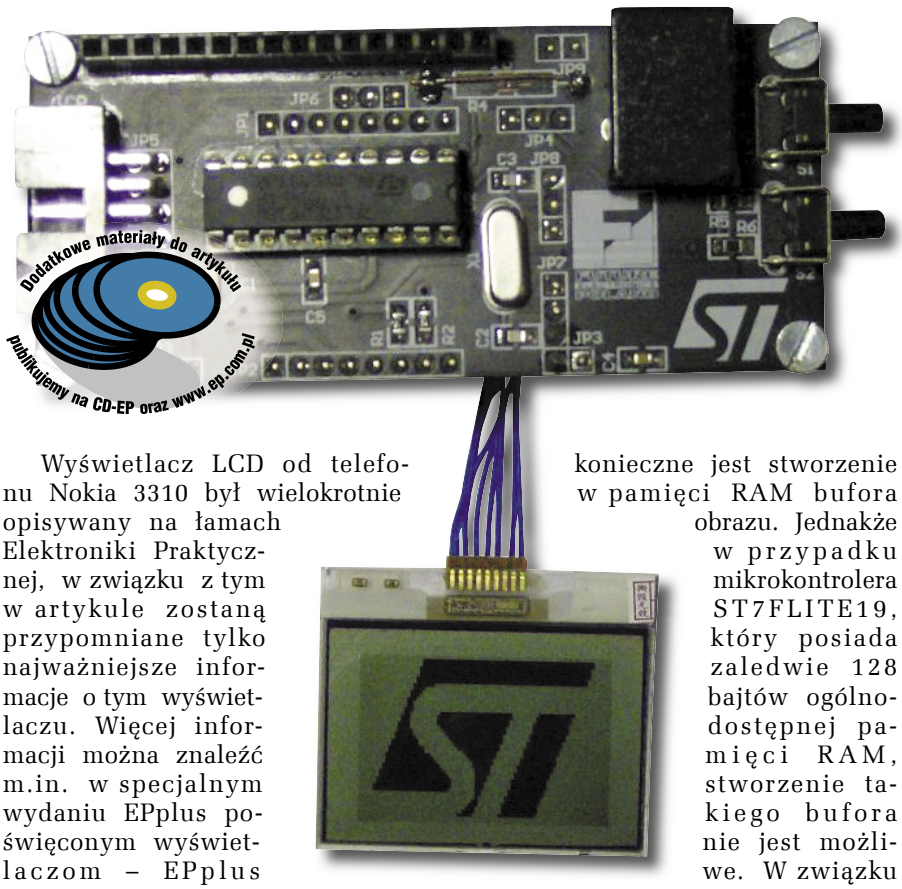
LITEcomp – aplikacje

Współpraca z wyświetlaczem LCD od telefonu NOKIA 3310

Bywa, że standardowy wyświetlacz alfanumeryczny 2x16 znaków będący „podstawowym" monitorem LITEcompa okazuje się niewystarczający dla potrzeb tworzonej aplikacji. W takim przypadku można wykorzystać popularny wyświetlacz LCD stosowany w telefonach Nokia 3310, którego największą zaletą jest bardzo niska cena i małe wymiary. Przykład współpracy takiego wyświetlacza z modułem LITEcomp przedstawiamy niżej.



Zrobione na LITEcomple

st7.ep.com.pl

LITEcomp

LITEcomp jest prostym komputerkiem wykonanym na mikrokontrolerze ST7FLITE19. LITEcomp jest w ramach promocji dodawany bezpłatnie do książki „Mikrokontrolery ST7LITE w praktyce" (autor Jacek Bogusz). Książka jest dostępna w *sklep.avt.pl* (numer katalogowy KS–260905).

Wyświetlacz LCD od telefonu Nokia 3310 był wielokrotnie opisywany na łamach Elektroniki Praktycznej, w związku z tym w artykule zostaną przypomniane tylko najważniejsze informacje o tym wyświetlaczu. Więcej informacji można znaleźć m.in. w specjalnym wydaniu EPplus poświęconym wyświetlaczom – EPplus 2/2007 „Displays". Warto również sięgnąć po podstawowe źródło, jakim jest dokumentacja układu PCD8544 dostępna na stronie *www.nxp.com*.

