie
  I/D = 0 – dekrementacja adresu po zapisie
  S = 1 – zamiast kursora przesuwa się okno
- D = 1 – wyświetlanie włączone
  C = 1 – kursor widoczny
  B = 1 – włączone miganie kursora
- S/C = 0 – przesuwanie kursora
  S/C = 1 – przesuwanie okna
  R/L = 0 – przesuwanie w lewo
  R/L = 1 – przesuwanie w prawo
- DL = 0 – interfejs 4-bitowy
  DL = 1 – interfejs 8-bitowy
  N = 0 – wyświetlacz 1-liniowy
  N = 1 – wyświetlacz 2-liniowy
  F = 0 – matryca znaku 5×8 pikseli
  F = 1 – matryca znaku 5×10 pikseli

LCD w zestawach

- 2 wiersze po 16 znaków, matryca znaku 5×8 pikseli
- Linia RW jest zwarta do masy.
- Linie D4 . . . D7 oraz RS i E są wyprowadzone na piny . . .
- . . . i domyślnie połączone zworkami z nogami portu D:
  - linia RS z PD0,
  - linia E z PD1,
  - linia D4 z PD2,
  - linia D5 z PD3,
  - linia D6 z PD4,
  - linia D7 z PD5.
- Bez podświetlania, choć płytka jest do tego przygotowana.

Symulacja LCD w VMLAB

- VMLAB potrafi symulować wyświetlacz LCD.
- Ekran wyświetlacza można oglądać w panelu sterującym.
- Jest dostępny podgląd pamięci DDRAM.
- Można włączyć rejestrowanie odbieranych rozkazów.
- CGRAM i polecenia z nią związane nie są symulowane.
- Konfiguracja w pliku projektu dostosowana do zestawu:

```
XLCD LCD(16 2 250k) pd0 gnd pd1 pd5 pd4 pd3 pd2
                                    gnd gnd gnd gnd
```

Inicjacja wyświetlacza w zestawie bardziej szczegółowo

- Będziemy wysyłać instrukcje:
  - zeruj linię RS (PD0),
  - zeruj linię E (PD1).
- Odczekaj 40 ms.
- Wyślij pierwszą instrukcję $(30)_{16}$:
  - ustaw linię E (PD1),
  - ustaw linię D4 (PD2),
  - ustaw linię D5 (PD3),
  - zeruj linię D6 (PD4),
  - zeruj linię D7 (PD5),
  - czekaj 1 do 4 taktów zegara (0 do 3 instrukcji, np. `nop` lub jakieś bardziej użyteczne),
  - zeruj linię E (PD1).
- Odczekaj 4,1 ms.
- Wyślij drugą instrukcję $(30)_{16}$:
  - ustaw linię E (PD1),
  - czekaj 1 do 4 taktów zegara (0 do 3 instrukcji),
  - zeruj linię E (PD1).
- Odczekaj 100 μs.

Inicjacja wyświetlacza w zestawie bardziej szczegółowo, cd.

- Wyślij trzecią instrukcję $(30)_{16}$:
  - ustaw linię E (PD1),
  - czekaj 1 do 4 taktów zegara (0 do 3 instrukcji),
  - zeruj linię E (PD1).
- Odczekaj 100 μs.
- Włącz interfejs 4-bitowy – wyślij instrukcję $(20)_{16}$:
  - ustaw linię E (PD1),
  - zeruj linię D4 (PD2),
  - czekaj 1 do 4 taktów zegara (0 do 3 instrukcji),
  - zeruj linię E (PD1).
- Odczekaj 40 μs.
- Dalej konfiguruj wyświetlacz, wysyłając instrukcje i dane w trybie 4-bitowym.

Transmisja jednego bajtu w trybie 4-bitowym bardziej szczegółowo

- Ustaw rodzaj transmisji na linii RS (PD0): 0 – instrukcja, 1 – dane.
- Ustaw linię E (PD1).
- Ustaw na liniach D4 (PD2) do D7 (PD5) odpowiednio bity 4 do 7 transmitowanego bajtu.
- Odczekaj 1 do 4 taktów zegara.
- Zeruj linię E (PD1).
- Odczekaj 1 do 3 taktów zegara.
- Ustaw linię E (PD1).
- Ustaw na liniach D4 (PD2) do D7 (PD5) odpowiednio bity 0 do 3 transmitowanego bajtu.
- Odczekaj 1 do 4 taktów zegara.
- Zeruj linię E (PD1).
- Odczekaj 40 μs.

Organizacja kodu w Asemblerze

- Asembler oferuje niewielkie możliwości strukturalizacji kodu.
- Warto przygotować sobie bibliotekę często używanych procedur: pętle opóźniające, obsługa LCD.
- Takie biblioteki należy pisać w miarę możliwości ogólnie, korzystając z nazw symbolicznych.
- Na początku każdego takiego „modułu” warto opisać, co trzeba ustawić w programie korzystającym z niego.
- Warto ustalić sobie pewne konwencje dotyczące zachowania procedur i ich parametrów, np.:
  - procedury nie modyfikują żadnych rejestrów
  - pierwszy parametr jest zawsze w rejestrze `R16`, kolejny w `R17` itd.

Przykładowa parametryzacja modułu obsługi LCD

```
; LCD_DATA_PORT - port danych
; LCD_E_PORT    - port linii wyzwalajacej E
; LCD_RS_PORT   - port linii RS

; LCD_DATA_DDR  - rejestr kierunku danych
; LCD_E_DDR     - rejestr kierunku linii E
; LCD_RS_DDR    - rejestr kierunku linii RS

; OE - numer bitu linii OE
; RS - numer bitu linii RS
; D4 - numer bitu linii D4
; D5 - numer bitu linii D5
; D6 - numer bitu linii D6
; D7 - numer bitu linii D7
```