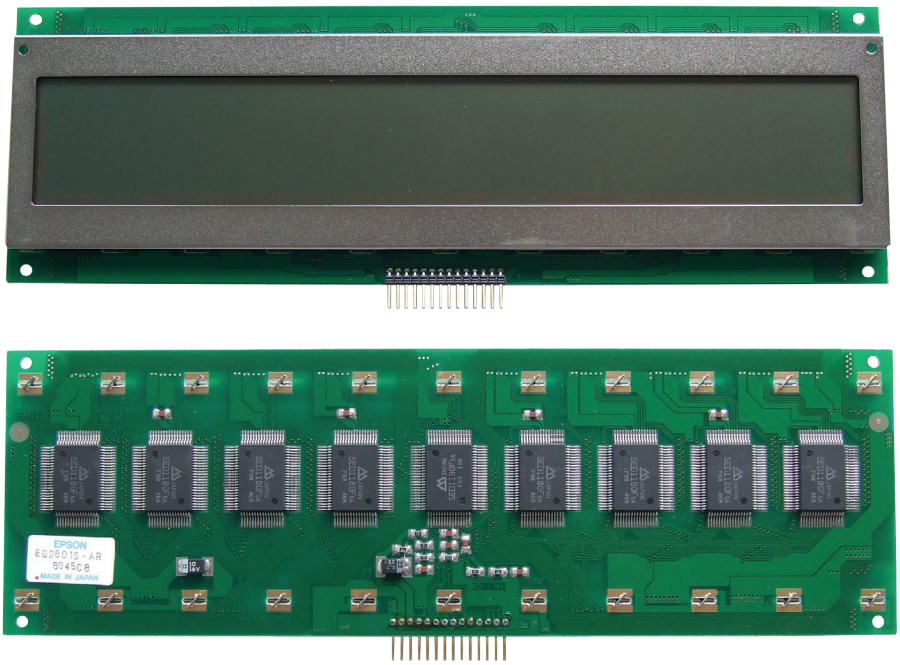
Sterownik wyświetlacza graficznego LCD typu EG2801S

Wyświetlacze graficzne LCD bez wbudowanego układu kontrolera mają dość złożony sposób sterowania, zmuszającym zazwyczaj konstruktorów do sięgania po specjalizowane układy scalone sterowników LCD. Jak jednak pokazuje praktyka, obsługa takich wyświetlaczy jest także możliwa bezpośrednio przez mikrokontroler, bez konieczności stosowania raczej trudno dostępnych kontrolerów graficznych typu S1D13xxx. Przykład takiego rozwiązania był już prezentowany na łamach Elektroniki Praktycznej. Artykuł przedstawia kolejny sposób dołączenia do mikrokontrolera typu STM32F107 graficznego wyświetlacza LCD bez wbudowanego układu kontrolera.



Fotografia 1. Wyświetlacz graficzny LCD typu EG2801S-AR firmy EPSON

Pokazany na **fotografii 1** moduł typu EG2801S-AR firmy EPSON jest monochromatycznym, graficznym wyświetlaczem LCD o rozdzielczości ekranu 512×64 piksele. Należy on do rodziny wyświetlaczy LCD, które nie są wyposażone w układ kontrolera, a jedynie w drajwery wierszy i kolumn. Zwykle wyświetlacze tego typu mają wbudowany jeden układ scalony sterownika wierszy i od jednego do kilkunastu połączonych szeregowo scalonych układów sterowników kolumn. Przykładowe modele wyświetlaczy tego typu umieszczono na wykazie w **tabeli 1**.

Wyświetlacz EG2801S-AR jest wyposażony w jeden układ sterownika wierszy SED1190F i 8 układów sterowników kolumn SED1180F. Interesujący może być fakt, iż moduł EG4401S-AR o rozdzielczości 256×128 pikseli ma taki sam schemat elektryczny. Różnica w budowie obu wyświetlaczy polega jedynie na tym, iż w module EG2801S cały łańcuch 8 układów SED1180F steruje jedną linią w panelu LCD, natomiast w module EG4401S pierwsze 4 układy SED1180F z łańcucha sterują linią w górnej połówce ekranu, podczas gdy kolejne 4 układy

Tabela 1. Moduły graficznych wyświetlaczy LCD firmy EPSON rodziny EGxxxS o współczynniku multipleksowania 1/64

Lp	Typ	Rozmiar ekranu	Sterownik wierszy	Sterownik kolumn
1	EG2201S-AR	128 × 64 piksele	1× SED1190F	2 × SED1180F
2	EG2401S-AR	256 × 64 piksele	1× SED1190F	4 × SED1180F
3	EG2801S-AR	512 × 64 piksele	1× SED1190F	8 × SED1180F
4	EG4401S-AR	256 × 128 pikseli	1× SED1190F	8 × SED1180F
5	EG4801S-AR	512 × 128 pikseli	1× SED1190F	16 × SED1180F