Komunikacja z kontrolerem wyświetlacza odbywa się poprzez jednokierunkowy interfejs szeregowy wspomagany dwoma liniami kontrolnymi: D/C (informuje kontroler czy transmitowany bajt jest rozkazem czy daną) oraz RESET (linia służącą do wyzerowania kontrolera). Maksymalna częstotliwość taktowania interfejsu szeregowego nie może przekraczać 4 MHz. Wyświetlacz posiada 504 bajty pamięci obrazu, które są zorganizowane w sześć stron po 84 bajty, co przekłada się na organizację ekranu 84x48 pikseli. Ponieważ nie jest możliwy odczyt zawartości pamięci obrazu, w celu pełnego wykorzystania możliwości graficznych konieczne jest stworzenie w pamięci RAM bufora obrazu. Jednakże w przypadku mikrokontrolera ST7FLITE19, który posiada zaledwie 128 bajtów ogólnodostępnej pamięci RAM, stworzenie takiego bufora nie jest możliwe. W związku z tym funkcjonalność wyświetlacza jest ograniczona wyłącznie do wyświetlania tekstu oraz obrazów przygotowanych na komputerze PC i zapisanych w pamięci programu. Nie będzie możliwe rysowanie figur geometrycznych czy linii bezpośrednio przez mikrokontroler.

Schemat układu jest przedstawiony na **rys. 1**. Napięcie zasilające nie może przekraczać +3,3 V ze względu na maksymalne napięcie pracy układu PCD8544. Przekroczenie tej wartości może spowodować trwałe uszkodzenie kontrolera wyświetlacza. W związku z niższym niż zwykle napięciem pracy modułu LITEcomp należy zwrócić uwagę na konfigurację bajtów Option Bytes mikrokontrolera ST7FLITE19, a w szczególności na konfigurację układu LVD. Nie należy oczywiście ustawiać progu

zadziałania układu na 4,1 V, gdyż mikrokontroler w takim przypadku nigdy by nie wystartował. Nie należy również wyłączać układu LVD, gdyż z doświadczeń przeprowadzonych przeze mnie przy takiej konfiguracji wynikło, iż mikrokontroler startuje znacznie wcześniej, niż układ PCD8544 jest zdolny do przyjmowania poleceń, co objawia się brakiem reakcji wyświetlacza na wysyłane przez mikrokontroler polecenia. W związku z tym próg zadziałania układu LVD należy ustawić na ok. 2,8 V.

Procedury obsługi interfejsu SPI

Mikrokontroler ST7FLITE19 komunikuje się z kontrolerem wyświetlacza za pomocą sprzętowego układu SPI, co znacznie upraszcza program w porównaniu do programowej realizacji interfejsu SPI. Przygotowałem dwie krótkie procedury: *initSPI*, która konfiguruje układ SPI oraz *sendSPI*, która wysyła bajt przekazany przez rejestr A. Kod procedur jest przedstawiony na **list. 1.**

