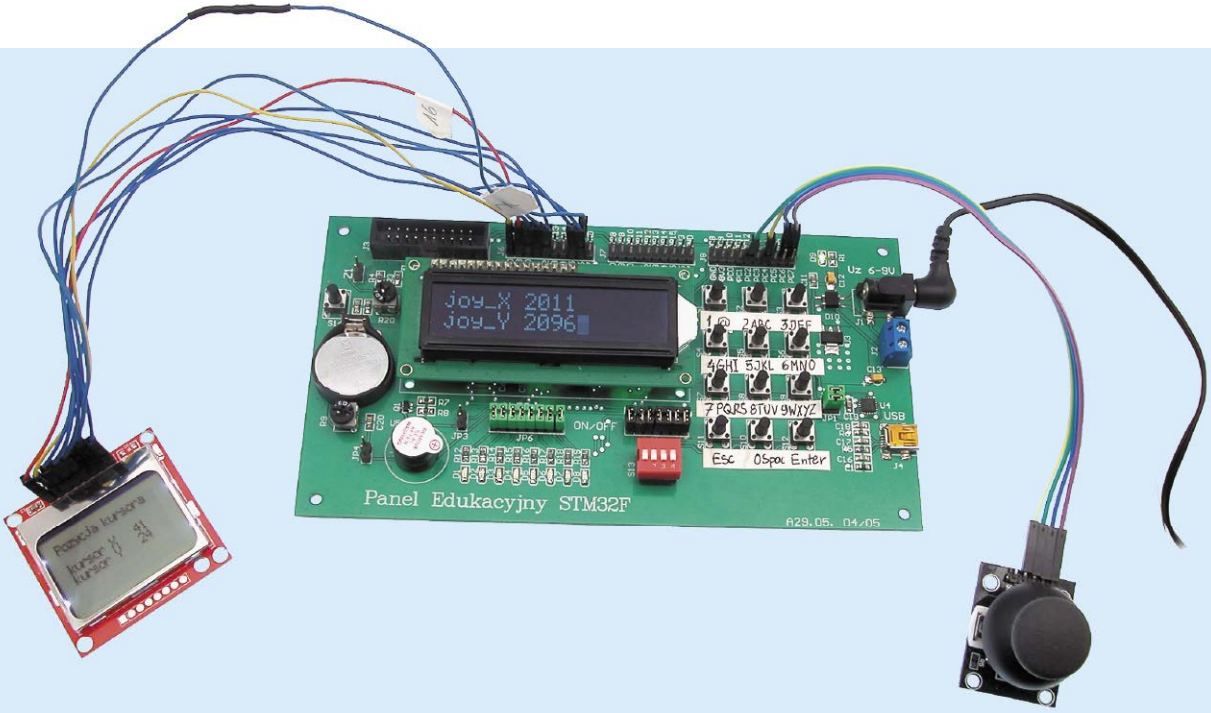


Dołączenie do STM32 manipulatora analogowego i wyświetlacza graficznego

W artykule przedstawiamy sposób podłączenia kolejnych urządzeń peryferyjnych do kontrolera STM32. Tym razem przykłady prostych procedur umożliwiających wykorzystanie manipulatora (joysticka) analogowego i wyświetlacza graficznego ze sterownikiem PCD8544.

Wyświetlacze ze sterownikiem PCD8544 były stosowane w starych modelach telefonów komórkowych Nokii. Obecnie łatwo dostępne i relatywnie tanie pozwolą poszerzyć możliwości urządzenia z STM-em.