Tabela2. Rozmieszczenie wyprowadzeń wyświetlacza EG2801S i ich funkcje

Pin nr	Sygnał	Nazwa	Opis
1	VDD	Power supply	Napięcie zasilania, $+5V_{DC}$ ±5%
2	VSS	Ground	Masa
3	VLCD	Power supply for LCD panel	Regulacja kontrastu LCD, $-17V_{DCmin}$
4	LP	Latch pulse	Zapis zawartości linii obrazu (aktywne zbocze opadające)
5	FR	Frame pulse	Sygnał AC sterowania panelem LCD
6	YDIS	Display control	Sterowanie modułem LCD (L – moduł wyłączony, H – moduł załączony)
7	YSCL	Y shift clock	Zegar wierszy (aktywne zbocze opadające)
8	DIN	Synchronizing pulse	Znacznik początku pola obrazu
9	XSCL	X shift clock	Zegar danych (aktywne zbocze opadające)
10	XECL	Enable clock	Zegar wyboru drivera kolumn
11	D0	Data 0	Bit b0 słowa danych (lsb)
12	D1	Data 1	Bit b1 słowa danych
13	D2	Data 2	Bit b2 słowa danych
14	D3	Data 3	Bit b3 słowa danych (msb)

z łańcucha sterują linią w dolnej połówce ekranu. Odmianą modułu EG2801S-AR jest także wyświetlacz firmy EPSON typu ECM-A0083. W zasadzie jest to moduł EG2801S-AR z zainstalowanym podświetleniem powierzchni ekranu za pomocą folii EL.

Wszystkie sygnały sterujące układów SED1180F i SED1190F oraz linie zasilania są wyprowadzone w wyświetlaczu EG2801S-AR na 14-pinowe złącze krawędziowe GoldPin o rastrze 2,54 mm. Rozmieszczenie wyprowadzeń opisuje **tabela 2**.

Listing 1. Funkcja konfigurująca układy mikrokontrolera STM32 do obsługi wyświetlacza EG2801S

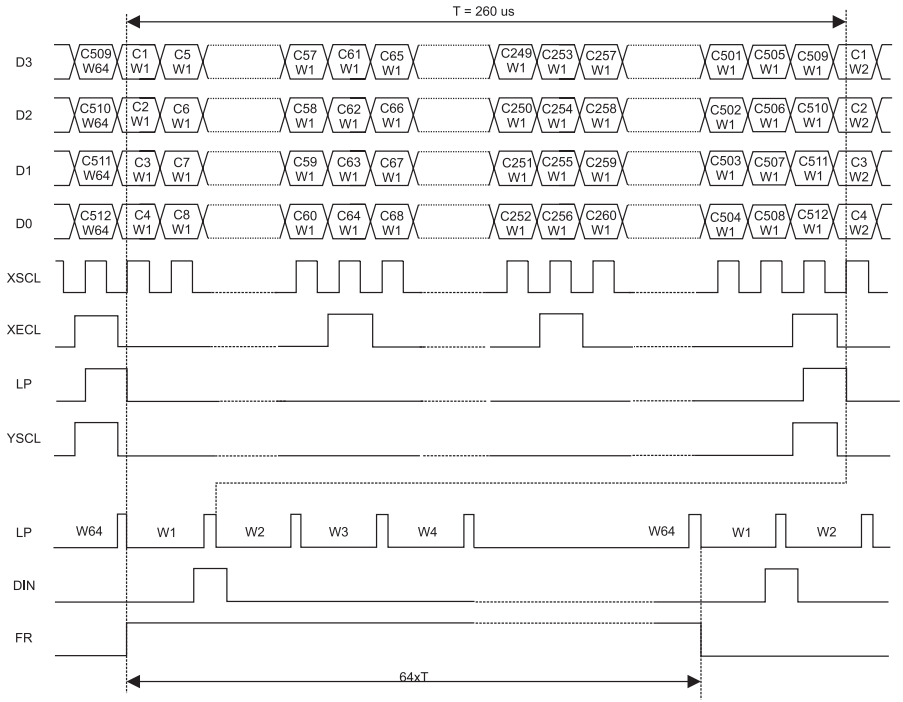
```
void LCD_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;
  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_OCInitTypeDef TIM_OCInitStructure;
  DMA_InitTypeDef DMA_InitStructure;

// ----------------------------------------------------------
// Initialization of the frame buffer
// Frame buffer contains patterns of DIN, YSCL and XECL control signals
// ----------------------------------------------------------
  LCD_InitFrameBuffer();
// ----------------------------------------------------------
// Clock enable of used peripherals
// LCD uses DMA1, TIM2, TIM4, TIM5, GPIOA, GPIOD and AFIO
// ----------------------------------------------------------
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2 | RCC_APB1Periph_TIM4 | \
                         RCC_APB1Periph_TIM5, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOD | \
                         RCC_APB2Periph_AFIO, ENABLE);

// ----------------------------------------------------------
// Configuration of LCD data lines and signal lines
// controlled from RAM. LCD DIN signal (row scan start-up pulse)
// is read-out from RAM to pin PD6. LCD YSCL signal (row scan shift
// clock) is read-out from RAM to pin PD5. LCD XECL signal (enable
// transition clock) is read-out from RAM to pin PD4. LCD data lines
// are connected to pins PD3..PD0
// ----------------------------------------------------------
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_5 | GPIO_Pin_4 \
                              | GPIO_Pin_3 | GPIO_Pin_2 | GPIO_Pin_1 | GPIO_
Pin_0;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOD, &GPIO_InitStructure);
// ----------------------------------------------------------
// Configuration of LCD X shift clock / data shift clock
// LCD data shift clock XSCL is generated on pin PD12 by channel
// 1 of timer TIM4
// ----------------------------------------------------------
// PD12 configuration as TIM4_CH1 output
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOD, &GPIO_InitStructure);
  GPIO_PinRemapConfig(GPIO_Remap_TIM4, ENABLE);
  // TIM4 time base configuration (fclk=72MHz)
  TIM_TimeBaseStructure.TIM_Period = LCD_T_XSCL-1;
  TIM_TimeBaseStructure.TIM_Prescaler = 0;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
  TIM_ARRPreloadConfig(TIM4, ENABLE);
  // TIM4 Channel_1 configuration: PWM1 Mode / 50%
  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
  TIM_OCInitStructure.TIM_Pulse = LCD_T_XSCL/2;
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OC1Init(TIM4, &TIM_OCInitStructure);
  TIM_OC1PreloadConfig(TIM4, TIM_OCPreload_Enable);
  // Clock source configuration: TIM4 is the master for timers TIM2 and TIM5
  TIM_SelectMasterSlaveMode(TIM4, TIM_MasterSlaveMode_Enable);
  TIM_SelectOutputTrigger(TIM4, TIM_TRGOSource_Update);
// ----------------------------------------------------------
// Configuration of LCD data latch pulse
// LCD data latch pulse LP is generated on pin PA0 by channel
// 1 of timer TIM5
// ----------------------------------------------------------
  // PA0 configuration as TIM5_CH1 output
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  // Clock source configuration: TIM5 is the slave of TIM4
  TIM_SelectSlaveMode(TIM5, TIM_SlaveMode_External1);
  TIM_SelectInputTrigger(TIM5, TIM_TS_ITR2);
  // TIM5 time base configuration (fclk=fTIM4)
  TIM_TimeBaseStructure.TIM_Period = LCD_COL_QTY/4 - 1;
  TIM_TimeBaseStructure.TIM_Prescaler = 0;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM5, &TIM_TimeBaseStructure);
  TIM_ARRPreloadConfig(TIM5, ENABLE);
  // TIM5 Channel1 configuration: PWM1 Mode
  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
  TIM_OCInitStructure.TIM_Pulse = LCD_COL_QTY/4 - 1;
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
  TIM_OC1Init(TIM5, &TIM_OCInitStructure);
  TIM_OC1PreloadConfig(TIM5, TIM_OCPreload_Enable);
```

