Sterownik graficznego wyświetlacza LCD 8,4” typu EG9018C z mikrokontrolera STM32

Wyświetlacze EG9018C były produkowane przez firmę EPSON. Co prawda, obecnie nie są one oferowane, ale nadal można je nabyć w bardzo atrakcyjnej cenie na rynku wtórnym, np. na aukcjach internetowych. Są to zarówno egzemplarze nowe, jak i też pochodzące z demontażu. Niestety, w przeciwieństwie do większości wyświetlaczy graficznych LCD o mniejszej przekątnej i rozdzielczościach ekranu, wyświetlacze EG9018C nie mają wbudowanego układu kontrolera, przez co ich implementacja w systemach mikroprocesorowych wydaje się być bardzo złożona lub wręcz niemożliwa. Prawdopodobnie jest to przyczyną niskiej ceny tych wyświetlaczy.

***Rekomendacje:** doskonały projekt referencyjny, który może posłużyć do zbudowania sterownika również dla innego modelu wyświetlacza.*

Fotografia 1. Wyświetlacz typu EG9018C firmy EPSON

Jak pokazuje praktyka, bezpośrednie sterowanie wyświetlaczem EG9018C nie jest aż takie trudne. W artykule przedstawiono sposób wykonania sprzętowej obsługi wyświetlacza EG9018C za pomocą mikrokontrolera STM32F107. Przykładową aplikację zrealizowano na bazie zestawu uruchomieniowego *STM32 Butterfly*.

Charakterystyka wyświetlacza

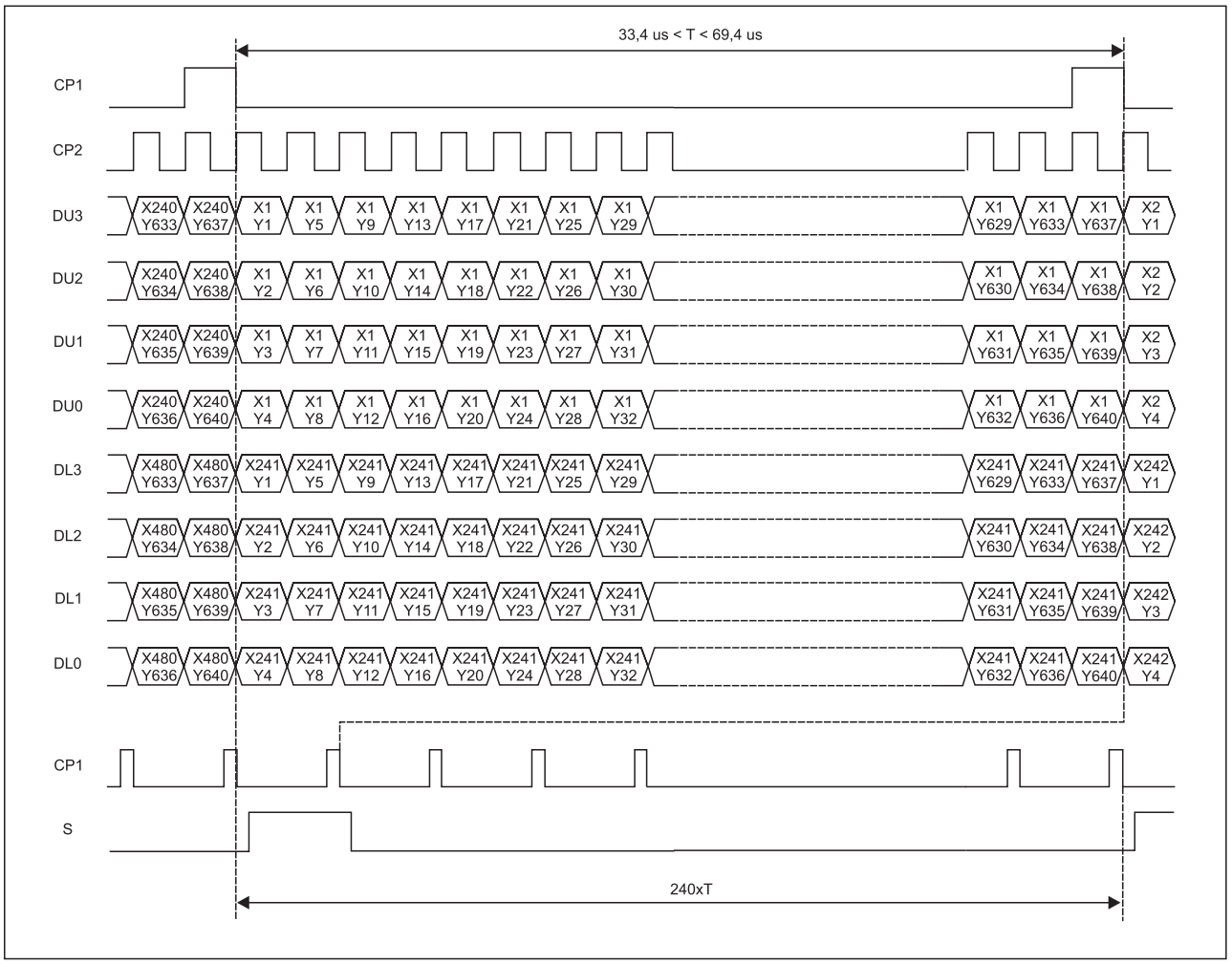
Pokazany na **fotografii 1** wyświetlacz typu EG9018C jest monochromatycznym wyświetlaczem LCD o przekątnej 8,4” i rozdzielczości 640×480 punktów (VGA). Ma powierzchnię roboczą ekranu o wymiarach 196,0 mm×147,6 mm. Ekran jest podświetlany lampą CCFL zasilaną przez obwody, w które jest wyposażony moduł. Oprócz tego wbudowana przetwornica generuje napięcia ujemne służące do polaryzacji ekranu LCD, dzięki czemu wyświetlacz wymaga tylko jednego napięcia zasilania o wartości +5 V. Wszystkie sygnały sterujące modułem wyświetlacza EG9018C wyprowadzono na złącze ZIFF 30 (**tabela 1**) przeznaczone do dołączenia płaskiego przewodu taśmowego FFC o rastrze 1,0 mm. Warto zwrócić uwagę, że firma Sharp produkowała identyczne wyświetlacze [1], jednak pozbawione układów zasilania lampy CCFL i generowania napięcia polaryzacji ekranu LCD.

Sygnały sterujące wyświetlaczem EG9018C pokazano na **rysunku 2**. Obsługa wyświetlacza sprowadza się do szeregowego wpisywania zawartości kolejnych linii obrazu. W jednym takcie sygnału zegarowego CP2 do wyświetlacza przesyłane są stany 8 pikseli: 4 punktów linii leżącej w górnej połówce ekranu (bity D3U…D0U) oraz 4 punktów linii leżącej w dolnej połówce ekranu (bity D3L…D0L). Przyporządkowanie poszczególnych bitów D3U…D0U, D3L…D0L do punktów ekranu określają podane na rys. 1 współrzędne X i Y, gdzie X oznacza numer wiersza, a Y – numer kolumny. Wpisywany bit o wartości 1 odpowiada punktowi zaświeconemu, natomiast bit o wartości 0 odpowiada punktowi zgaszonemu.

Po przesłaniu do wyświetlacza zawartości całej linii, co zajmuje 160 impulsów zegara CP2, opadające zbocze impulsu na linii CP1 zatrzaskuje wpisane dane. W tym momencie rozpoczyna się ich wyświetlanie. Jednocześnie wyświetlane są 2 linie: jedna w górnej połówce i jedna w dolnej połówce ekranu. Obie linie są wyświetlane przez czas wpisywania do wyświetlacza zawartości kolejnych dwóch linii obrazu, po czym cały cykl powtarza się. Obraz na wyświetlaczu tworzony jest więc dynamicznie w drodze wyświetlania kolejnych linii. Jak łatwo zauważyć, pełen obraz powstaje po 240 impulsach sygnału CP1. Synchronizację obrazu zapewnia linia S, której stan wysoki podczas impulsu CP1 sygnalizuje początek pola obrazu, tj. zatrzaskiwanie zawartości linii nr 1 i 241.