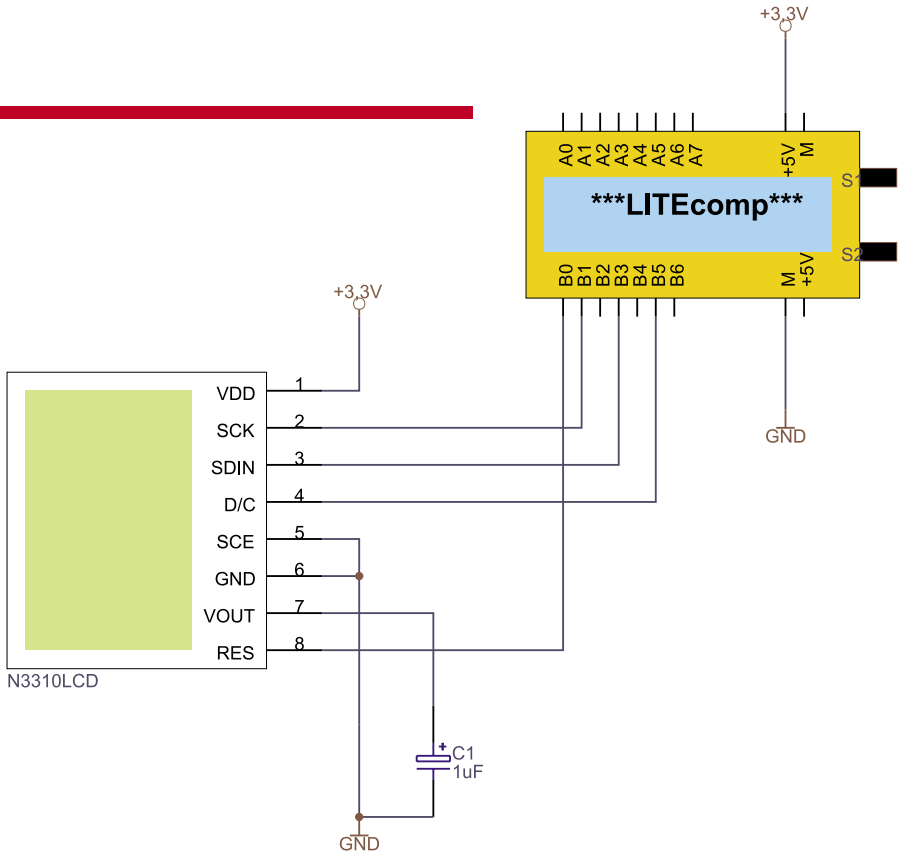
Zapis rozkazu i danej do wyświetlacza

Najważniejszymi procedurami wykorzystywanymi podczas komunikacji z wyświetlaczem są procedury zapisu rozkazu oraz danych do kontrolera. Rozkaz jest zapisywany przy niskim stanie na wyprowadzeniu D/C, natomiast dane są zapisywane przy wysokim stanie. Kod procedur zapisu rozkazu i danej do wyświetlacza jest przedstawiony na **list. 2.**

Inicjalizacja wyświetlacza

Procedura inicjalizacji kontrolera wyświetlacza rozpoczyna się od skonfigurowania wyprowadzeń I/O kontrolujących linie DC i RES oraz inicjalizacji interfejsu SPI. Następnie jest dokonywana konfiguracja kontrolera zgodnie z danymi zawartymi w dokumentacji układu PCD8544 oraz czyszczona jest zawartość pamięci obrazu. Kod procedury inicjalizacji wyświetlacza przedstawiono na **list. 3**.

Przed nawiązaniem komunikacji z kontrolerem wyświetlacza konieczne jest jego wyzerowanie poprzez podanie na wyprowadzenie RST impulsu rozpoczynającego się opadający zboczem. Kod procedury generującej impuls zerujący jest przedstawiony na **list. 4**.



Rys. 1. Schemat interfejsu dla wyświetlacza graficznego od telefonu NOKIA 3310

Czyszczenie pamięci obrazu

Wyczyszczenie zawartości pamięci obrazu sprowadza się do zapisania bajtów o wartości zero do wszystkich 504 komórek pamięci. Kod procedury czyszczącej pamięć obrazu jest przedstawiony na list. 5.

Wyświetlenie znaku

Procedura wyświetlająca znak jest stosunkowo skomplikowana ze względu na fakt, iż kontroler PCD8544 nie posiada własnej tablicy czcionek. W związku z tym konieczne jest przechowywanie tablicy z czcionkami w pamięci programu mikrokontrolera ST7FLITE19. Tablica ze wzorami czcionek jest zdefiniowana w pliku *font5x8.asm*. Parametrem procedury