Manipulator (joystick) analogowy

Manipulator może być wykorzystany jako wygodny w użyciu interfejs sterujący lub wprowadzający dane. Element użyty w przykładach nosi nazwę handlową „joystick PS2". Jego schemat elektryczny pokazano na **rysunku 1**. Niewielki manipulator ma wymiary 35 mm×27 mm×32 mm. Głównymi elementami są dwa potencjometry zamontowane na płytce drukowanej pod kątem prostym. Oski potencjometrów połączone są z pionowym drążkiem za pomocą przegubu. Drążek ma możliwość wychylenia w płaszczyźnie X-Y a jego ruch przenoszony jest na oski potencjometrów. W zależności od położenia drążka napięcie na suwakach potencjometrów zmienia się w zakresie od 0 V do napięcia zasilania (na schemacie zaznaczone jako +5 V). Napięcie zasilania może mieć dowolną wartość a w układach z kontrolerem STM32 nie powinno przekraczać 3,3 V. Dodatkowo,

Tabela 1. Sygnały wyprowadzone na 2 złączach 8 stykowych

Styk	Nazwa	Funkcja
1	VCC	Napięcie zasilania od 2,7V do 3,3V
2	GND	Masa
3	SCE	(Chip Enable) odblokowanie dostępu do wyświetlacza, poziom aktywny niski (LOW)
4	RES	(Reset) zerowanie wyświetlacza, poziom aktywny niski (LOW)
5	D/C	Określenie rodzaju przesyłanych danych: (HIGH) – przesyłana treść do wyświetlenia na wyświetlaczu (LOW) – przesyłane rozkazy sterujące
6	SDIN	Linia danych przesyłanych szeregowo bit po bicie
7	SCLK	Linia zegara taktującego przesyłane bity danych
8	LED	Napięcie zasilania diod podświetlenia od +2,7 V do +3,3 V

Tabela 2. Rozkazy sterownika PCD8544

Rozkaz	Bajt rozkazu								Opis
	b7	B6	B5	B4	B3	B2	B1	B0	
NOP	0	0	0	0	0	0	0	0	Nic nie rób
Function Set	0	0	1	0	0	PD	V	H	Tryb obniżonego poboru energii (PD), adresowania (V) i komendy rozszerzone (H)
H=0 podstawowy zestaw komend									
Display Control	0	0	0	0	1	D	0	E	Konfiguracja wyświetlacza
Set Y address	0	1	0	0	0	Y2	Y1	Y0	Ustawienie licznika wierszy 0…5
Set X address	1	X6	X5	X4	X3	X2	X1	X0	Ustawienie licznika kolumn 0…83
H=1 rozszerzony zestaw komend									
Temperat. Control	0	0	0	0	0	1	TC1	TC0	Współczynnik temperaturowy
Bias System	0	0	0	1	0	BS2	BS1	BS0	Ustawienie BIAS
Set Vop	1	Vp6	Vp5	Vp4	Vp3	Vp2	Vp1	Vp0	Napięcie zasilania matrycy (kontrast)

naciśnięcie drążka powoduje zadziałanie przycisku SW zwierając do masy wyprowadzenie J-1.

Wyświetlacz graficzny

Jako wyświetlacz wybrany został moduł z wyświetlaczem LCD stosowanym kiedyś w telefonach Nokia 5110. Na **rysunku 2** pokazano jego wygląd i wymiary. Monochromatyczny wyświetlacz ma rozdzielczość 84×48 piksele. Może być zasilany napięciem 3,3 V. W module można skorzystać z podświetlenia matrycy 4 białymi diodami LED. Pracą wyświetlacza steruje zamontowany na płytce modułu układ PCD8544 lub jego odpowiednik. W **tabeli 1** umieszczono zestawienie sygnałów wyprowadzonych na 2 złączach 8 stykowych.

Format transmisji do wyświetlacza

Przesyłanie danych i rozkazów do wyświetlacza odbywa się w formacie podobnym do SPI. Wykorzystywane są trzy sygnały:

1. SDIN linia szeregowa, po której przesyłane są dane.
2. SCLK linia zegara, którego zbocza narastające zatrzaskują kolejne przesyłane bity danych.
3. SCE – linia uaktywniająca dostęp do wyświetlacza, gdy jej poziom będzie niski.

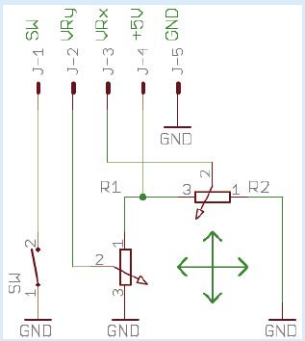
Na **rysunku 3** przedstawiono przebiegi czasowe podczas transmisji 1 bajtu do wyświetlacza. Do wyświetlacza można wysyłać większą liczbę danych przy sygnale SCE stale utrzymywanym na poziomie niskim.