Obsługa wyświetlacza przez mikrokontroler STM32

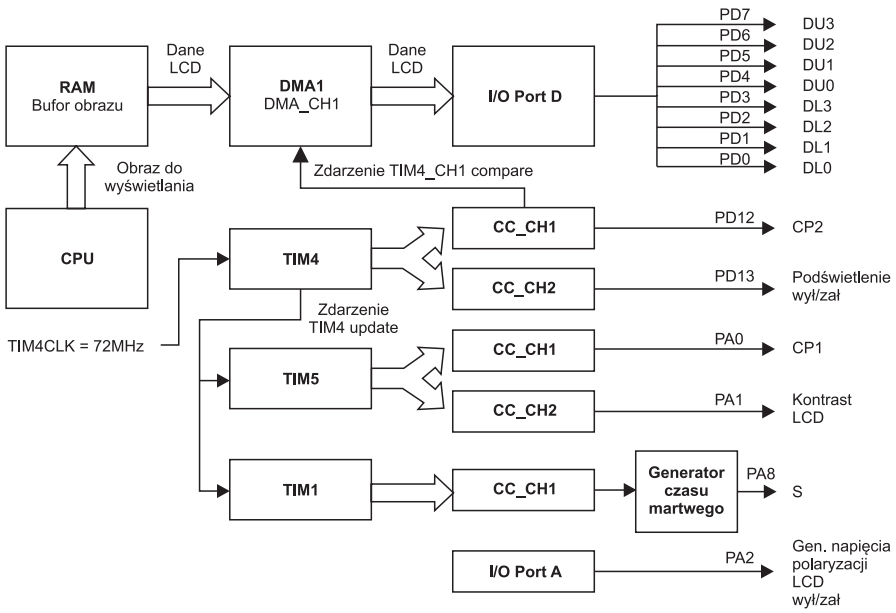
Na pierwszy rzut oka mogłoby się wydawać, że sygnały sterujące wyświetlaczem EG9018C mogą zostać bez trudu wygenerowane przez dowolny mikrokontroler w sposób programowy. Dokładniejsza analiza pokazuje jednak, że nie jest to aż tak łatwe. Problem leży w szybkości generowania sygnałów. Aby wyświetlać pełen obraz, wyświetlacz musi być ciągle odświeżany. Ponie-



Rysunek 2. Przebiegi charakteryzujące przesyłanie danych do wyświetlacza EG9018C

waż minimalna częstotliwość odświeżania pola obrazu dla modułu EG9018C jest równa 60 Hz, dane do wyświetlacza muszą być wpisywane w tempie co najmniej równym 60 Hz×240 linii×160 B/linię = 2304000 B/s. Uzyskanie takiej szybkości transferu w sposób programowy nie jest proste, mimo iż szybki mikrokontroler, taki jak STM32, jest w stanie sobie z tym poradzić. W Internecie publikowane są projekty układów, w których obsługa wyświetlacza EG9018C jest realizowana w ten sposób [2]. Te rozwiązania mają jednak podstawową wadę: programowe generowanie sygnałów sterujących wyświetlaczem tak silnie obciąża mikrokontroler, że pozostaje mu niewiele mocy obliczeniowej na wykonywanie innych zadań.

W wypadku mikrokontrolerów STM32 istnieje inna możliwość realizacji obsługi wyświetlacza EG9018C. Co prawda, te mikrokontrolery nie mają kontrolera LCD przeznaczonego do obsługi wyświetlacza EG9018C, ale ich bogate wyposażenie w elastycznie konfigurowane układy peryferyjne pozwala na implementację takiego sterownika w oparciu o peryferia mikrokontrolera. Co więcej, taki sterownik LCD może zostać zrealizowany w sposób całkowicie sprzętowy,



Rysunek 3. Schemat blokowy sterownika wyświetlacza LCD zbudowanego na bazie układów peryferyjnych mikrokontrolera STM32

Tabela 1. Rozkład wyprowadzeń wyświetlacza EG9018C i ich funkcje

Pin Nr	Sygnał	Opis
1	DU0	Bit b0 słowa danych dla górnej połówki wyświetlacza
2	CCFL brightness control	Regulacja jasności lampy CCFL (L – jasność minimalna, H – jasność maksymalna) Uwaga: Nie wszystkie wersje wyświetlaczy EG9018C mają zaimplementowaną tą funkcję
3	DU1	Bit b1 słowa danych dla górnej połówki wyświetlacza
4	GND	Masa
5	DU2	Bit b2 słowa danych dla górnej połówki wyświetlacza
6	VEE control	Regulacja kontrastu / ujemnego napięcia polaryzacji LCD (0..+5V)
7	DU3	Bit b3 słowa danych dla górnej połówki wyświetlacza
8	GND	Masa
9	DL0	Bit b0 słowa danych dla dolnej połówki wyświetlacza
10	CCFL inverter ON	Załączenie podświetlenia ekranu (L – zasilanie CCFL załączone, H – zasilanie CCFL wyłączone)
11	DL1	Bit b1 słowa danych dla dolnej połówki wyświetlacza
12	GND	Masa
13	DL2	Bit b2 słowa danych dla dolnej połówki wyświetlacza
14	VEE inverter ON	Załączenie generatora ujemnego napięcia polaryzacji LCD (L – zasilanie generatora załączone, H – zasilanie generatora wyłączone)
15	DL3	Bit b3 słowa danych dla dolnej połówki wyświetlacza
16	GND	Masa
17	CP2	Zegar danych (aktywne zbocze opadające)
18	GND	Masa
19	CP1	Zapis zawartości linii obrazu (aktywne zbocze opadające)
20	S	Znacznik początku pola obrazu
21	GND	Masa
22,23	Vcc for VEE inverter	Zasilanie generatora ujemnego napięcia polaryzacji LCD (+5V)
24,25	Vdd	Zasilanie (+5V)
26	GND	Masa
27		Przekrosowany pin 3 złącza J203
28		Przekrosowany pin 1 złącza J203
29		Przekrosowany pin 2 złącza J203
30		Przekrosowany pin 4 złącza J203

niewymagający po skonfigurowaniu układów peryferyjnych żadnej uwagi ze strony jednostki centralnej. Schemat blokowy proponowanego interfejsu-sterownika wyświetlacza LCD pokazano na **rysunku 3**.

Struktura układu sterownika LCD została w dużej mierze narzucona przez ograniczenia zestawu uruchomieniowego STM32 Butterfly, który posłużył do budowy prototypu. W tym zestawie nie wszystkie porty mikrokontrolera STM32F107 są wyprowadzone na złącza szpilkowe. W związku z tym, dostępne w postaci złącz porty mikrokontrolera zdeterminowały grupę układów czasowo-licznikowych, które mogły zostać wykorzystane w projekcie. Te z kolei wpłynęły na wybór kontrolera DMA. W przypadku realizacji układu na innym zestawie uruchomieniowym lub też innym typie mikrokontrolera STM32, nic nie stoi na przeszkodzie, aby wybrać inny zestaw układów czasowo-licznikowych oraz portów mikrokontrolera, na które będą wyprowadzone sygnały sterujące wyświetlaczem LCD.

Rdzeniem sterownika wyświetlacza LCD jest układ czasowo-licznikowy TIM4. Układ ten pracuje w roli timera odmierzający okres sygnału zegara CP2. Licznik TIM4 jest taktowany niepodzielonym sygnałem zegarowym o częstotliwości 72 MHz uzyskiwanym z magistrali APB1 i liczy on do $N_{TIM4}=27$, co daje częstotliwość sygnału CP2 równą $f_{CP2}=72\ MHz/(N_{TIM4}+1)=2{,}571\ MHz$. Częstotliwość odświeżania wyświetlacza EG9018C jest w tym przypadku równa $f_{FRM}=2{,}571\ MHz/160$ *zapisów/linię/240 linii/obraz=67 obrazów/s*. Kanał pierwszy układu TIM4 pracuje w trybie PWM generując przebieg o wypełnieniu 50% i okresie równym okresowi przepełniania licznika TIM4, to jest 388,89 ns. Przebieg ten jest wyprowadzony na port PD12 mikrokontrolera w roli sygnału zegarowego CP2 dla wyświetlacza LCD.

Zdarzenie przeładowania licznika TIM4 (zdarzenie *TIM4 update*) taktuje układy czasowo-licznikowe TIM5 i TIM1, dzięki czemu generowane przez nie sygnały są zsynchronizowane z sygnałem zegarowym CP2, generowanym przez kanał pierwszy układu czasowo-licznikowego TIM4.

Układ czasowo-licznikowy TIM5 pracuje w trybie timera odmierzającego czas trwania jednej linii obrazu, równy 160 okresom sygnału CP2. Kanał pierwszy tego układu pracuje w trybie PWM zmieniając poziom na wyjściu z niskiego na wysoki podczas ostatniego taktu zegara CP2 w danej linii obrazu. Przebieg ten jest wyprowadzony na port PA0 mikrokontrolera jako sygnał CP1 zapisu linii obrazu do wyświetlacza LCD.