List. 1. Procedury *initSPI* i *sendSPI*

```
;*****************************************************************************
; Procedura inicjalizacji interfejsu SPI
;*****************************************************************************
initSPI
     LD     A, #%00000011  ; programowe zarządzanie trybem pracy, SS -> GPIO
     LD     SPISR, A
     LD     A, #%01010000  ; MASTER: Enable, Fsck = Fcpu/8
     LD     SPICR, A
     RET

;*****************************************************************************
; Procedura  transmitująca bajt danych przez interfejs SPI
;*****************************************************************************
sendSPI
     LD     SPIDR, A       ; załadowanie danej do wysłania do SPIDR
     BTJF   SPISR, #7, *   ; oczekiwanie na zakończenie transmisji
     LD     A, SPIDR       ; załadowanie odebranej danej do A
     RET
```

List. 2. Kod procedur zapisu rozkazu i danej do wyświetlacza

```
;*****************************************************************************
; Procedura zapisująca do wyświetlacza rozkaz
;*****************************************************************************
.N3310_WriteCommand
     BRES   PBDR, #N3310_DC
     CALL   sendSPI
     RET

;*****************************************************************************
; Procedura zapisująca do wyświetlacza daną
;*****************************************************************************
.N3310_WriteData
     BSET   PBDR, #N3310_DC
     CALL   sendSPI
     RET
```

wyświetlającej znak jest jego kod ASCII. Ponieważ tablica czcionek zawiera definicje znaków od znaku o kodzie 32 (spacja), konieczne jest odjęcie od przekazanego przez rejestr A kodu liczby 32. Następnie tak otrzymana liczba jest mnożona przez liczbę bajtów składających się na jeden znak, czyli przez pięć. Otrzymany w ten sposób indeks tablicy jest dodawany do adresu pierwszego elementu i wynik jest ładowany do wskaźnika `fontPtr` i na podstawie tego adresu oraz zawartości rejestru X odczytywany jest bajt z tablicy czcionek. Odczyt z tablicy jest wykonywany pięciokrotnie w pętli wraz z inkrementacją wartości rejestru X. W ten sposób zostaną odczytane wszystkie bajty składające się na znak o określonym kodzie ASCII. Po wyświetleniu wszystkich pięciu bajtów składowych znaku do pamięci wyświetlacza zapisywany jest bajt o wartości zero, który jest odstępem pomiędzy kolejnymi znakami. Kod procedury wyświetlającej znak jest przedstawiony na **list. 6**.

Procedura wyświetlenia znaku stanowi podstawę procedur wyświetlających tekst, wartości liczb itp. Zaprezentowane we wcześniejszych odcinkach kursu procedury prezentacji danych alfanumerycznych wykorzystujące procedurę *writechar* można zaadaptować do współpracy z wyświetlaczem prezentowanym w bieżącym artykule w bardzo prosty sposób. Wystarczy po prostu zamienić wywołanie procedury *writechar* na *N3310_WriteChar*.

List. 3. Kod procedury inicjalizacji wyświetlacza

```
;*****************************************************************************
; Procedura inicjalizacji wyświetlacza
;*****************************************************************************
.N3310_Init
     BSET   PBDDR, #N3310_DC
     BSET   PBOR, #N3310_DC
     BSET   PBDDR, #N3310_RES
     BSET   PBOR, #N3310_RES
     BSET   PBDR, #N3310_RES

     CALL   initSPI

     CALL   N3310_Reset

     LD     A, #N3310C_FS
     OR     A, #N3310C_FS_EXTENDED
     CALL   N3310_WriteCommand

     LD     A, #N3310C_VOP
     OR     A, #72
     CALL   N3310_WriteCommand

     LD     A, #N3310C_TEMP
     OR     A, #2
     CALL   N3310_WriteCommand

     LD     A, #N3310C_BIAS
     OR     A, #3
     CALL   N3310_WriteCommand

     LD     A, #N3310C_FS
     OR     A, #N3310C_FS_BASIC
     OR     A, #N3310C_FS_HORIZONTAL
     CALL   N3310_WriteCommand

     LD     A, #N3310C_DC
     OR     A, #N3310C_DC_NORMAL
     CALL   N3310_WriteCommand

     CALL   N3310_ClearScreen
     RET
```