Sekwencje sygnałów sterujących modułem EG2801S przedstawia **rysunek 2**. Dane do wyświetlacza są wpisywane synchronicznie w takt sygnału zegara danych XSCL. Zapis danych jest wykonywany opadającym zboczem tego sygnału. W jednym takcie zegara wpisywane są stany czterech kolejnych punktów w linii C_x, C_{x+1}, C_{x+2}, C_{x+3} (gdzie x=1, 5, 9, ..61), odpowiadające odpowiednio bitom b3, b2, b1 i b0. Ustawiony bit odpowiada zapalonemu pikselowi. Po każdym 16-tym impulsie sygnału zegara danych XSCL opadające zbocze na linii XECL aktywuje odbiór danych w kolejnym układzie sterownika kolumn SED1180F w łańcuchu, w wyniku czego następne 16 słów danych (czyli stany 64 pikseli) jest zapisywane w tym układzie. Po zapisie zawartości całej linii obrazu, co w przypadku wyświetlacza EG2801S zajmuje 128 takty zegara XSCL, opadające zbocze na linii LP zatrzaskuje w sterownikach kolumn wpisane dane. Od tego momentu rozpoczyna się ich wyświetlanie, a do rejestrów przesuwnych sterowników kolumn SED1180F wpisywana jest zawartość kolejnej linii obrazu, po czym cały cykl się powtarza. Obraz jest wyświetlany dynamicznie, linia po linii.

Wybór kolejnych wyświetlanych linii obrazu jest realizowany przez układ sterownika wierszy SED1190F. Podczas trwania pierwszej linii obrazu na wejście DIN sterownika jest podawany stan wysoki, który jest zapisywany opadającym zboczem sygnału zegara wierszy YSCL. W tym momencie jest aktywowane wyświetlanie pierwszej linii obrazu. Kolejne impulsy zegara YSCL powodują przesuwanie tego stanu na następne wyjścia układu i tym samym wyświetlanie kolejnych linii obrazu. Aby obraz był stabilny, przełączanie wierszy musi się odbywać synchronicznie z zapisem do sterowników kolumn wyświetlanych danych, a więc tuż przed sygnałem LP zapisu linii. Standardowa częstotliwość odświeżania obrazu dla wyświetlacza EG2801S wynosi 60 obrazów na sekundę [1.], co przekłada się na częstotliwości sygnałów zegarowych YSCL i XSCL odpowiednio równe: 3840 Hz i 491520 Hz.

Oprócz sygnałów sterujących wpisywaniem do wyświetlacza danych i wybieraniem kolejnych wyświetlanych linii obrazu, moduł EG2801S wymaga dodatkowego sygnału kontrolnego FR. Sygnał ten zmienia swój stan na przeciwny na początku każdego pola obrazu i jest wykorzystywany przez sterowniki wierszy i kolumn do generowania napięć sterujących panelem LCD o zerowej składowej stałej.