Organizacja pamięci obrazu

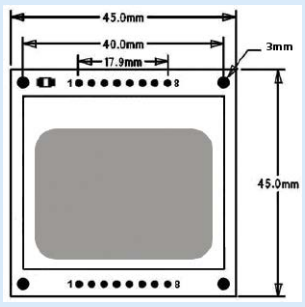
W module wyświetlacza z PCD8544 każdy piksel obrazu jest zapalany lub gaszony stanem pojedynczego bitu pamięci obrazu. Przy rozdzielczości matrycy 84×48 daje to 504 bajty. Cała pamięć zorganizowana jest w 6 bankach po 84 bajty każdy (**rysunek 4**). Po wpisaniu do pierwszego bajtu banku 0 wartości 0xFF w lewym górnym rogu wyświetlacza pojawi się pionowa linia o długości 8 pikseli.

Za pomocą rozkazów sterujących można określić miejsce w pamięci obrazu, do którego ma zostać wpisany bajt danych. Licznik wierszy można ustawić w przedziale od 0 do 5. Licznik kolumn można ustawić w przedziale od 0 do 83.

Po zapisaniu do pamięci obrazu danej, liczniki kolumn i wierszy zostaną automatycznie zwiększone. Zależnie od ustawionego wcześniej trybu pracy wyświetlacza inkrementacja liczników przebiegnie odmiennie. W trybie *poziomym* (horyzontalnym) zwiększany zostanie licznik kolumn. Po osiągnięciu wartości 83 przed kolejnym wpisem do pamięci obrazu zostanie wyzerowany natomiast licznik rzędów zwiększy się o 1 (**rysunek 5**). W trybie *pionowym* (wertykalnym) zwiększony zostanie licznik rzędów. Po osiągnięciu wartości 5 przed kolejnym wpisem do pamięci obrazu zostanie wyzerowany natomiast licznik kolumn zwiększy się o 1 (**rysunek 6**).



Rysunek 1. Schemat ideowy joysticka PS2



Rysunek 2. Wygląd wyświetlacza Nokii 5110

Tabela 3. Ustawienia bitów konfiguracyjnych

Oznaczenie	Gdy bit =0	Gdy bit =1
PD	Układ aktywny	Tryb obniżonego poboru mocy
V	Adresowanie poziome	Adresowanie pionowe
H	Podstawowy zestaw instrukcji	Rozszerzony zestaw instrukcji
D E 0 0 0 1 1 0 1 1	 Wyświetlacz wygaszony Tryb normalny Wszystkie segmenty „zaświecone” Wyświetlanie w inwersji	
TC1 TC0 0 0 0 1 1 0 1 1	 VLCD współczynnik temperaturowy 0 VLCD współczynnik temperaturowy 1 VLCD współczynnik temperaturowy 2 (standardowy) VLCD współczynnik temperaturowy 3	
Użyteczny zakres ustawiania napięcia Vp (kontrastu) to 30 – 90.		

Tabela 4. Wykaz połączeń pomiędzy wyświetlaczem, manipulatorem a Panelem Edukacyjnym

Interfejs		Panel Edukacyjny
Styk	Funkcja	Gniazdo
Manipulator (joystik)		
J-1	SW	Niepodłączone
J-2	VRy	J8-PC4
J-3	VRx	J8-PC2
J-4	+5V (zasilanie)	J8-3V3
J-5	GND	J8-GND
Moduł wyświetlacza z PCD8544		
1	VCC	J6-3V3
2	GND	J6-GND
3	SCE	J6-PA6
4	RES	J6-PA5
5	D/C	J6-PA2
6	SDIN	J6-PA1
7	SCLK	J6-PA0
8	LED	J6-3V3