Również ostatni z układów czasowo-licznikowych TIM1 pracuje w trybie timera. Odmierza on czas trwania pola obrazu, który jest równy 38400 okresom sygnału CP2. Kanał pierwszy tego układu pracuje w trybie PWM zmieniając poziom wyjściowy z wysokiego na niski po zakończeniu transmisji pierwszej linii obrazu. W układzie prototypowym przyjęto, że moment ten następuje po 166 takcie zegara CP2, licząc od początku pola obrazu. Otrzymany w ten sposób sygnał początku pola obrazu jeszcze nie nadaje się do sterowania wyświetlaczem EG9018C, ponieważ narastające zbocze tego sygnału wypada dokładnie na początku pola obrazu, a powinno być w stosunku do niego opóźnione (rys. 1). Odpowiednie opóźnienie realizuje generator czasu martwego, w jaki jest wyposażony kanał 1 układu TIM1. Uzyskany w ten sposób przebieg

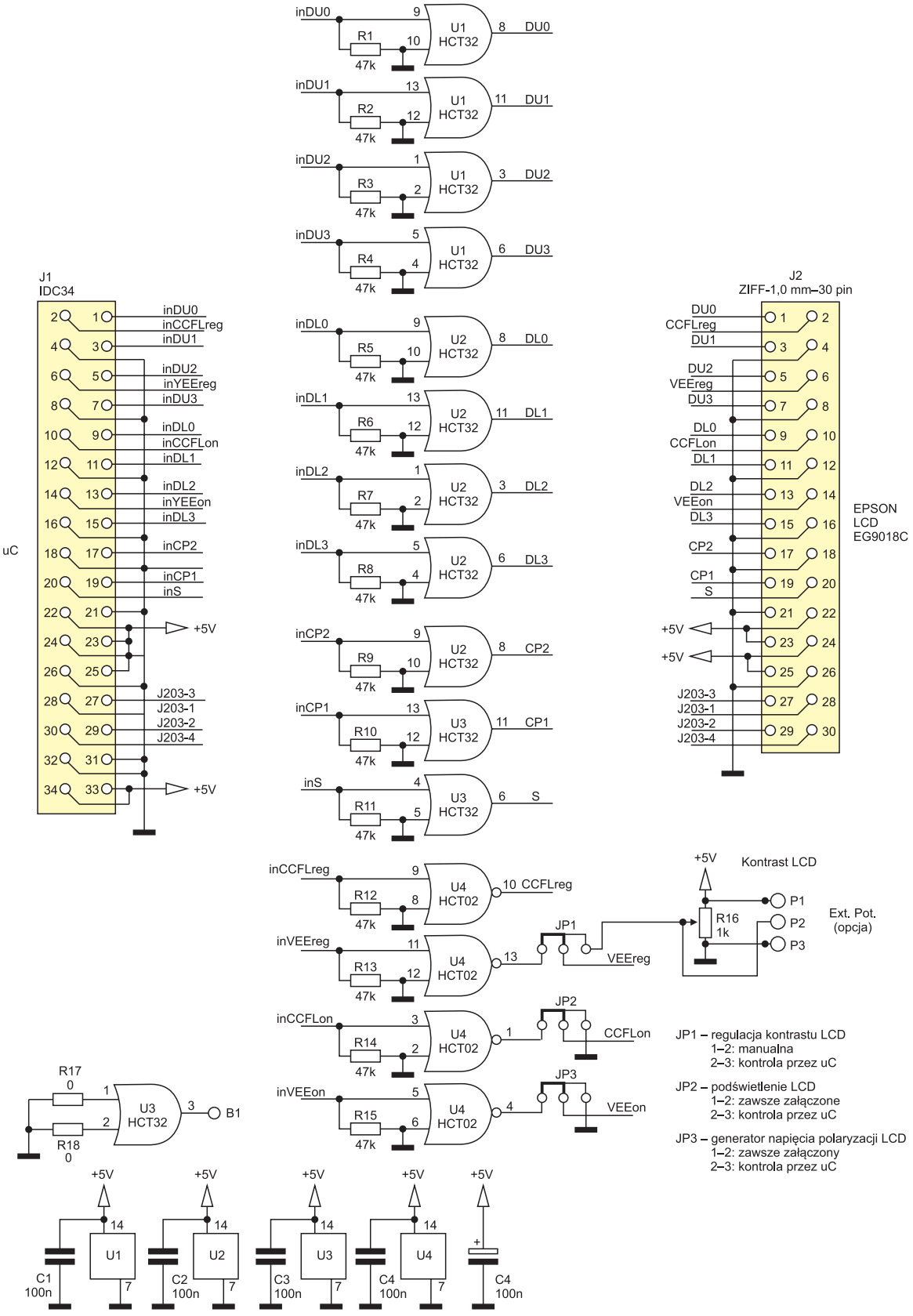
Tabela 2. Połączenia pomiędzy modułem wyświetlacza EG9018C a płytką STM32 Butterfly

Sygnał EG9018C	Złącze LCD	Kierunek sygnału	Złącze STM32 Butterfly	Port STM32
inDU0	J1/pin 1	<——	Con1/pin 4	PD4
inDU1	J1/pin 3	<——	Con1/pin 6	PD5
inDU2	J1/pin 5	<——	Con1/pin 8	PD6
inVEEreg	J1/pin 6	<——	JP1/pin 10	PA1
inDU3	J1/pin 7	<——	Con1/pin 10	PD7
inDL0	J1/pin 9	<——	Con1/pin 1	PD0
inCCFLon	J1/pin 10	<——	JP3/pin 1	PD13
inDL1	J1/pin 11	<——	Con1/pin 3	PD1
inDL2	J1/pin 13	<——	Con1/pin 5	PD2
inVEEon	J1/pin 14	<——	JP1/pin 6	PA2
inDL3	J1/pin 15	<——	Con1/pin 7	PD3
inCP2	J1/pin 17	<——	JP1/pin 13	PD12
inCP1	J1/pin 19	<——	JP1/pin 2	PA0
inS	J1/pin 20	<——	JP2/pin 11	PA8
GND	J1/pin 21	——	Con6/pin 6	GND
Vdd	J1/pin 24	——	Con6/pin 1	+5V

jest wyprowadzony na port PA8 mikrokontrolera jako sygnał S.

Przy ewentualnym modyfikowaniu obwodu generatora sygnału S należy pamiętać, że nie każdy układ czasowo-licznikowy w mikrokontrolerze STM32F107 jest wyposażony w generator czasu martwego. Właściwość tą posiadają tylko kanały 1, 2 i 3 układów TIM1 i TIM8. W związku z tym tylko one mogą zostać wykorzystane do generowania sygnału S znacznika początku pola obrazu LCD.

Bufor obrazu jest zorganizowany w pamięci RAM mikrokontrolera, gdzie zajmuje 38400B. Dane z bufora są przesyłane na wyjścia PD7..PD0 przez kanał 1 kontrolera DMA1 pracującego w trybie kołowym z postinkrementacją adresu pamięci. Transmisja DMA jest inicjowana przez zdarzenie porównania z kanału 1 timera TIM4 (zdarzenie *TIM4_CH1 compare*). Zdarzenie to jest generowane w połowie okresu sygnału zegara CP2, tj. w momencie opadającego zbocza CP2. W odpowiedzi na to zdarzenie



Rysunek 4. Schemat ideowy układu dopasowania poziomów logicznych sygnałów dla wyświetlacza EG9018C

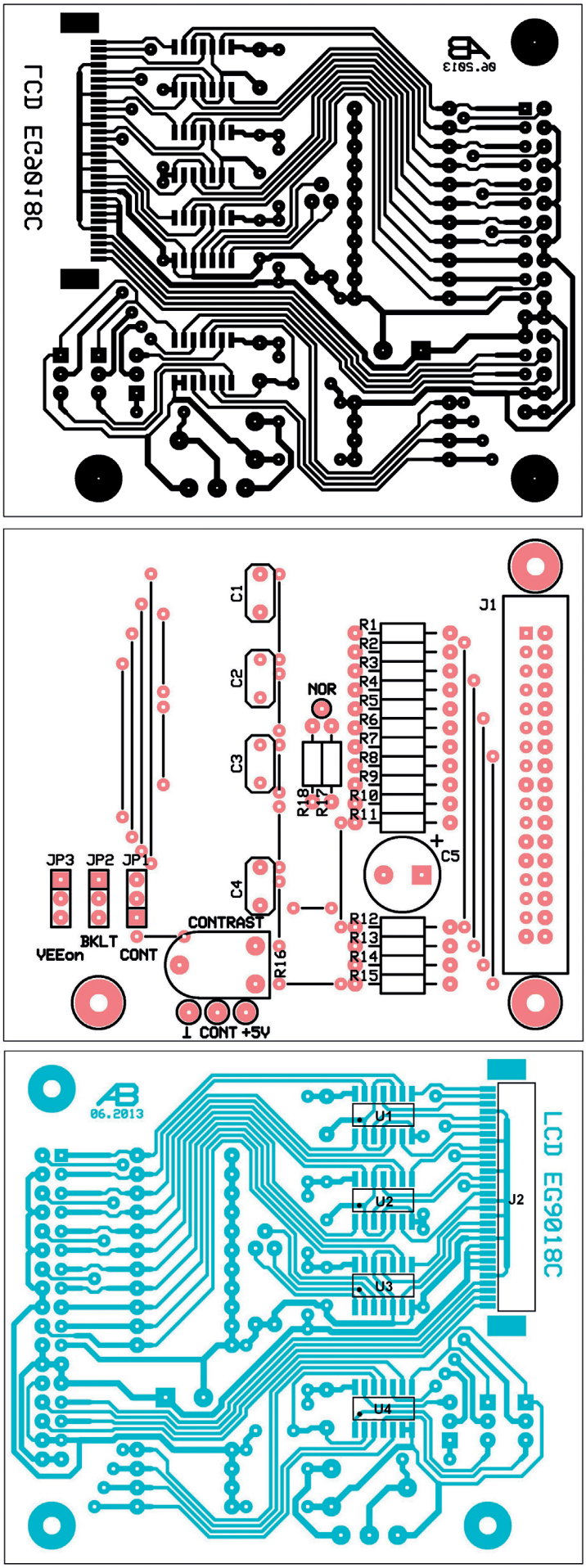
kontroler DMA1 pobiera z bufora obrazu 1B danych i wpisuje go do rejestru *ODR* sterującego wyjściami portu D. Warto zauważyć, ze ten tryb dostępu do rejestru *ODR* portu D blokuje możliwość ewentualnego wykorzystania pozostałych linii portu D (tj. linii PD15..PD8) jako wyjść ogólnego przeznaczenia. Wynika to z faktu, że kontroler DMA zapisuje cały rejestr *ODR* portu D, a nie tylko jego 8 najmłodszych bitów. Nie oznacza to jednak, że linie PD15..PD8 stają się bezużyteczne. Wciąż mogą one być używane tylko jako linie wejściowe ogólnego przeznaczenia bądź też jako wejścia/wyjścia funkcji alternatywnych.