Jak więc widać, sterowanie wyświetlaczem EG2801S jest dość złożone. Problem stanowi zarówno liczba sygnałów sterujących jak i ich wzajemne zależności czasowe. Z tego też powodu do obsługi wyświetlaczy graficznych LCD firma EPSON opracowała specjalizowane układy sterowników,

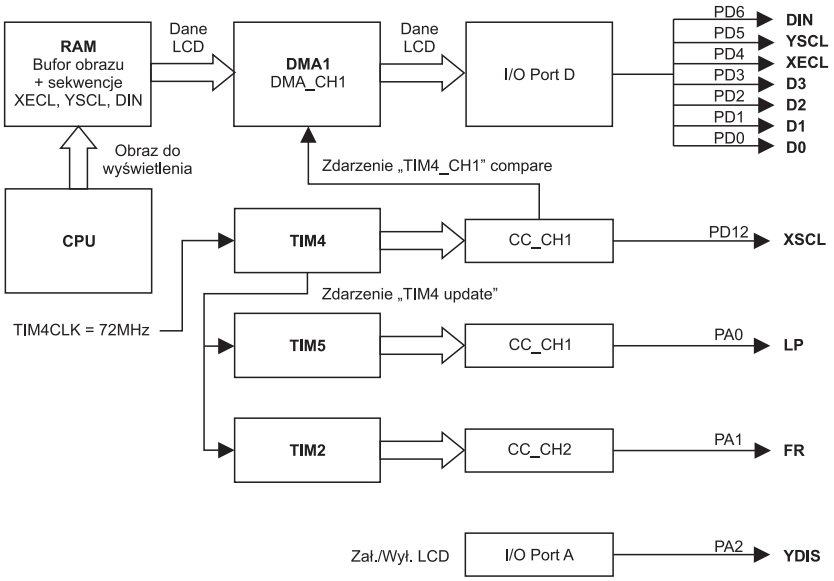


Rysunek 2. Przebiegi sterujące wyświetlaczem EG2801S-AR (Cx – numer kolumny, Wy – numer wiersza)

np. E-1330. Jednak w przeciwieństwie do samych wyświetlaczy, układy te są mało popularne i trudno dostępne. Okazuje się jednak, że wyświetlacz EG2801S może być sterowany bezpośrednio przez mikrokontroler rodziny STM32, bez potrzeby używania specjalizowanego układu sterownika LCD. Co więcej, taka obsługa wyświetlacza LCD może być zrealizowana w mikrokontrolerze w sposób całkowicie sprzętowy, nie obciążający jednostki centralnej.

Sterownik wyświetlacza na bazie mikrokontrolera STM32

Mikrokontrolery STM32 nie są wyposażone w układ sterownika wyświetlaczy graficznych LCD, który pozwalałby na bezpośrednią obsługę modułu EG2801S. Tym niemniej taki sterownik może zostać zrealizowany na bazie układów peryferyjnych mikrokontrolera. Przykład implementacji podobnego sterownika dla wyświetlacza typu EG9018S był już opisany w Elektronice Praktycznej [2]. Tamten wyświetlacz ma jednak znacznie prostsze sterowanie w porównaniu z modułem EG2801S, przez co jego sterownik nie może być użyty do obsługi wyświetlacza EG2801S. Przedstawiony w artykule sterownik modułu EG2801S stanowi zmodyfikowaną wersję wspomnianego wcześniej układu, zapewniającą generowanie wszystkich koniecznych sygnałów sterujących wyświetlaczem EG2801S. Schemat blokowy układu przedstawia **rysunek 3**.



Rysunek 3. Schemat blokowy sterownika wyświetlacza EG2801S-AR zbudowanego na bazie układów peryferyjnych mikrokontrolera STM32

Sterownik wyświetlacza EG2801S jest zbudowany z trzech układów czasowo-licznikowych TIM4, TIM5 i TIM2 oraz układu DMA1. Timer TIM4 pracuje jako generator sygnału zegara danych XSCL. Układ ten jest taktowany niepodzielonym sygnałem zegarowym magistrali APB1 i odmierza okres sygnału XSCL. Kanał CH1 tego timera pracuje w trybie PWM generując na swoim wyjściu przebieg prostokątny o wypełnieniu 50%, którego opadające zbocze występuje w połowie okresu zliczania TIM4. Sygnał ten jest doprowadzony do modułu LCD jako zegar danych XSCL.

Zdarzenie przeładowania licznika TIM4 (tj. TIM4 update event) taktuje układy czasowo-licznikowe TIM5 i TIM2, dzięki czemu generowane przez nie sygnały są zsynchronizowane z sygnałem zegara danych XSCL. Timer TIM5 odmierza czas trwania jednej linii obrazu, wynoszący w przypadku modułu EG2801S 128 takty zegara XSCL. Analogicznie to timera TIM4, kanał CH1 układu TIM5 także pracuje w trybie PWM. Zmienia on stan swego wyjścia z niskiego na wysoki podczas ostatniego taktu zegara XSCL w danej linii, generując tym samym dla wyświetlacza LCD impuls LP zapisu zawartości linii obrazu.

Układ czasowo-licznikowy TIM2 realizuje z kolei syntezę sygnału FR. W tym celu timer TIM2 odmierza czas trwania dwóch pełnych pól obrazu, co w przypadku wyświetlacza EG2801S trwa 16384 takty zegara XSCL. Kanał CH2 tego timera pracuje w trybie PWM generując przebieg prostokątny o wypełnieniu 50%, zmieniający swój stan na przeciwny podczas każdego pola obrazu.