List. 4. Kod procedury generującej impuls zerujący

```
;*****************************************************************************
; Procedura generująca ujemny impuls na linii Reset
;*****************************************************************************
.N3310_Reset
     BRES   PBDR, #N3310_RES
     NOP
     NOP
     BSET   PBDR, #N3310_RES
     RET
```

List. 5. Kod procedury czyszczącej pamięć obrazu

```
;*****************************************************************************
; Procedura czyszcząca pamięć obrazu
;*****************************************************************************
.N3310_ClearScreen
     CLR    X               ;
     CLR    Y               ; ustawienie współrzędnych ekranu X = 0 Y = 0
     CALL   N3310_GoTo      ;

     LD     Y, #6           ; liczba wierszy
ClearScreenLoop1
     LD     X, #84          ; liczba kolumn
ClearScreenLoop2
     LD     A, #0           ;
     CALL   N3310_WriteData ; zapis wartości czyszczącej ekran

     DEC    X               ; X-- i sprawdzenie czy to ostatnia kolumna?
     JRNE   ClearScreenLoop2  ; jeśli nie to skok do ClearScreenLoop2

     DEC    Y               ; Y-- i sprawdzenie czy to ostatni wiersz?
     JRNE   ClearScreenLoop1  ; jeśli nie to skok do ClearScreenLoop1
     RET
```

Wyświetlenie tekstu

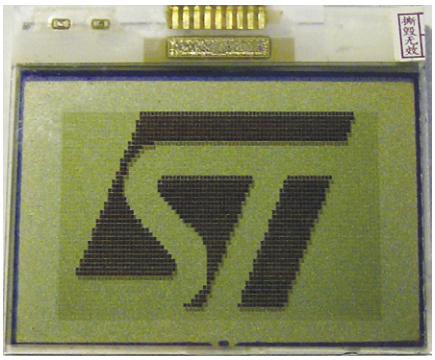
Wyświetlenie tekstu na wyświetlaczu realizowane jest przez makroinstrukcję *WTXT*, której kod znajduje się w pliku *n3310lcd.inc*. Kod procedury jest prawie identyczny jak w przypadku wyświetlana napisów na wyświetlaczu alfanumerycznym HD44780 z tą różnicą, że zamiast procedury *writechar* wywoływana jest procedura *N3310_WriteChar*. Kod makroinstrukcji *WTXT* przedstawiony jest na **list. 7.**

Parametrem makroinstrukcji jest adres pierwszego znaku tekstu zapisanego w pamięci (etykieta). Adres ten musi być znany na etapie kompilacji programu. Tekst musi być zakończony bajtem o wartości zero. Efekt zastosowania procedury wyświetlającej tekst jest przedstawiony na **fot. 2**.

Procedura wyświetlająca bitmapę

Na wyświetlaczu można wyświetlić obraz graficzny o rozmiarach 84x48 pikseli. Bitmapa wy-

Fot. 2. Efekt działania procedury wyświetlającej tekst



Fot. 3. Przykładowa bitmapa wyświetlona za pomocą procedury *N3310_Bitmap*

pełnia cały ekran począwszy od współrzędnych (0,0) i musi mieć rozmiar 84x48 pikseli. W przypadku, gdy rzeczywisty obraz, który zamierzamy wyświetlić, jest mniejszych rozmiarów, należy przygotować bitmapę o rozmiarach 84x48 i niewykorzystane miejsce pozostawić białe. Adres tablicy z zawartością obrazu należy przekazać przez rejestry X:Y. Bitmapa może być przechowywana w dowolnym obszarze w przestrzeni adresowej mikrokontrolera: RAM, EEPROM lub FLASH. Kod procedury wyświetlającej bitmapę przedstawiony jest na **list. 8**.