Układ sterownika LCD, poza sygnałami obsługującymi zapisywanie danych do wyświetlacza, generuje także trzy sygnały pomocnicze, odpowiadające za: sterowanie podświetleniem, regulację kontrastu oraz załączenie generatora napięcia polaryzacji ekranu LCD.

Do sterowania podświetleniem ekranu LCD została użyta linia PD13 mikrokontrolera. Ponieważ, jak już wcześniej wspomniano, linia ta nie może w trybie wyjścia ogólnego przeznaczenia, skorzystano z możliwości sterowania tym wyjściem z poziomu funkcji alternatywnej. Sygnał sterujący załączaniem i wyłączaniem podświetlenia ekranu LCD jest generowany przez kanał drugi timera TIM4 jako sygnał PWM o dwóch możliwych współczynnikach wypełnienia (0% i 100%), które odpowiadają odpowiednio poziomowi niskiemu i wysokiemu na wyjściu PD13.

Ponieważ wszystkie dostępne dla autora moduły wyświetlaczy EG9018C nie miały obwodów odpowiedzialnych za regulowanie jasności podświetlania ekranu LCD, funkcja ta nie została zaimplementowana w układzie sterownika. W razie potrzeby, regulację jasności podświetlenia ekranu można zrealizować na drodze sterowania za pomocą sygnału PWM, wykorzystując jeden z wolnych kanałów timera TIM5. W ten właśnie sposób jest uzyskiwany sygnał służący do regulowania kontrastu wyświetlacza LCD. Jest on generowany w kanale 2 timera TIM5. Kontrast wyświetlacza LCD jest zmieniany poprzez programową zmianę współczynnika wypełnienia sygnału PWM generowanego w tym kanale i dostępnego na porcie PA1 mikrokontrolera.

Sygnał załączający generator ujemnego napięcia polaryzacji ekranu LCD jest wyprowadzony na port PA2 mikrokontrolera. Podczas normalnej pracy układu możliwość programowego sterowania generatorem ujemnego napięcia polaryzacji ekranu LCD jest mało przydatna i w takiej sytuacji sygnał ten może być zastąpiony przez doprowadzenie potencjału masy do nóżki 14 złącza wyświetlacza. Sterowanie generatorem ujemnego napięcia polaryzacji ekranu LCD jest



Rysunek 5. Propozycja wykonania płytki drukowanej dla sterownika: a) wzór płytki drukowanej, b) strona elementów, c) strona lutowania

przydatne podczas pracy nad programem. W momencie programowania mikrokontrolera, jego wszystkie układy wewnętrzne są zerowane i w efekcie przestają być generowane sygnały sterujące wyświetlaczem LCD. Grozi to uszkodzeniem wyświetlacza na skutek podania napięcia polaryzującego o niezerowej składowej stałej. Niebezpieczeństwo degradacji ciekłego kryształu jest tym większe, im dłużej trwa programowanie mikrokontrolera i tym samym przerwa w odświeżaniu wyświetlacza. Można temu zaradzić odłączając moduł wyświetlacza od układu na czas programowania mikrokontrolera, co jest kłopotliwe ze względu na dużą liczbę sygnałów sterujących. Inną metodą jest wyłączenie w module wyświetlacza generatora ujemnego napięcia polaryzacji ekranu LCD. W prezentowanym układzie sterownika LCD skorzystano z tej drugiej możliwości. Układ sterowania wykorzystuje fakt, iż podczas restartu mikrokontrolera linia PA2 (podobnie jak wszystkie inne porty I/O) jest konfigurowana jako wejście o wysokiej impedancji i ten tryb jest utrzymywany w trakcie programowania mikrokontrolera. W takich warunkach o stanie sygnału sterującego generatorem ujemnego napięcia polaryzacji LCD decyduje wewnętrzny układ pull-up w module EG9018C, który utrzymuje generator w stanie wyłączonym.

Realizacja praktyczna sterownika LCD

Testowy układ obsługujący wyświetlacz LCD Epson EG9018C wykonano na bazie zestawu uruchomieniowego STM32 Butterfly. Ponieważ moduł wyświetlacza EG9018C jest przystosowany do sterowania sygnałami o napięciu +5 V, a mikrokontroler STM32 pracuje z sygnałami o napięciu +3,3 V, do połączenia wyświetlacza z modułem STM32 Butterfly opracowano układ dopasowania poziomów sygnałów. Jego schemat ideowy pokazano na **rysunku 4**. Podstawowym zadaniem układu jest konwersja poziomów logicznych sygnałów z +3,3 V na +5 V. Tę funkcję realizują bramki OR i NOR z serii 74HCT. Rezystory R1...R15 na wejściach bramek wstępnie polaryzują wejścia na wypadek odłączenia od bramki sygnału sterującego.

Układ dopasowania poziomów sygnałów dodatkowo izoluje porty mikrokontrolera od wejść wyświetlacza. Co prawda, większość portów mikrokontrolera STM32F107 może pracować z sygnałami logicznymi o napięciu +5 V, a początkowe testy pokazały, że wyświetlacz EG9018C daje się sterować sygnałami o napięciu +3,3 V, jednak bezpośrednie połączenie wyświetlacza z mikrokontrolerem jest dość ryzykowne, ponieważ nie wszystkie wyprowadzenia układu STM32F107 są odporne na podanie napięcia przekraczającego wartość napięcia zasilania mikrokontrolera. Wydawać by się mogło, dla wyświetlacza EG9018C nie ma to większego znaczenia, ponieważ wszystkie jego linie są wejściami, jednak niektóre z nich mają własne rezystory pull-up podciągające wejście do napięcia +5 V. Groźba uszkodzenia mikrokontrolera przy omyłkowym podłączeniu takiego spolaryzowanego wejścia wyświetlacza do wyprowadzenia mikrokontrolera, które nie jest odporne na podanie sygnału o poziomie +5 V, jest jak najbardziej realna. Zastosowanie interfejsu pośredniczącego pomiędzy wyświetlaczem a zestawem STM32 Butterfly eliminuje to niebezpieczeństwo. Przy okazji płyta interfejsu oferuje bardziej przyjazne dla prototypowania złącze IDC w porównaniu z oryginalnym złączem ZIFF wyświetlacza.

Opracowany układ dopasowania poziomów sygnałów pozwala w razie potrzeby na rezygnację z programowego sterowania kontrastem i podświetleniem ekranu. Odpowiednie sygnały sterujące mogą zostać odłączone za pomocą zworek JP2 i JP3 od portów mikrokontrolera i na stałe ustawione w stanie aktywnym. Podobnie za pomocą zworki JP1 można załączyć manualna regulacja kontrastu wyświetlacza. W takiej konfiguracji kontrast jest ustawiany potencjometrem R16.