Sekwencje pozostałych sygnałów sterujących wyświetlaczem LCD, tj.: zegar wyboru drivera kolumn XECL, zegar wierszy YSCL, znacznik początku pola obrazu DIN, są zapisane w pamięci RAM w buforze obrazu razem z danymi do wyświetlenia. Sekwencje przebiegów XECL, YSCL i DIN są zapisane odpowiednio na bitach nr b4, b5 i b6, podczas, gdy dane obrazu są zapisane na bitach b3…b0.

Przesyłanie danych z bufora obrazu do portów I/O PD6..PD0 mikrokontrolera realizuje kanał 1 kontrolera DMA1, pracującego w trybie kołowym z postinkrementacją adresu pamięci. Transmisja danych jest inicjowana przez zdarzenie porównania z kanału CH1 timera TIM4 (tj. TIM4_CH1 compare event). Ma ono miejsce w połowie okresu zliczania układu TIM4, tj. w momencie opadającego zbocza sygnału zegara XSCL. W odpowiedzi na to zdarzenie kontroler DMA1 przesyła jeden bajt z pamięci bufora obrazu do rejestru wyjściowego ODR portu PD. Przy częstotliwości sygnału zegara magistrali AHB równej 72 MHz, transmisja ta zajmuje ok. 120…160 ns. Takie jest też opóźnienie zboczy sygnałów XECL, YSCL i DIN w stosunku do zbocza opadającego sygnału zegara XSCL.

Listing 2. Funkcja inicjalizująca bufor obrazu

```
__INLINE void LCD_InitFrameBuffer(void)
{
  uint16_t i, j;  // loop row and column counters
  uint8_t  init_data;  // LCD frame buffer init data
  // write signal patterns and LCD init data to LCD frame buffer
  for (i=0; i < LCD_ROW_QTY; i++){
    for (j=0; j < LCD_COL_QTY/4; j++){
    // Set picture pixels to off state (bits b3..b0)
    init_data = LCD_PIXEL_OFF << 3 | LCD_PIXEL_OFF << 2 | LCD_PIXEL_OFF << 1 |
LCD_PIXEL_OFF;
    // Write XECL pattern (bit b4)
    if (j%16 == 15) init_data |= 0x10;  // bit is set in each 16th byte
    // Write YSCL pattern (bit b5)
    if (j == LCD_COL_QTY/4-1) init_data |= 0x20;  // bit is set in the last
byte of each row
    // Write DIN pattern (bit b6)
    if (i == 0 && j == LCD_COL_QTY/4-1) init_data |= 0x40; // bit is set in
the last byte of the first row
    if (i == 1 && j == 0) init_data |= 0x40;  // and the first byte of the
second row
    LCD_frame_buffer[i][j] = init_data;  // store init data in frame buffer
    }
  }
}

//-------------------------------------------------------------
// Configuration of LCD AC driving signal
// LCD AC driving signal FR is generated on pin PA1 by
// channel 2 of timer TIM2
// ------------------------------------------------------------
  // PA1 configuration as TIM2_CH2 output
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  // Clock source configuration: TIM2 is the slave of TIM4
  TIM_SelectSlaveMode(TIM2, TIM_SlaveMode_External1);
  TIM_SelectInputTrigger(TIM2, TIM_TS_ITR3);
  // TIM2 time base configuration (fclk=fTIM4)
  TIM_TimeBaseStructure.TIM_Period = 2 * LCD_ROW_QTY * LCD_COL_QTY/4 - 1;
  TIM_TimeBaseStructure.TIM_Prescaler = 0;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
  TIM_ARRPreloadConfig(TIM2, ENABLE);
  // TIM2 Channel2 configuration: PWM1 Mode
  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
  TIM_OCInitStructure.TIM_Pulse = LCD_ROW_QTY * LCD_COL_QTY/4;
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
  TIM_OC2Init(TIM2, &TIM_OCInitStructure);
  TIM_OC2PreloadConfig(TIM2, TIM_OCPreload_Enable);
// ------------------------------------------------------------
// Configuration of LCD data transfer
// LCD data is transfered to pins PD3..PD0 by DMA1 controller
// via channel 1. DMA transfer is requested by compare event in channel 1 of
timer TIM4
// ------------------------------------------------------------
  // DMA1 Channel 1 configuration
  DMA_DeInit(DMA1_Channel1);
  DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&(GPIOD->ODR);
  DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)&LCD_frame_buffer[0][0];
  DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST;
  DMA_InitStructure.DMA_BufferSize = LCD_ROW_QTY * LCD_COL_QTY/4;
  DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
  DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
  DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Word;
  DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
  DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
  DMA_InitStructure.DMA_Priority = DMA_Priority_VeryHigh;
  DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
  DMA_Init(DMA1_Channel1, &DMA_InitStructure);
  // Enable DMA1 Channel 1
  DMA_Cmd(DMA1_Channel1, ENABLE);
  // Set compare event in channel 1 of timer TIM4 as request
  //to start DMA transfer
  TIM_DMACmd(TIM4, TIM_DMA_CC1, ENABLE);

// ------------------------------------------------------------
// Configuration of display control signal
// LCD display is enabled by high level on pin PA2 (YDIS signal)
// ------------------------------------------------------------
  // PA2 configuration as GPIO output
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  // Enable LCD module
  LCD_DISPLAY_ON;
// ------------------------------------------------------------
// Start servicing LCD
// ------------------------------------------------------------
// Request the first DMA transfer
  TIM_GenerateEvent(TIM4, TIM_EventSource_CC1);
  // Enable TIM2 counter
  TIM_Cmd(TIM2, ENABLE);
  // Enable TIM5 counter
  TIM_Cmd(TIM5, ENABLE);
  // Enable TIM4  counter
  TIM_Cmd(TIM4, ENABLE);
}
```