Przykładową bitmapę wyświetloną za pomocą procedury *N3310_Bitmap* przedstawiono na **fot. 3**. Do konwersji bitmapy z postaci pliku *.bmp na postać źródłową asemblera ST7 opracowałem specjalny program dla komputera PC (**rys. 4**). Obsługa programu sprowadza się do wskazania lokalizacji

Rys. 4. Winietka programu służącego do konwersji bitmapy z pliku *.bmp na postać źródłową asemblera ST7

List. 6. Kod procedury wyświetlającej znak

```
;*****************************************************************************
; Procedura wyświetlająca na wyświetlaczu znak o kodzie ASCII przekazanym
w A
;*****************************************************************************
.N3310_WriteChar
     PUSH    X                  ; zapamiętanie X na stosie
     SUB     A, #32             ; konwersja kodu
     LD      X, A               ;
     LD      A, #5              ; obliczenie przesunięcia (5* kod znaku)
     MUL     X, A               ;

     LD      {fontPtr+0},X      ; przepisanie wyniku mnożenia do fontPtr
     LD      {fontPtr+1},A      ;

     LD      A, {fontPtr+1}     ; (LSB)
     ADD     A, #font5x8.l      ; Zsumowanie obliczonego przesunięcia z
     LD      {fontPtr+1}, A     ; adresem początku tablicy czcionek

     LD      A, {fontPtr+0}     ; (MSB)
     ADC     A, #font5x8.h      ;
     LD      {fontPtr+0}, A     ;

     CLR     X
WriteCharLoop
     LD      A, ([fontPtr.w],X) ; odczyt danych z tablicy
     CALL    N3310_WriteData    ; wyświetlenie na wyświetlaczu
     INC     X
     CP      X, #5              ; czy to już cały znak?
     JRNE    WriteCharLoop      ; jeśli nie to skok do WriteCharLoop

     LD      A, #0              ;
     CALL    N3310_WriteData    ; odstęp pomiędzy znakami

     POP     X
     RET
```

List. 7. Kod makroinstrukcji *WTXT*

```
;*****************************************************************************
; Makroinstrukcja wyświetlająca tekst
;*****************************************************************************
WTXT MACRO  pstr
     LOCAL  wt1,wt2
     CLR    X               ; wyzerowanie rejestru x
wt1
     LD     A, (pstr,X)     ; załadowanie do rejestru A znaku spod adresu
będącego sumą pstr i rejestru X
     JREQ   wt2             ; jeśli odczytany znak jest zerem (koniec napi-
su) to skok do wt2
     INC    X               ; inkrementacja rejestru X (przygotowanie do po-
brania następnego znaku)
     CALL   N3310_WriteChar ; wyświetlenie znaku na wyświetlaczu
     JRA    wt1             ; skok do wt1
wt2
     MEND
```

List. 8. Kod procedury wyświetlającej bitmapę

```
;*****************************************************************************
; Procedura wyświetlająca bitmapę
;*****************************************************************************
.N3310_Bitmap
     LD      {fontPtr+0}, X  ; przepisanie adresu bitmapy do wskaźnika
     LD      {fontPtr+1}, Y  ;

     CLR     X               ;
     CLR     Y               ;
     CALL    N3310_GoTo      ; ustawienie współrzędnych ekranu X = 0 Y = 0
     LD      Y, #6           ; liczba wierszy
BitmapLoop1
     LD      X, #84          ; liczba kolumn
BitmapLoop2
     LD      A,[fontPtr.w]   ; odczytanie bajtu z tablicy
     CALL    N3310_WriteData ; wyświetlenie na wyświetlaczu

     LD      A, {fontPtr+1}  ;
     ADD     A, #1           ; inkrementacja młodszego bajtu wskaźnika
     LD      {fontPtr+1}, A  ;

     LD      A, {fontPtr+0}  ;
     ADC     A, #0           ; inkrementacja starszego bajtu wskaźnika
     LD      {fontPtr+0}, A  ;

     DEC     X               ; X-- i sprawdzenie czy to ostatnia kolumna?
     JRNE    BitmapLoop2     ; jeśli nie to skok do BitmapLoop2

     DEC     Y               ; Y-- i sprawdzenie  czy to ostatni wiersz?
     JRNE    BitmapLoop1     ; jeśli nie to skok do BitmapLoop1
     RET
```