Przkład dopasowania poziomów sygnałów dla wyświetlacza EG9018C został zmon-

Listing 1. Funkcja konfigurująca układy mikrokontrolera STM32 obsługujące wyświetlacz EG9018C

```
void EG9018C_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;
  TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
  TIM_OCInitTypeDef TIM_OCInitStructure;
  TIM_BDTRInitTypeDef TIM_BDTRInitStructure;
  DMA_InitTypeDef DMA_InitStructure;

  //-------------------------------------------------------------
  //  Clock enable of used peripherals
  //  EG9018C uses DMA1, TIM1, TIM4, TIM5, GPIOA, GPIOD and AFIO
  //-------------------------------------------------------------
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4 | RCC_APB1Periph_TIM5, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOD | RCC_APB2Periph_AFIO | RCC_APB2Periph_TIM1,
ENABLE);

  //-------------------------------------------------------------
  //  Configuration of LCD data lines
  //  EG9018C data lines are connected to pins PD7..PD0
  //-------------------------------------------------------------
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7 | 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(EG9018C_DATA, &GPIO_InitStructure);
  //-------------------------------------------------------------
  //  Configuration of LCD data clock CP2 and BACKLIGHT on/off
  // EG9018C data clock CP2 is generated on pin PD12 by channel
  // 1 of timer TIM4. bBacklight on/off sigal is generated on pin
  // PD13 by channel 2 of timer TIM4
  ///-------------------------------------------------------------
  // PD12 configuration as TIM4_CH1 output, PD13 configuration as
  // TIM4_CH2 output
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12 | GPIO_Pin_13;
  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 (f_clk=72MHz)
  TIM_TimeBaseStructure.TIM_Period = EG9018C_CP2_PERIOD-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 Channel1 configuration: PWM1 Mode / 50%
  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
  TIM_OCInitStructure.TIM_Pulse = EG9018C_CP2_PERIOD/2;
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
```

towany na jednostronnej płytce drukowanej, wykonanej metodą termotransferu. Schemat montażowy układu wraz z widokiem mozaiki ścieżek obwodu drukowanego przedstawia **rysunek 5**. Montaż układu należy rozpocząć od wlutowania wszystkich mostków z drutu. Ze względu na niewielkie odległości pomiędzy poszczególnymi mostkami jest zalecane używanie przewodu izolowanego o średnicy 0,5 mm. Następnie należy zamontować od strony lutowania układy scalone w obudowach SMD i złącze ZIFF wyświetlacza, a na końcu wlutować wszystkie elementy przewlekane. W razie potrzeby potencjometr R16 do regulacji kontrastu wyświetlacza może zostać zamontowany poza płytką układu dopasowującego. Do tego celu służą punkty lutownicze oznaczone jako: *GND*, *CONT* i *+5 V*.

Zmontowany układ dopasowania poziomów sygnałów został zamocowany z tyłu wyświetlacza na trzech kołkach dystansowych przyklejonych do płytki przetwornic (**fotografia 6**). W efekcie uzyskano zwartą konstrukcję modułową, którą dołączono do zestawu uruchomieniowego STM32 Butterfly (**fotografia 7**). Połączenia pomiędzy modułem wyświetlacza EG9018C a zestawem STM32 Butterfly opisuje **tabela 2**.

Widoczny na zdjęciu zmontowanego zestawu (po prawej stronie) moduł STM32F0 Discovery służył podczas pracy nad programem jako programator/debugger SWD pełniąc funkcję usługową dla modułu STM32 Butterfly. Sam mikrokontroler STM32F051 z zestawu STM32F0 Discovery nie był użyty w projekcie sterownika wyświetlacza EG9018C ze względu na niewystarczające zasoby sprzętowe.

Urządzenie modelowe było zasilane za pośrednictwem modułu STM32F0 Discovery z portu USB komputera, na którym opracowywano oprogramowanie sterownika. Należy zwrócić uwagę, że przy załączonym podświetleniu wyświetlacza EG9018C pobór prądu wynosi ok. 1,125 A. W starszych komputerach PC może to powodować problemy, ponieważ typowo wydajność prądowa portu USB jest równa 0,5 A. W takim wypadku należy zasilać układ z zewnętrznego zasilacza +5 V o wystarczającej wydajności prądowej lub zrezygnować z używania podświetlania ekranu LCD podczas pracy nad programem.

Wykaz elementów

R1…R15: 47 kΩ/0,125W (przewlekany)
R16: 1 kΩ (potencjometr nastawny)
R17, R18: 0 Ω lub zworka z drutu
C1…C4: 100 nF (przewlekany, raster 5,0 mm)
C5: 100 µF/6,3 V (przewlekany 04U)
U1…U3: 74HCT32D (SMD)
U4: 74HCT02D (SMD)
J1: gniazdo IDC34, pionowe
J2: gniazdo ZIFF-1,0mm-30pin
JP1…JP3: listwa goldpin 1×3, raster 2,54 mm

Listing 1. c.d.

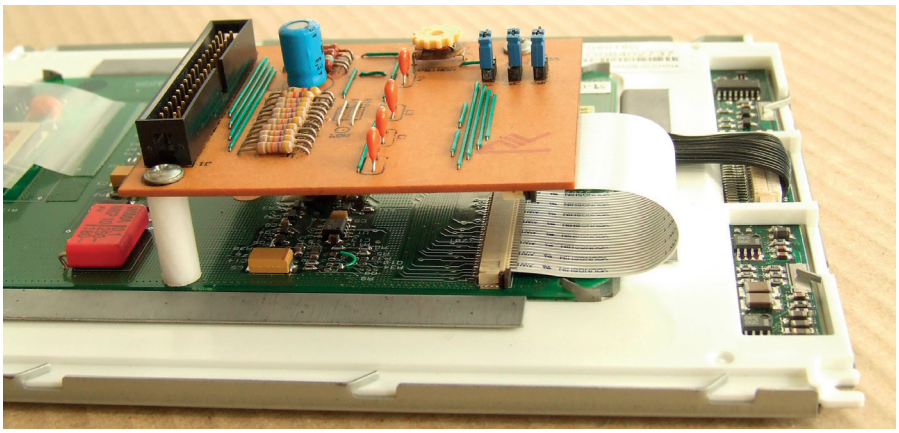
```
TIM_OC1Init(TIM4, &TIM_OCInitStructure);
TIM_OC1PreloadConfig(TIM4, TIM_OCPreload_Enable);
// TIM4 Channel2 configuration: PWM1 Mode / 0% or 100%
TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
TIM_OCInitStructure.TIM_Pulse = EG9018C_BKLT_OFF;
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
TIM_OC2Init(TIM4, &TIM_OCInitStructure);
TIM_OC2PreloadConfig(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 CP1 and CONTRAST adjustment
// EG9018C data latch CP1 is generated on pin PA0 by channel 1
// of timer TIM5. Contrast adjustment voltage is generated on
// pin PA1 by channel 2 of timer TIM5
//-------------------------------------------------------------
// PA0 configuration as TIM5_CH1 output, PA1 configuration as
// TIM5_CH2 output
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | 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: TIM5 is the slave of TIM4
TIM_SelectSlaveMode(TIM5, TIM_SlaveMode_External1);
TIM_SelectInputTrigger(TIM5, TIM_TS_ITR2);
// TIM5 time base configuration (f_clk=fTIM4)
TIM_TimeBaseStructure.TIM_Period = EG9018C_CP1_PERIOD-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 = EG9018C_CP1_PERIOD-EG9018C_CP1_LEN;
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);
// TIM5 Channel2 configuration: PWM1 Mode
TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
TIM_OCInitStructure.TIM_Pulse = EG9018C_CNTR_ADJ;
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
TIM_OC2Init(TIM5, &TIM_OCInitStructure);
TIM_OC2PreloadConfig(TIM5, TIM_OCPreload_Enable);
//-------------------------------------------------------------
// Configuration of LCD start-up S
// EG9018C start-up S is generated on pin PA8 by channel 1
// of timer TIM1. Start-up pulse delay is provided by TIM1
// OCx output dead time generator
/// -----------------------------------------------------------
// PA8 configuration as TIM1_CH1 output
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);
// Clock source configuration: TIM1 is the slave of TIM4
TIM_SelectSlaveMode(TIM1, TIM_SlaveMode_External1);
TIM_SelectInputTrigger(TIM1, TIM_TS_ITR3);
// TIM1 time base configuration (f_clk=fTIM4)
TIM_TimeBaseStructure.TIM_Period = EG9018C_S_PERIOD-1;
TIM_TimeBaseStructure.TIM_Prescaler = 0;
TIM_TimeBaseStructure.TIM_ClockDivision = 0;
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseStructure.TIM_RepetitionCounter = 0;
TIM_TimeBaseInit(TIM1, &TIM_TimeBaseStructure);
TIM_ARRPreloadConfig(TIM1, ENABLE);
// TIM1 Channel1 configuration: PWM1 Mode
TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
TIM_OCInitStructure.TIM_Pulse = EG9018C_S_LEN;
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
TIM_OCInitStructure.TIM_OCIdleState = TIM_OCIdleState_Reset;
TIM_OCInitStructure.TIM_OutputNState = TIM_OutputNState_Enable;
TIM_OCInitStructure.TIM_OCNPolarity = TIM_OCNPolarity_High;
TIM_OCInitStructure.TIM_OCNIdleState = TIM_OCNIdleState_Reset;
TIM_OC1Init(TIM1, &TIM_OCInitStructure);
TIM_OC1PreloadConfig(TIM1, TIM_OCPreload_Enable);
// TIM1 Dead Time
TIM_BDTRInitStructure.TIM_DeadTime = 0xFF;
TIM_BDTRInitStructure.TIM_AutomaticOutput = TIM_AutomaticOutput_Disable;
TIM_BDTRInitStructure.TIM_Break = TIM_Break_Disable;
TIM_BDTRInitStructure.TIM_BreakPolarity = TIM_BreakPolarity_Low;
TIM_BDTRInitStructure.TIM_LOCKLevel = TIM_LOCKLevel_OFF;
TIM_BDTRInitStructure.TIM_OSSIState = TIM_OSSIState_Disable;
TIM_BDTRInitStructure.TIM_OSSRState = TIM_OSSRState_Disable;
TIM_BDTRConfig (TIM1,&TIM_BDTRInitStructure);
// TIM1 Main Outputs Enable
TIM_CtrlPWMOutputs(TIM1, ENABLE);


//-------------------------------------------------------------
// Configuration of LCD data transfer
// EG9018C data is transfered to pins PD7..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);
```