Ostatni z sygnałów kontrolnych wyświetlacza EG2801S, tj. YDIS sterujący załączaniem i wyłączaniem modułu LCD, jest uzyskiwany w drodze programowej obsługi linii nr 2 portu PA.

Realizacja praktyczna sterownika

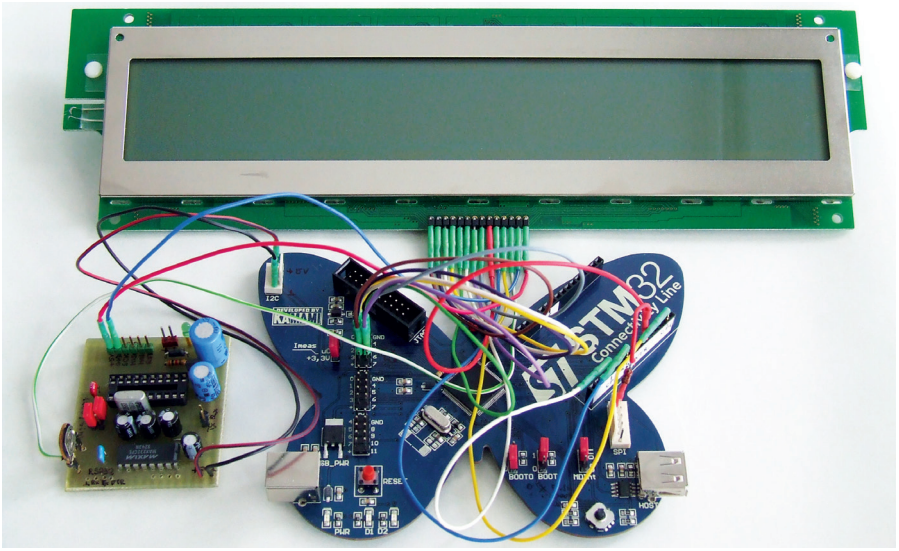
Model sterownika został wykonany na bazie zestawu uruchomieniowego STM32 Butterfly, wyposażonego w mikrokontroler STM32F107 (**fotografia 4**). Schemat połączeń modułu wyświetlacza EG2801S z płytką STM32 Butterfly pokazano na **rysunku 5**. Wyświetlacz LCD jest bezpośrednio dołączony do portów mikrokontrolera. Co prawda, moduł EG2801S jest przeznaczony do sterowania sygnałami logicznymi w standardzie 5V, jednak jak wykazały próby z egzemplarzem wyświetlacza typu EG2801S-AR oraz typu ECM-A0083, sterowanie tych modułów sygnałami o poziomie +3,3V działa poprawnie.

Linia YDIS sterująca załączaniem / wyłączaniem modułu EG2801S jest dodatkowo spolaryzowana rezystorem R1 do masy. Dzięki temu podczas restartu mikrokontrolera, jak również podczas jego programowania, linia YDIS znajduje się w stanie niskim, wymuszając tym samym wyłączenie modułu wyświetlacza, co zabezpiecza panel LCD przed uszkodzeniem wskutek pojawienia się na nim napięcia o niezerowej składowej stałej. Sytuacja taka może mieć miejsce przy braku sygnałów sterujących wyświetlaczem LCD. Układ zabezpieczający wykorzystuje fakt, iż podczas restartu mikrokontrolera wszystkie jego porty I/O, w tym linia PA2, są konfigurowane jako niespolaryzowane wejścia. O poziomie napięcia w linii YDIS decyduje w takich warunkach rezystor R1.