Listing 1. Konfigurowanie zegara oraz inicjowanie portów

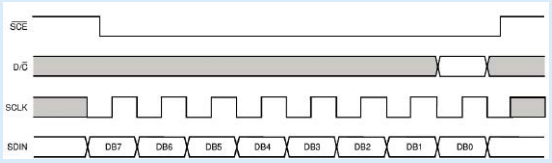
```
//procedura podłączenia taktowania do przetworników i portów
void RCC_Configuration(void)
{
    /* ADCCLK = PCLK2/4 */
    RCC_ADCCLKConfig(RCC_PCLK2_Div4);
    /* Enable peripheral clocks */
    /* Enable DMA1 and DMA2 clocks */
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1 | RCC_AHBPeriph_DMA2, ENABLE);
    /* Enable ADC1, ADC3 and GPIOC clocks */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1 | RCC_APB2Periph_ADC3 | RCC_APB2Periph_GPIOC, ENABLE);
}

//inicjowanie portów kanału ADC12 i ADC14
void GPIO_Configuration(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    /* Configure PC.02, PC.04 (ADC Channel12, ADC Channel14) as analog inputs */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_4;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
}
```

Listing 2. Inicjowanie przetworników i kanału DMA

```
//konfigurowanie DMA i przetworników ADC
void DMA_ADC_Konfiguracja(void)
{
ADC_InitTypeDef ADC_InitStructure;
DMA_InitTypeDef DMA_InitStructure;
    /* DMA1 channel1 configuration */
    DMA_DeInit(DMA1_Channel1);
    DMA_InitStructure.DMA_PeripheralBaseAddr = ADC1_DR_Address;
    DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)&ADC1ConvertedValue;
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
    DMA_InitStructure.DMA_BufferSize = 1;
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Disable;
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
    DMA_InitStructure.DMA_Priority = DMA_Priority_High;
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
    DMA_Init(DMA1_Channel1, &DMA_InitStructure);
    /* Enable DMA1 channel1 */
    DMA_Cmd(DMA1_Channel1, ENABLE);
    /* DMA2 channel5 configuration */
    DMA_DeInit(DMA2_Channel5);
    DMA_InitStructure.DMA_PeripheralBaseAddr = ADC3_DR_Address;
    DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)&ADC3ConvertedValue;
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
    DMA_InitStructure.DMA_BufferSize = 1;
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Disable;
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
    DMA_InitStructure.DMA_Priority = DMA_Priority_High;
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
    DMA_Init(DMA2_Channel5, &DMA_InitStructure);
    /* Enable DMA2 channel5 */
    DMA_Cmd(DMA2_Channel5, ENABLE);
    /* ADC1 configuration */
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_NbrOfChannel = 1;
    ADC_Init(ADC1, &ADC_InitStructure);
    /* ADC1 regular channels configuration */
    ADC_RegularChannelConfig(ADC1, ADC_Channel_14, 1, ADC_SampleTime_28Cycles5);
    /* Enable ADC1 DMA */
    ADC_DMACmd(ADC1, ENABLE);
    /* ADC3 configuration */
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_NbrOfChannel = 1;
    ADC_Init(ADC3, &ADC_InitStructure);
    /* ADC3 regular channel14 configuration */
    ADC_RegularChannelConfig(ADC3, ADC_Channel_12, 1, ADC_SampleTime_28Cycles5);
    /* Enable ADC3 DMA */
    ADC_DMACmd(ADC3, ENABLE);
    /* Enable ADC1 */
    ADC_Cmd(ADC1, ENABLE);
    /* Enable ADC1 reset calibration register */
    ADC_ResetCalibration(ADC1);
    /* Check the end of ADC1 reset calibration register */
    while(ADC_GetResetCalibrationStatus(ADC1));
    /* Start ADC1 calibration */
    ADC_StartCalibration(ADC1);
    /* Check the end of ADC1 calibration */
    while(ADC_GetCalibrationStatus(ADC1));
    /* Enable ADC3 */
    ADC_Cmd(ADC3, ENABLE);
    /* Enable ADC3 reset calibration register */
    ADC_ResetCalibration(ADC3);
    /* Check the end of ADC3 reset calibration register */
    while(ADC_GetResetCalibrationStatus(ADC3));
```