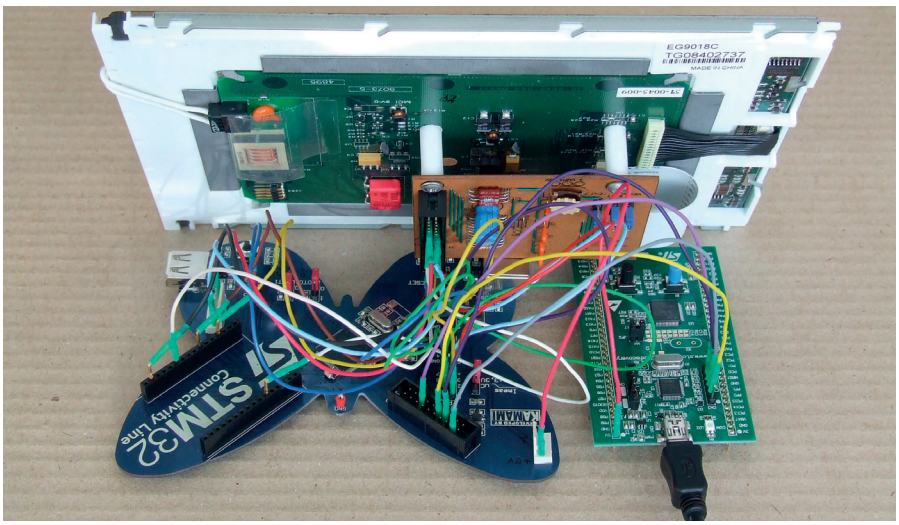
Na CD: karty katalogowe i noty aplikacyjne elementów oznaczonych w wykazie elementów kolorem czerwonym

Listing 1. c.d.

```
  DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&(EG9018C_DATA->ODR);
  DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)&EG9018C_frame_buffer[0]
[0];
  DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST;
  DMA_InitStructure.DMA_BufferSize = (EG9018C_XRES/4)*(EG9018C_YRES/2);
  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 to start DMA transfer
  TIM_DMACmd(TIM4, TIM_DMA_CC1, ENABLE);
  //-------------------------------------------------------------
  // CONTRAST VOLTAGE GENERATOR enable
  // EG9018C contrast voltage generator is enabled by high level on
  // pin PA2
  //-------------------------------------------------------------
#if (EG9018C_CNTR_GEN_SWD == 1)
  // PA2 configuration as output
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  // enable EG9018C contrast voltage generator
  GPIO_SetBits(GPIOA, GPIO_Pin_2);
#endif
  //-------------------------------------------------------------
  // Start servicing LCD
  //-------------------------------------------------------------
  // Request the first DMA transfer
  TIM_GenerateEvent(TIM4, TIM_EventSource_CC1);
  // Enable TIM1 counter
  TIM_Cmd(TIM1, ENABLE);
  // Enable TIM5 counter
  TIM_Cmd(TIM5, ENABLE);
  // Enable TIM4  counter
  TIM_Cmd(TIM4, ENABLE);
}
```



Fotografia 6. Sposób montażu układu dopasowania poziomów logicznych sygnałów w wyświetlaczu EG9018C



Fotografia 7. Układ testowy, który posłużył do opracowania sterownika wyświetlacza EG9018C

Oprogramowanie sterownika

Źródło funkcji konfigurującej układy peryferyjne mikrokontrolera STM32F107 do obsługi wyświetlacza EG9018C zamieszczono na **listingu 1**. Programowanie układów rozpoczyna się od załączenia sygnału zegarowego dla wszystkich peryferii używanych przez sterownik LCD. Następnie konfigurowane są linie 0…7 portu D, którymi będą przesyłane dane do wyświetlacza. Pracują one w trybie wyjść typu push-pull o minimalnym czasie narastania sygnału.

Kolejną czynnością wykonywaną przez funkcję *EG9018C_Init* jest konfigurowanie generatora sygnału zegara CP2. Przy okazji jest też konfigurowane wyjście sterowania podświetleniem ekranu. Najpierw sprogramowane są linie wyjściowe tych sygnałów, tj. PD12 i PD13. Pracują one jako wyjścia push-pull i pełnią funkcje alternatywne. Ponieważ te linie nie są domyślnymi wyprowadzeniami układu czasowo-licznikowego TIM4, jest konieczne dodatkowe przekrosowanie sygnałów TIM4 na porty PD12 i PD13, co realizuje komenda *GPIO_PinRemapConfig(GPIO_Remap_TIM4, ENABLE)*.

Układ czasowo-licznikowy TIM4 jest pracuje w trybie licznika odmierzającego okres sygnału zegara CP2 wyświetlacza LCD. Jako źródło sygnału taktującego dla licznika TIM4 jest wybierany niepodzielony sygnał zegarowy magistrali APB1 (*TIM_TimeBaseStructure.TIM_Prescaler = 0* i *TIM_TimeBaseStructure.TIM_ClockDivision* = 0). Odpowiedzialne za generowanie sygnałów CP2 i sterowania podświetleniem kanały 1 i 2 układu czasowo-licznikowego TIM4 są konfigurowane do pracy w trybie PWM1. Na zakończenie konfiguracji generatora sygnału CP2, zdarzenie przeładowania licznika TIM4 jest ustawiane jako źródło sygnału taktującego dla liczników slave (komendy *TIM_SelectMasterSlaveMode(TIM4, TIM_MasterSlaveMode_Enable)* i *TIM_SelectOutputTrigger(TIM4, TIM_TRGOSource_Update)*).

Po skonfigurowaniu generatora sygnału zegara CP2, funkcja EG9018C_Init konfiguruje generator sygnału CP1 oraz wyjście

sterowania kontrastem wyświetlacza LCD. Podobnie jak poprzednio, najpierw programowane są linie wyjściowe sygnałów CP1 i VEEreg, tj. PA0 i PA1. Pracują one w trybie wyjść push-pull i pełnią funkcje alternatywne. Ponieważ są one domyślnymi wyprowadzeniami układu czasowo-licznikowego TIM5, na tym kończy się ich konfigurowanie.

W odróżnieniu od timera TIM4, układ czasowo-licznikowy TIM5 nie jest taktowany sygnałem zegarowym uzyskanym z magistrali APB1. Z tego też względu układ

Listing 2. Funkcje realizujące zapalanie i gaszenie piksela

```
/*-------------------------------------------------------
  Name         :  LCD_PixelOn
  Description  :  Function draws pixel
  Argument(s)  :  Xpos -> pixel x coordinate (0..639)
                  Ypos -> pixel y coordinate (0..479)
  Return value :  none
-------------------------------------------------------*/
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 < EG9018C_XRES && Ypos < EG9018C_YRES){
// Calculate pixel position in LCD frame buffer
    colNo = Xpos/4;
    rowNo = (Ypos < EG9018C_YRES/2) ? Ypos : Ypos - EG9018C_YRES/2;
    bitNo = (Ypos < EG9018C_YRES/2) ? 7 - Xpos%4 : 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)&EG9018C_frame_buffer[rowNo][colNo]-SRAM_BASE)<<5)|((bitNo)<<2)) = LCD_
PIXEL_ON;
  }
}

/**-----------------------------------------------------
  Name         :  LCD_PixelOff
  Description  :  Function clears pixel
  Argument(s)  :  Xpos -> pixel x coordinate (0..639)
                  Ypos -> pixel y coordinate (0..479)
  Return value :  none
-------------------------------------------------------*/
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 < EG9018C_XRES && Ypos < EG9018C_YRES){
    // Calculate pixel position in LCD frame buffer
    colNo = Xpos/4;
    rowNo = (Ypos < EG9018C_YRES/2) ? Ypos : Ypos - EG9018C_YRES/2;
    bitNo = (Ypos < EG9018C_YRES/2) ? 7 - Xpos%4 : 3 - Xpos%4;
    // Clear pixel
    // (acces to pixel bit is performed via bit-banding region of SRAM)
    *(__IO uint32_t*)(SRAM_BB_BASE|(((uint32_t)&EG9018C_frame_buffer[rowNo][colNo]-SRAM_BASE)<<5)|((bitNo)<<2)) = LCD_
PIXEL_OFF;
  }
}
```