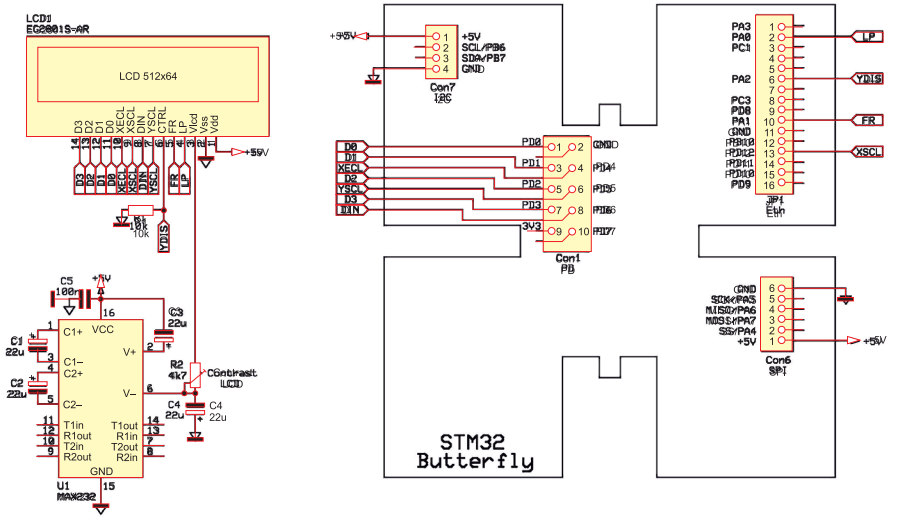
Do generacji ujemnego napięcia VLCD, koniecznego do polaryzacji panelu LCD, wykorzystano pompę ładunkową z układu MAX232. W temperaturze pokojowej optymalny kontrast obrazu uzyskano przy napięciu na wejściu VLCD równym -7V, natomiast obraz całkowicie znikał / był całkowicie zaczerniony przy napięciu na wejściu VLCD odpowiednio równym -6V i -9V. Poziom napięcia ujemnego dostarczanego przez układ MAX232 okazał się w warunkach modelowych zupełnie wystarczający. Należy jednak mieć na uwadze, iż nie jest to regułą i w innych warunkach może być konieczny znacznie szerszy zakres regulacji napięcia VLCD. Podawany przez producenta wyświetlacza EG2801S zakres regulacji napięcia polaryzacji panelu LCD zawiera się w granicach -4…-17V [1]

Oprogramowanie

Kod źródłowy funkcji *LCD_Init()* konfigurującej układy peryferyjne mikrokontrolera STM32F107 do pracy jako sterownik wyświetlacza EG2801S zamieszczono na **listingu 1**. Programowanie poszczególnych



Fotografia 4. Układ testowy, który posłużył do opracowania sterownika wyświetlacza EG2801S-AR (tutaj z podłączonym modułem typu ECM-A0083)



Rysunek 5. Schemat podłączenia wyświetlacza EG2801S do zestawu uruchomieniowego STM32 Butterfly

układów nie różni się w sposób znaczący od konfiguracji peryferii mikrokontrolera STM32, szczegółowo opisanej przy okazji omawiania konstrukcji sterownika dla wyświetlacza EG9018C. Z tego też względu osoby zainteresowane szczegółami programowania poszczególnych układów peryferyjnych powinny sięgnąć do tamtego artykułu. Jedyną nowością w stosunku do sterownika EG9018C jest wywołanie funkcji *LCD_InitFrameBuffer()* (**listing 2**). Jej zadanie jest dwojakie. Po pierwsze, czyści ona pamięć obrazu, zapisując na pozycjach bitowych b3..b0 wartości zerowe. Po drugie, funkcja ta zapisuje w buforze obrazu sekwencje sygnałów XECL, YSCL i DIN sterujących wyświetlaczem. W przypadku sygnału XECL inicjalizacja polega na ustawieniu bitu b4 w każdym 16-tym bajcie w buforze obrazu, co odpowiada wygenerowaniu dodatniego impulsu na linii XECL podczas każdego 16-tego impulsu zegara XSCL. Z kolei w przypadku sygnału YSCL funkcja inicjalizacji bufora obrazu ustawia bit b5 w każdym ostatnim słowie danych należącym do danej linii obrazu, co odpowiada wygenerowaniu dodatniego impulsu na linii YSCL razem z impulsem LP zapisu danych w tej linii. Wreszcie w przypadku sygnału DIN, inicjalizacja bufora obrazu polega na ustawieniu bitu b6 w ostatnim bajcie pierwszej linii i pierwszym bajcie drugiej linii, w wyniku czego na linii DIN pojawia się stan wysoki podczas pierwszego opadającego zbocza zegara YSCL w danym polu obrazu.

Skonfigurowany sterownik wyświetlacza LCD nie wymaga dalszej uwagi ze strony jednostki centralnej. Generacja sygnałów sterujących wyświetlaczem oraz przesyłanie do niego danych do wyświetlenia są realizowane w tle. Z poziomu programu użytkow-

Listing 3. Funkcje realizujące zaświecanie i gaszenie piksela

```
void LCD_PixelOn(uint16_t Xpos, uint16_t Ypos)
{
// column number, row number and bit number in frame buffer
  uint8_t colNo, rowNo, bitNo;

// if pixel coordinates within range
  if (Xpos < LCD_XSIZE && Ypos < LCD_YSIZE){
// Calculate pixel position in LCD frame buffer
    rowNo = (Ypos < LCD_ROW_QTY) ? Ypos : Ypos - LCD_ROW_QTY;
    colNo = (Ypos < LCD_ROW_QTY) ? Xpos/4 : (Xpos + LCD_XSIZE)/4;
    bitNo = 3 - Xpos%4;
    // Draw pixel
// (acces to pixel bit is performed via bit-banding region of SRAM)
    *(__IO uint32_t *)(SRAM_BB_BASE | (((uint32_t)&LCD_frame_buffer[rowNo][colNo] - SRAM_BASE) << 5) \
                       | ((bitNo) << 2)) = LCD_PIXEL_ON;
  }
}

void LCD_PixelOff(uint16_t Xpos, uint16_t Ypos)
{
// column number, row number and bit number in frame buffer
  uint8_t colNo, rowNo, bitNo;

// if pixel coordinates within range
  if (Xpos < LCD_XSIZE && Ypos < LCD_YSIZE){
    // Calculate pixel position in LCD frame buffer
    rowNo = (Ypos < LCD_ROW_QTY) ? Ypos : Ypos - LCD_ROW_QTY;
    colNo = (Ypos < LCD_ROW_QTY) ? Xpos/4 : (Xpos + LCD_XSIZE)/4;
    bitNo = 3 - Xpos%4;
// Draw pixel
// (acces to pixel bit is performed via bit-banding region of SRAM)
    *(__IO uint32_t *)(SRAM_BB_BASE | (((uint32_t)&LCD_frame_buffer[rowNo][colNo] - SRAM_BASE) << 5) \
                       | ((bitNo) << 2)) = LCD_PIXEL_OFF;
  }
}
```