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 82218, pass: aagt5gj6



Rysunek 3. Przebiegi czasowe podczas transmisji 8 bitów do wyświetlacza

Wyświetlacz rozkazy sterujące

Oprócz rozkazów ustawiających liczniki kolumn i wierszy inne wpływają na tryb pracy, ustawienie kontrastu wyświetlacza itd. Przesyłane dane traktowane są jak komendy sterujące, gdy linia D/C jest wyzerowana. Rozkazy sterownika PCD8544 zebrane zostały w **tabeli 2**, natomiast ustawienia bitów konfiguracyjnych w **tabeli 3**.

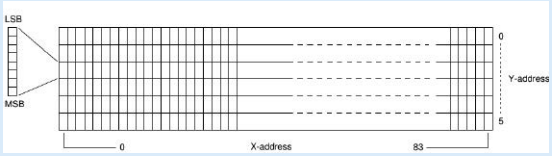
Przogram przykładowy

Przykładowy program ma działać tak by zamieniać wychylenie manipulatora na liczby odpowiednio dla osi X (poziomej) i Y (pionowej). Chwilowa pozycja wskazywana przez joystik wyświetlana będzie na wyświetlaczu. Ze względu na łatwość dołączenia wyświetlacza i manipulatora do testów użyty został opisywany we wcześniejszych numerach EP Panel Edukacyjny. W **tabeli 4** podano wykaz połączeń pomiędzy wyświetlaczem, manipulatorem a Panelem. Dodatkowo, w gnieździe J5 Panelu można zamocować wyświetlacz LCD 2×16.

Przykładowy program został przygotowany dla środowiska CooCox.

Procedury do obsługi joystika analogowego

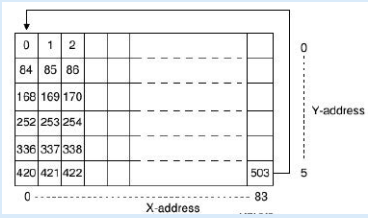
Ponieważ joystick dostarcza na wyjściach poziomy napięć proporcjonalne do ustawionej pozycji, w systemie z mikrokontrolerem należy je najpierw przekształcić do postaci cyfrowej. STM32F103RC zamontowany na Panelu wyposażony jest w konwerter analogowo-cyfrowy o rozdzielczości 12 bitów a więc zupełnie wystarczającej.



Rysunek 4. Organizacja pamięci obrazu

Przedstawiona dalej procedura została oparta na przykładzie z biblioteki *stm3210x_stdpheriph_lib*. Do obsługi konwersji napięć z dwu kanałów X i Y joystika wykorzystano interfejsy ADC1 i ADC3 przetwornika oraz DMA. Dzięki takiemu rozwiązaniu po jednokrotnej inicjacji konwersje przebiegają całkowicie automatycznie wraz z odświeżaniem wyniku we wskazanych rejestrach programu.

Zgodnie z tym, co zostało podane w tabeli, napięcie z potencjometru X manipulatora podawane jest



Rysunek 5. Wpisywanie bajtów do pamięci obrazu w trybie horyzontalnym

Listing 2. c.d.

```
    /* Start ADC3 calibration */
    ADC_StartCalibration(ADC3);
    /* Check the end of ADC3 calibration */
    while(ADC_GetCalibrationStatus(ADC3));
    /* Start ADC1 Software Conversion */
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);
    /* Start ADC3 Software Conversion */
    ADC_SoftwareStartConvCmd(ADC3, ENABLE);
}
```

Listing 3. Wstępne ustawienia linii I/O

```
GPIO_PCD8544_High(PCD8544_SCLK_WY);
GPIO_PCD8544_Low(PCD8544_DN_WY);
GPIO_PCD8544_Low(PCD8544_DnC_WY);
GPIO_PCD8544_High(PCD8544_RST_WY);
GPIO_PCD8544_High(PCD8544_SCE_WY);

//ustawienie linii PCD8544
void GPIO_PCD8544_High(PCD8544_IO_TypeDef Linia)
{
    PCD8544_IO_PORT[Linia]->BSRR =PCD8544_IO_PIN[Linia];
}

//ustawienie stanu niskiego na linii PCD8544
void GPIO_PCD8544_Low(PCD8544_IO_TypeDef Linia)
{
    PCD8544_IO_PORT[Linia]->BRR =PCD8544_IO_PIN[Linia];
}
```