Tabela 3. Funkcje modułu LCD_EG9018C

Lp	Nazwa funkcji	Argumenty	Opis
1	EG9018C_Init	brak	Inicjalizacja sterownika LCD
2	LCD_Contrast	uint8_t contrast	Regulacja kontrastu LCD
3	LCD_Backlight	uint8_t bkltState	Sterowanie podświetleniem ekranu LCD (podświetlenie wyłączone, gdy bkltState = EG9018C_BKLT_OFF, podświetlenie załączone gdy bkltState = EG9018C_BKLT_ON)
4	LCD_PixelOn	uint16_t Xpos uint16_t Ypos	Zapalenie piksela o współrzędnych (Xpos, Ypos)
5	LCD_PixelOff	uint16_t Xpos uint16_t Ypos	Zgaszenie piksela o współrzędnych (Xpos, Ypos)
6	LCD_Clear	brak	Wyczyszczenie powierzchni ekranu
7	LCD_DrawLine	uint16_t Xpos uint16_t Ypos uint16_t length uint8_t direction	Rysowanie poziomego (direction = LCD_HORIZONTAL) lub pionowego (direction = LCD_VERTICAL) odcinka o początku w punkcie o współrzędnych (Xpos, Ypos) i długości length
8	LCD_DrawRect	uint16_t Xpos uint16_t Ypos uint16_t width uint16_t height	Rysowanie prostokąta o wymiarach width x height, którego lewy górny róg ma współrzędne (Xpos, Ypos)
9	LCD_DrawCircle	uint16_t Xpos uint16_t Ypos uint16_t radius	Rysowanie okręgu o środku położonym w punkcie o współrzędnych (Xpos, Ypos) i promieniu radius
10	LCD_DrawPicture	uint16_t Xpos uint16_t Ypos uint16_t Xsize uint16_t Ysize uint8_t *p_pict	Wyświetlenie mapy bitowej o rozmiarze Xsize x Ysize pikseli, której lewy górny róg ma współrzędne (Xpos, Ypos)
11	LCD_DrawText	uint16_t Xpos uint16_t Ypos uint8_t font uint8_t *p_text	Wyświetlenie tekstu, którego lewy górny róg ma współrzędne (Xpos, Ypos) z użyciem podanego zestawu czcionek font. W module dostępne są czcionki: FONT_FIVE_DOT, FONT_SIX_DOT, FONT_SEVEN_DOT, FONT_NINE_DOT, FONT_TEN_DOT, FONT_FIFTEEN_DOT, FONT_EIGHTEEN_DOT, FONT_FONT_MT1

TIM5 pracuje w trybie slave (komenda *TIM_SelectSlaveMode(TIM5, TIM_SlaveMode_External1)*) ze wskazaniem źródła sygnału taktującego na linię TIM_TS_ITR2 (komenda *TIM_SelectInputTrigger (TIM5, TIM_TS_ITR2)*). W mikrokontrolerach STM32F1 ta linia jest przypisana w układzie TIM5 do doprowadzania sygnałów taktujących z układu TIM4. Pozostała część programu źródłowego konfigurującego generator sygnału CP1 jest analogiczna do konfigurującego generator sygnału CP2.

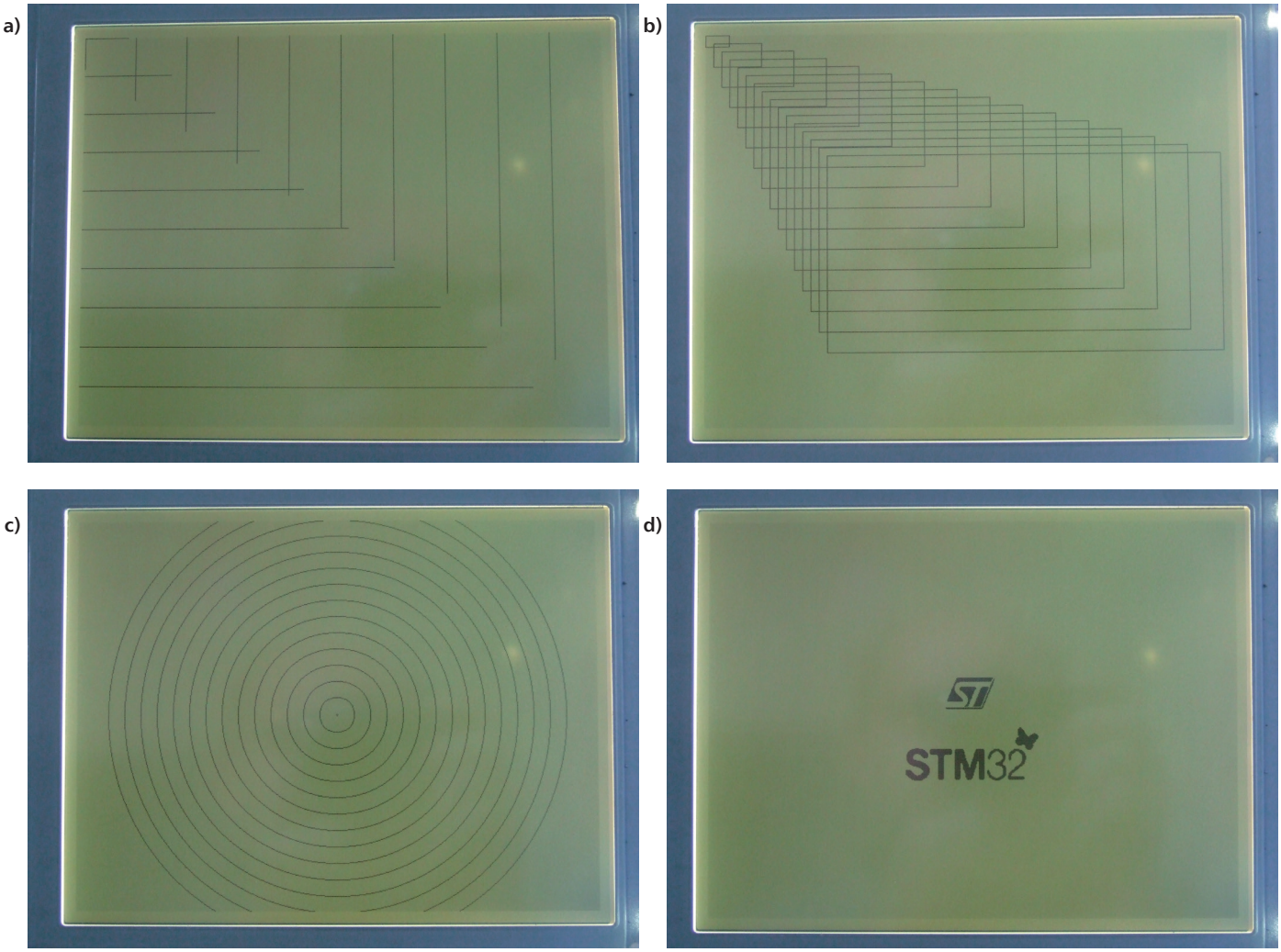
Następnym krokiem wykonywanym przez funkcję *EG9018C_Init*, jest konfigurowanie generatora sygnału S. Ponieważ ten generator jest realizowany w oparciu o układ czasowo-licznikowy TIM1, który jest bardziej złożony od układów TIM4 i TIM5, program wykonujący jego konfigurowanie jest bardziej złożony. Rozpoczyna się on od skonfigurowania linii wyjściowej sygnału S, identycznie jak w wypadku generatorów sygnałów CP2 i CP1. Linia 8 portu PA pracuje w trybie wyjścia typu push-pull i pełni funkcję alternatywną.

Podobnie jak w wypadku generatora sygnału CP1, źródłem taktowania układu czasowo-licznikowego TIM1 jest układ TIM4. W tym celu układ TIM1 pracuje w trybie slave (komenda *TIM_SelectSlaveMode(TIM1, TIM_SlaveMode_External1)*), ze wskazaniem źródła sygnału taktującego na linię TIM_TS_ITR3 (komenda *TIM_SelectInputTrigger(TIM1, TIM_TS_ITR3)*). Ta linia jest w mikrokontrolerach STM32F1 przypisana w układzie TIM1 do doprowadzania sygnałów taktujących z układu TIM4. Jak w poprzednio, układ czasowo-licznikowy TIM1 pracuje w trybie licznika odmierzającego okres sygnału S, a jego kanał pierwszy, fizycznie generujący sygnał, jest konfigurowany do pracy w trybie PWM1.

W przeciwieństwie do poprzednio konfigurowanych układów TIM4 i TIM5, kanał pierwszy układu TIM1 ma wyjścia komplementarne. W sterowniku LCD oba wyjścia kanału pierwszego są aktywowane (komendy *TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable* i *TIM_OCInitStructure.TIM_OutputNState = TIM_OutputNState_Enable*), mimo iż sygnał S jest pobierany wyłącznie z wyjścia *TIM1_CH1*. Jest to konieczne, ponieważ tylko przy załączeniu obu wyjść komplementarnych działa generator czasu martwego. Ponieważ odpowiadający nieużywanemu wyjściu *TIM1_CH1N* port PB13 nie jest jednocześnie programowany jako wyjście sygnału funkcji alternatywnej i nadal może być używany w programie użytkownika jako wejście lub wyjście ogólnego przeznaczenia.