Fotografia 6. Efekty pracy programu demonstracyjnego „Life": a) ekran startowy, b) jeden z kroków symulacji, c) koniec symulacji

nika obsługa wyświetlacza ogranicza się do operacji wykonywanych na pamięci bufora obrazu. Zapalenie na wyświetlaczu piksela wymaga ustawienia w buforze obrazu bitu odpowiadającego temu punktowi, natomiast zgaszenie piksela – skasowania tego bitu. Kody funkcji realizujących obie operacje przedstawia **listing 3**.

Same funkcje sterowania pojedynczym pikselem nie są zazwyczaj wywoływane w programie użytkownika bezpośrednio, lecz służą do budowy funkcji wyższej warstwy, np. rysujących odcinki, figury geometryczne, czy też wyświetlających mapy bitowe. Nie inaczej jest w przypadku sterownika wyświetlacza EG2801S. Znajdujący się w dołączonych do artykułu materiałach moduł programowy *lcd_EG2201_4801.c,* poza funkcją *LCD_Init(),* zawiera także podstawowe funkcje rysowania odcinków, prostokątów, okręgów, map bitowych oraz tekstów. Są to te same funkcje, które zostały stworzone na potrzeby wspomnianego wcześniej sterownika wyświetlacza EG9018C. Ich dokładniejsze omówienie można znaleźć w artykule opisującym ten sterownik.

W celu demonstracji działania omówionego w artykule sterownika wyświetlacza EG2801S, został opracowany prosty program realizujący dobrze wszystkim znaną grę w życie. Algorytm programu implementuje oryginalne reguły zdefiniowane przez autora tej gry, brytyjskiego matematyka Johna Conwaya [3]. Efekty działania programu pokazano na **fotografii 6**.

Podsumowanie

Opisany sterownik wyświetlacza LCD został praktycznie przetestowany z modułami typu EG2801S-AR oraz ECM-A0083, jednak powinien on również działać z innymi wyświetlaczami LCD o takiej samej konstrukcji. Łatwo je rozpoznać po zamontowanych na obwodzie drukowanym wyświetlacza układach sterowników wierszy SED1190F i sterowników kolumn SED1180F. Deklaracje w pliku nagłówkowym *lcd_EG2201_4801.h* umożliwiają przystosowanie sterownika do obsługi każdego z typów wyświetlaczy wymienionych w tabeli 1, bez konieczności dokonywania jakichkolwiek zmian w kodzie programu.

Ponieważ zaprezentowany sterownik realizuje bezpośrednie sterowanie drajwerami kolumn i wierszy panelu LCD, umożliwia on zaimplementowanie funkcji nieprzewidzianej oryginalnie przez producenta wyświetlacza, a mianowicie uzyskanie na wyświetlaczu dodatkowego stopnia szarości. W tym celu należy powiększyć dwukrotnie rozmiar bufora obrazu, tak aby obejmował on dwa kolejne obrazy. Dodatkowy stopień szarości jest uzyskiwany w takiej konfiguracji przez odpowiednie kodowanie stanu piksela w obu obrazach. Jeżeli w obu obrazach dany piksel będzie wyzerowany, to będzie on zgaszony. Z kolei, jeżeli w obu obrazach piksel będzie ustawiony, to będzie on zaświecony. Natomiast w przypadku kombinacji stanu piksela w obu obrazach równej 10 lub 01 uzyskany zostanie półton.

W powyższej konfiguracji bufora obrazu możliwe jest też zrezygnowanie z generowania sygnału kontrolnego FR przez układ czasowo-licznikowy TIM2 i zapisanie sekwencji tego sygnału w buforze obrazu na pozycji nieużywanego obecnie bitu b7. Bit ten powinien być równy 0 we wszystkich słowach danych pierwszego obrazu i być ustawiony we wszystkich słowach danych drugiego obrazu.

Aleksander Borysiuk
alex_priv@wp.pl

Bibliografia

[1] Specifications for 1/64 Multiplexing LCD Module EG2201S-AR (128 x 64 dots), EG2401S-AR (256 x 64 dots), EG2801S-AR (512 x 64 dots), EG4401S-AR (256 x 128 dots), EG4801S-AR (512 x 128 dots), informacje techniczne firmy Seiko Epson Corporation
[2] Aleksander Borysiuk, Sterownik graficznego wyświetlacza LCD 8,4" typu EG9018C z mikrokontrolera STM32, Elektronika Praktyczna 11/2013
[3] Gra w życie, http://pl.wikipedia.org/wiki/Gra_w_życie