Listing 4. Procedura programowej transmisji przez SPI do PCD8544

```
void WriteSPI_PCD8544(unsigned char dana)
{
char x;
    GPIO_PCD8544_Low(PCD8544_SCLK_WY);
    Delay_us(10);
    for(x=0;x<8;x++)
    {
        if((dana &0x80) !=0x0) GPIO_PCD8544_High(PCD8544_DN_WY);  //bit jest 1
        else GPIO_PCD8544_Low(PCD8544_DN_WY);  //bit jest 0
        Delay_us(10);
        GPIO_PCD8544_High(PCD8544_SCLK_WY);  //zbocze _/ impulsu SPI
        Delay_us(10);
        GPIO_PCD8544_Low(PCD8544_SCLK_WY);  //zbocze \_ impulsu SPI
        dana =dana<<1;
    }
}
```

Listing 5. Procedura inicjacji sterownika PCD8544

```
void Inicjacja_PCD8544(void)
{
    int x;
    GPIO_PCD8544_Inicjacja();
    GPIO_PCD8544_Low(PCD8544_RST_WY);
    Delay_ms(100);
    GPIO_PCD8544_High(PCD8544_RST_WY);
    WriteCmd_PCD8544(0x21);//komendy rozszerzone
    WriteCmd_PCD8544(0x05);//komenda „przelaczajaca” sterownik w tryb pracy zgodny PCD8544
    WriteCmd_PCD8544(0xb8);//ustawienie Vop
    WriteCmd_PCD8544(0x06);//korekcja temperatury dla PCD8544
    WriteCmd_PCD8544(0x14);//współczynnik multipleksowania
    WriteCmd_PCD8544(0x20);//komendy standardowe - adresowanie poziome
    WriteCmd_PCD8544(0x01);
    WriteCmd_PCD8544(0x0c);//tryb wyświetlania Standard Mode
    WriteCmd_PCD8544(0x40);//zerowanie licznika wierszy
    WriteCmd_PCD8544(0x80);//zerowanie licznika kolumn
    for(x=0;x<504;x++)
    WriteData_PCD8544(0x00);//zerowanie pamięci obrazu
}
```

Listing 6. Procedura ustawienia pozycji kursora
```
void LcdUstawKursor(uint8_t poz_X, uint8_t poz_Y)
{
unsigned char dana;

    //Ustawienie licznika kolumn
    dana =0x80 | (unsigned char) poz_X;
    WriteCmd_PCD8544(dana);
    //Ustawienie licznika wierszy
    dana =0x40 | (unsigned char) poz_Y;
    WriteCmd_PCD8544(dana);
}
```

na wejście portu PC2, zaś napięcie z potencjometru Y na wejście PC4. Oba porty podłączone są wewnętrznie w kontrolerze do kanałów 12 i 14 przetwornika A/C.

Na początku należy zadeklarować adresy bazowe interfejsów ADC1 i ADC3 oraz zmiennych, w których po konwersji będą umieszczane wyniki dla kanału X i Y:
```
#define ADC1_DR_Address
((uint32_t)0x4001244C)
#define ADC3_DR_Address
((uint32_t)0x40013C4C)
```