Realizujący opóźnienie przedniego zbocza sygnału S generator czasu martwego timera TIM1 jest programowany na maksymalną dopuszczalną wartość opóźnienia (komenda *TIM_BDTRInitStructure.TIM_DeadTime = 0xFF*). Sama wartość uzyskiwanego opóźnienia nie jest w tym wypadku krytyczna. Chodzi tylko o to, aby narastające zbocze sygnału S następowało po opadającym zboczu sygnału CP1. Jeżeli warunek ten nie będzie spełniony, to linie nr 240 i 480 mogą zostać potraktowane jako linie początku pola obrazu, co będzie się objawiało jego rozmyciem w pionie. Aktywacja wyjść PWM układu TIM1 (komenda *TIM_CtrlPWMOutputs(TIM1, ENABLE)*) kończy programowanie generatora sygnału S.

Mając zakończone konfigurowanie linii wyjściowych oraz generatorów sygnałów CP2, CP1 i S, funkcja *EG9018C_Init* konfiguruje układ DMA, który będzie realizował przesyłanie danych z bufora obrazu do rejestru wyjściowego *ODR* portu D. Dokładnie będzie to kanał pierwszy układu DMA1, ponieważ w mikrokontrolerze STM32F107 właśnie ten kanał jest przeznaczony do ob-



Fotografia 8. Efekty działania funkcji rysowania: a) odcinków, b) prostokątów, c) okręgów, d) map bitowych

sługi zdarzeń z kanału pierwszego układu TIM4. Układ DMA jest programowany do pracy w trybie transferu danych z pamięci do układu peryferyjnego (komenda *DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST*). Kontroler DMA będzie pobierał dane jako słowa o długości 1 B z bufora obrazu zorganizowanego w pamięci RAM, tj. tablicy *EG9018C_frame_buffer* (komenda *DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)&EG9018C_frame_buffer[0][0]*) i zapisywał je do rejestru wyjściowego *ODR* portu D (komenda *DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&(EG9018C_DATA->ODR)*). Po każdym transferze adres danej do pobrania z pamięci RAM będzie inkrementowany. Układ DMA będzie pracował w trybie kołowym, dzięki czemu po przesłaniu zawartości całego bufora obrazu, cały cykl pracy rozpocznie się automatycznie od początku. Każdy transfer DMA będzie inicjowany przez zdarzenie porównania, które będzie generowane przez kanał pierwszy układu TIM4 (komenda *TIM_DMACmd(TIM4, TIM_DMA_CC1, ENABLE)*), a więc w momencie wystąpienia opadającego zbocza sygnału zegara CP2.

Następny punkt wykonywany przez funkcję EG9018C_Init to konfiguracja wyjścia PA2 załączającego w module wyświetlacza EG9018C generator ujemnego napięcia polaryzacji ekranu LCD. Punkt ten jest wykonywany opcjonalnie, zależnie od wybranej opcji kompilacji programu. Steruje nią definicja *EG9018C_CNTR_GEN_SWD* w pliku nagłówkowym *LCD_EG9018C.h*.

Po zakończeniu konfiguracji układów peryferyjnych pozostaje jedynie wygenerowanie pierwszego żądania transferu dla układu DMA (*komenda TIM_GenerateEvent(TIM4, TIM_EventSource_CC1)*) oraz włączenie układów TIM1, TIM5 i TIM4. Od tego momentu skonfigurowany układ sterownika LCD rozpoczyna pracę i nie wymaga już żadnej uwagi ze strony jednostki centralnej.

Obsługa wyświetlacza

Mając uruchomiony sterownik LCD, dalsza obsługa wyświetlacza EG9018C ogranicza się do operacji wykonywanych na pamięci RAM bufora obrazu. Zapalenie lub zgaszenie punktu na wyświetlaczu sprowadza się do ustawienia lub wyzerowania odpowiedniego bitu w pamięci bufora obrazu. W przypadku mikrokontrolera STM32 operacje te najwygodniej wykonywać korzystając z dostępu atomowego do pamięci. Kody obu funkcji realizujących załączanie i wyłączanie punktu pokazano na **listingu 2**. W każdej z funkcji przed wykonaniem operacji ustawiania/kasowania bitu jest przeprowadzana kontrola współrzędnych punktu, dzięki czemu nie ma niebezpieczeństwa „mazania" przez funkcję po pamięci RAM, co mogłoby się zdarzyć w wypadku podania parametrów wejściowych wykraczających poza obszar bufora obrazu. W tym miejscu kontrola zakresu zmiennych upraszcza też pisanie funkcji graficznych wyższego poziomu, ponieważ odpada w nich konieczność kontrolowania, czy rysowany element nie wychodzi poza obszar pamięci bufora obrazu.

W oparciu o funkcje włączania i wyłączania pojedynczego piksela zostały stworzone funkcje rysowania odcinków, prostokątów, okręgów, wyświetlania map bitowych oraz wyświetlania tekstów. Funkcje te zebrane są w module programowym LCD_EG9018C (**tabela 3**), który jest dołączony do artykułu. Rezultaty ich działania pokazano na **fotografiach 8** i **9**.

Fotografia 9. Efekt działania funkcji wyświetlania tekstów

Szerszego omówienia wymaga funkcja wyświetlania tekstów LCD_DrawText(). Standardowo implementowane w oprogramowaniu wyświetlaczy o niskiej rozdzielczości i małych wymiarach ekranu czcionki w formacie 7x5 pikseli są w przypadku wyświetlacza EG9018C mało przydatne, ze względu na ich słabą czytelność, o czym można się łatwo przekonać spoglądając na fotografię 5. Z tego też względu funkcja LCD_DrawText() umożliwia stosowanie czcionek o definiowanych wymiarach. Aby nie tworzyć własnych tablic z kształtami czcionek, w projekcie sterownika LCD wykorzystano moduł programowy fonts.c, który został udostępniony w sieci Internet przez Martina Thomasa [3]. Moduł ten obejmuje 8 zestawów czcionek, poczynając od wspomnianych wcześniej standardowych znaków w rastrze 7×5 pikseli, a kończąc na czcionkach o wysokości 40 pikseli. Wszystkie zestawy czcionek są pokazane na fotografii 5. Należy zauważyć, iż zestawy dwóch największych czcionek w module fonts.c (tj. czcionki *FONT_EIGHTEEN_DOT*, *FONT_FONT_MT1*) są niepełne. Mają one zdefiniowane tylko cyfry oraz niektóre znaki przestankowe i niektóre symbole działań matematycznych. Tym niemniej, czcionki o takiej wielkości mogą okazać się przydatne w tworzonych aplikacjach. W razie potrzeby deklaracje w pliku nagłówkowym *fonts.h* pozwalają na wyłączenie nieużywanych zestawów czcionek, co pozwala zaoszczędzić pamięć programu mikrokontrolera.

Podsumowanie

Opisany w artykule sterownik LCD został opracowany z myślą o podłączeniu wyświetlacza EG9018C firmy EPSON do mikrokontrolera SMT32, jednak zastosowanie przedstawionego sterownika nie ogranicza się tylko do tego jednego typu wyświetlacza i można go łatwo przystosować do obsługi innych typów wyświetlaczy graficznych LCD, które nie mają wbudowanego układu kontrolera. Takie wyświetlacze mają zazwyczaj zbliżony sposób sterowania do użytego w projekcie modułu EG9018C. Różnice występują jedynie w szerokości szyny danych, w liczbie impulsów zegara przypadających na jedną linię oraz w liczbie zapisywanych linii przypadających na pole obrazu, co bezpośrednio wynika z rozdzielczości wyświetlacza. Na przykład, dla typowego wyświetlacza monochromatycznego LCD o rozdzielczości 1/4 VGA (tj. 320×240 punktów) szerokość szyny danych wynosi 4 bity, liczba impulsów zegara przypadających na jedną linię jest równa 80, zaś liczba linii w polu obrazu jest równa 240. Aby móc obsługiwać taki wyświetlacz, w opisanym sterowniku LCD należy zmodyfikować definicje stałych odpowiadających za powyższe parametry. W razie potrzeby można też zmienić częstotliwość odświeżania wyświetlacza, dostosowując ją do potrzeb obsługiwanego modułu. Odbywa się to poprzez zmianę okresu zliczania timera TIM4.

Aleksander Borysiuk
alex_priv@wp.pl

Bibliografia

[1] Specification for Passive Matrix LCD Unit, Model No. LM64183P, Sharp Corporation

[2] Wojciech Kaczmarski, Biblioteka C dla ARM obsługująca wyświetlacz monochromatyczny 8.4 EPSON EG9018C o rozdzielczości 640×480px, http://xtal.tk/?p=1038

[3] Martin Thomas, ARM-Projects by Martin Thomas. "T"-Clock: DCF77 radio-clock-receiver with Graphics-LCD display for LPC2106 (ARM7TDMI), http://www.siwawi.arubi.uni-kl.de/avr_projects/arm_projects/glcd_dcf77/index.html