List. 9. Kod programu demonstracyjnego

```
st7/
;*****************************************************************************
; LITEcomp - aplikacje
; Biblioteka procedur obsługi wyświetlacza NOKIA 3310
; Plik : main.asm
; Autor : Radosław Kwiecień, radoslaw.kwiecien@ep.com.pl
;*****************************************************************************

.    NOLIST
     #INCLUDE  'st7flite19.inc'
     #INCLUDE  'n3310lcd.inc'
     .LIST

     extern    obrazek.w
     extern    obrazek1.w

;*****************************************************************************
; obszar pamięci danych RAM
;*****************************************************************************
     BYTES
     SEGMENT   'ram0'
fontPtr         DS.W     1

;*****************************************************************************
; obszar pamięci programu
;*****************************************************************************
     WORDS
     SEGMENT 'rom'

tekst1
     STRING  ' www.ep.com.pl'
     STRING  '   LITEcomp   '
     STRING  '  aplikacje   '
     STRING  '  NOKIA 3310  '
     STRING  '     LCD      '
     STRING  '  ST7FLITE19  ',0

;*****************************************************************************
; Początek programu głównego
;*****************************************************************************
._reset
     CALL    N3310_Init     ; inicjalizacja

mainLoop
     LD      X, #obrazek.h
     LD      Y, #obrazek.l
     CALL    N3310_Bitmap   ; wyświetlenie bitmapy

     CALL    waitTimer      ; oczekiwanie ok 4s

     LD      X, #obrazek1.h
     LD      Y, #obrazek1.l
     CALL    N3310_Bitmap   ; wyświetlenie bitmapy

     CALL    waitTimer      ; oczekiwanie ok 4s

     LD      X, #0
     LD      Y, #0
     CALL    N3310_GoTo     ; ustawienie współrzędnych
     WTXT    tekst1         ; wyświetlenie napisu

     CALL    waitTimer      ; oczekiwanie ok 4s

     JRA     mainLoop       ; skok na początek pętli głównej

;*****************************************************************************
;
;*****************************************************************************
waitTimer
     LD      A, #%00001000  ; włączenie timera F=1KHz
     LD      ATCSR, A
     BTJF    ATCSR, #2, *   ; oczekiwanie na przepełnienie
     CLR     ATCSR
     RET

     END
;*****************************************************************************
; Koniec pliku main.asm
;*****************************************************************************
```

pliku *.bmp oraz lokalizacji wyjściowego pliku *.asm. Wygenerowany przez program plik jest gotowy do natychmiastowego dołączenia do projektu.

Program demonstracyjny

Program demonstrujący działanie przedstawionych w artykule procedur wyświetla w pętli nieskończonej naprzemiennie bitmapę oraz tekst. Wyjaśnienia wymaga sposób deklaracji tekstu. Pomimo, iż zapis może sugerować, że tekst jest podzielony na sześć wierszy, w rzeczywistości jest to jeden ciągły tekst kończący się bajtem o wartości zero. Podział na wiersze został zastosowany tylko w celu odwzorowania organizacji ekranu wyświetlacza, natomiast tak naprawdę jest to jeden ciągły napis o długości 84 znaków, którym jest zapełniany cały ekran. W związku z tym, że po osiągnięciu maksymalnej wartości adresu kolumn adres wierszy jest automatycznie inkrementowany, w przypadku ciągłego zapisu do pamięci obrazu nie jest wymagane ustawianie adresu kolejnych wierszy przez program. W przypadku wyświetlania krótszych napisów konieczne jest ich zakończenie bajtem o wartości zero oraz ustawienie odpowiednich współrzędnych ekranowych. W przypadku próby wyświetlenia napisu, który nie został zakończony zerem, program będzie zapisywał do wyświetlacza dane tak długo, aż nie natrafi na bajt o wartości zero. Objawi się to oczywiście wyświetleniem „śmieci" bądź też kolejnych umieszczonych w pamięci napisów. Kod programu przedstawiony jest na **list. 9.**

Pliki z kodem źródłowym, program demonstracyjny jak również program konwertera plików BMP2DC.exe można pobrać ze strony *www.ep.com.pl*, są one również zamieszczone na płycie CD–EP12/2007B.

Radosław Kwiecień, EP
radoslaw.kwiecien@ep.com.pl