Listing 7. Konwersja łańcucha znaków na ich reprezentację graficzną przesyłaną do wyświetlenia
```
//procedura wyświetlenia pozycjonowanego napisu
void LcdNapisPoz(uint8_t poz_X, uint8_t poz_Y, char *p_napis)
{
    uint8_t ile_znakow, x;
    if (poz_X >=LCD_X) return;//błąd
    if (poz_Y >=LCD_Y_WIERSZE_MAX) return;//błąd
    ile_znakow =strlen(p_napis);
    if (ile_znakow >((LCD_X/5)*(LCD_Y/8))) return;//błąd
    //wyswietlenie nowego napisu
    LcdUstawKursor(poz_X, poz_Y);
    LcdString(p_napis);
}

//procedura wyświetlająca łańcuch znaków
void LcdString(char *p_napis)
{
    while (*p_napis)
    {
        LcdCharacter(*p_napis++);
    }
}

//procedura wyświetlająca pojedynczy znak
void LcdCharacter(char znak)
{
int index;

    for (index = 0; index < 5; index++)
    {
        WriteData_PCD8544(ASCII[znak - 0x20][index]);//ASCII -tabela z matrycą znaków 5x8 pikseli
    }
}

//tabela z matrycą znaków
static const uint8_t ASCII[][5] =
{
 {0x00, 0x00, 0x00, 0x00, 0x00} // 20
,{0x00, 0x00, 0x5f, 0x00, 0x00} // 21 !
,{0x00, 0x07, 0x00, 0x07, 0x00} // 22 „
,(pominięty fragment tabeli)
,{0x44, 0x64, 0x54, 0x4c, 0x44} // 7a z
,{0x00, 0x08, 0x36, 0x41, 0x00} // 7b {
,{0x00, 0x00, 0x7f, 0x00, 0x00} // 7c |
,{0x00, 0x41, 0x36, 0x08, 0x00} // 7d }
,{0x10, 0x08, 0x08, 0x10, 0x08} // 7e ‹
,{0x78, 0x46, 0x41, 0x46, 0x78} // 7f ›
};
```

Listing 8. Pętla główna programu
```
#define LCD_X     84c
#define LCD_Y     48

typedef struct NAPIS_BAK{
    uint8_t poz_X;
    uint8_t poz_Y;
    uint8_t ile_znakow;
}napis_bak_typedef;

while(1)
{
    //wyswietlenie napisów na wyswietlaczu graficznym 84x48 z PCD8544
    //wymazanie napisu1 z poprzednią pozycją X joystika
    LcdUstawKursor(napis1_bak.poz_X, napis1_bak.poz_Y);
    for (x=0; x<napis1_bak.ile_znakow; x++)
    {
        LcdCharacter(, ,);
    }
    //wyswietlenie nowego napisu1 z aktualną pozycją X joystika
    poz_kursor_X =(ADC1ConvertedValue *LCD_X) /(4095+1);
    napis1_bak.ile_znakow =snprintf(&bufor_disp[0], 20, „kursor X %4d”, poz_kursor_X);
    LcdNapisPoz(napis1_bak.poz_X, napis1_bak.poz_Y, &bufor_disp[0]);
    //wymazanie napisu2 z poprzednią pozycją Y joystika
    LcdUstawKursor(napis2_bak.poz_X, napis2_bak.poz_Y);
    for (x=0; x<napis2_bak.ile_znakow; x++)
    {
        LcdCharacter(, ,);
    }
    //wyswietlenie nowego napisu2 z aktualną pozycją Y joystika
    poz_kursor_Y =(ADC3ConvertedValue *LCD_Y) /(4095+1);
    napis2_bak.ile_znakow =snprintf(&bufor_disp[0], 20, „kursor Y %4d”, poz_kursor_Y);
    LcdNapisPoz(napis2_bak.poz_X, napis2_bak.poz_Y, &bufor_disp[0]);
    Delay_ms(200);
}
```

```
uint16_t ADC1ConvertedValue,
ADC3ConvertedValue;
```

Potem do portów i DMA należy podłączyć wewnętrzny zegar i zainicjować porty w analogowym trybie pracy (**listing 1**). Kolejna procedura inicjuje interfejsy ADC1, ADC3, kanały DMA oraz uruchamia automatyczną konwersję. Zamieszczono ją na **listingu 2**. Od tego momentu zmienne *ADC1ConvertedValue, ADC3ConvertedValue* będą automatycznie odświeżane po zakończeniu kolejnej konwersji. Ponieważ przetwornik pracuje z 12 bitową rozdzielczością wyniki konwersji umieszczane w zmiennych będą mogły przyjąć wartość od 0 do 4095.

Procedury obsługi modułu wyświetlacza

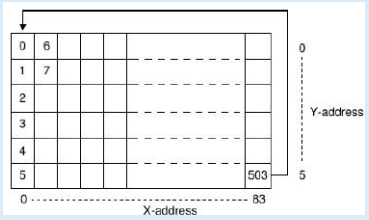
Najpierw należy zadeklarować porty kontrolera, które będą sterować odpowiednimi liniami wyświetlacza. Połączenia pomiędzy portami Panelu i liniami wyświetlacza podane zostały w tabeli. W procedurach zamiast nazwą portu wygodnie jest posłużyć się etykietą:

- PCD8544_SCLK_WY – PA0, sygnał SCLK wyświetlacza.
- PCD8544_DN_WY – PA1, sygnał DN wyświetlacza.
- PCD8544_DnC_WY – PA2, sygnał D/N wyświetlacza.
- PCD8544_RST_WY – PA5, sygnał RESET wyświetlacza.
- PCD8544_SCE_WY – PA6, sygnał SCE wyświetlacza.

Najpierw znanym sposobem należy podłączyć sygnał zegarowy do portów (w tym przypadku PA) i zainicjować je w trybie wyjścia z podciąganiem *GPIO_Mode_Out_PP*. Potem wstępnie ustawiamy odpowiednie poziomy na poszczególnych portach – **listing 3**.

Podstawowa procedura transmisji pojedynczego bajtu do modułu wyświetlacza może być taka, jak na **listingu 4**. Na tej procedurze oparto procedury transmisji danych do pamięci obrazu oraz wysyłania rozkazów sterujących. Wyglądają one następująco:

```
//procedura zapisu danej obrazu do ster-
ownika PCD8544
void WriteData_PCD8544(unsigned char dana)
{
        GPIO_PCD8544_Low(PCD8544_SCE_WY);
        GPIO_PCD8544_High(PCD8544_DnC_
WY);//wysyłanie danych obrazu
        WriteSPI_PCD8544(dana);
        GPIO_PCD8544_
High(PCD8544_SCE_WY);
}
//procedura zapisu
rozkazu do sterownika
PCD8544
void WriteCmd_
PCD8544(unsigned char
dana)
{
        GPIO_PCD8544_
Low(PCD8544_SCE_WY);
        GPIO_PCD8544_
Low(PCD8544_DnC_
WY);//wysyłanie komend
        WriteSPI_PCD8544(dana);
        GPIO_PCD8544_High(PCD8544_SCE_WY);
}
```



Rysunek 6. Wpisywanie bajtów do pamięci obrazu w trybie wertykalnym

Po opisanym wcześniej zainicjowaniu portów kontrolera należy wywołać procedurę inicjacji samego wyświetlacza. Sprowadza się ona do przesłania do wysłania sekwencji komend a na koniec do wyczyszczenia pamięci obrazu. Całą sekwencję zaprezentowano na **listingu 5**. Kolejna procedura służy do ustawienia licznika wierszy i kolumn na pożądaną wartość. Ustawienia te wyznaczą na wyświetlaczu pozycję, od której rozpocznie się wyświetlanie nowego tekstu (**listing 6**). Następne procedury konwertują łańcuch znaków na ich graficzną reprezentację przesyłaną do wyświetlenia. Kształt każdego znaku zapisany jest w tabeli-matrycy znaków 5*8 pikseli. Zamieszczono je na **listingu 7**.

Główna pętla programu

W głównej, nieskończonej pętli programu (**listing 8**) następuje:

- Przekształcenie otrzymanych po konwersji wartości do zakresu odpowiadającego rozdzielczości poziomej i pionowej.
- Wymazanie poprzednich napisów na wyświetlaczu.
- Wyświetlenie informacji o nowej pozycji joysticka.

Program demonstracyjny został przygotowany jako pakiet środowiska programistycznego CooCox.

Ryszard Szymaniak, EP