Przetwornik obrazu w systemie z STM32, czyli jak dołączyć kamerę do mikrokontrolera

Miniaturowe przetworniki obrazu są powszechnie stosowane w sprzęcie elektronicznym. Począwszy od zabawek przez telefony komórkowe, komputery przenośne, tablety po urządzenia automatyki przemysłowej. Bywają także dostępne jako moduły, które można zastosować w systemie z mikrokontrolerem wyposażonym w kilkadziesiąt kilobajtów pamięci RAM i trochę wolnych linii portów IO.

Poco stosować przetworniki obrazu?

Pierwsza odpowiedź jest oczywista: bo to ciekawe i możliwe do wykonania. Bo obrazek jako efekt działania układu elektronicznego jest atrakcyjny i w dodatku niesie sporo informacji. Nie tylko estetycznych. Obrazu można użyć jako źródła danych dla sterowników, detektorów ruchu, przy pomiarach obiektów, detekcji zdarzeń itp. Możliwości jest sporo, jednak aby coś z obrazem zrobić, należy najpierw go zarejestrować. Można w tym celu posłużyć się telefonem komórkowym czy kamerką laptopa, jednak w systemie z mikrokontrolerem najwygodniej mieć do dyspozycji zintegrowany przetwornik obrazu, którym można sterować i kontrolować jego ustawienia w zależności od potrzeb.

Podstawowe właściwości

System-On-A-Chip (SOC) CMOS Digital Image Sensor – pod taką nazwą kryją się interesujące nas elementy. Przetworniki obrazu mogą być wykonane w formie układów scalonych współpracujących z zewnętrznym systemem optycznym czyli obiektywem. Takie przetworniki stosowane są w popularnym sprzęcie, takim jak aparaty cyfrowe czy kamery. Przetworniki mogą być także zintegrowane z systemem optycznym w jednej kostce, z wyprowadzonymi złączami sygnałowymi. Są to elementy o małych gabarytach, ale zwykle gorszych parametrach, niż układy z zewnętrzną optyką. Zintegrowane przetworniki obrazu są stosowane w telefonach komórkowych, jako kamerki w laptopach, tabletach, zabawkach itp.

Drugim ważnym rozróżnieniem jest format danych wyjściowych, które można odczytać z przetwornika. Może to być strumień prawie surowych danych obrazu zawierający informacje o stanie pojedynczych pikseli albo obraz zakodowany i skompresowany np. w formacie JPG. Obrazy zakodowane są mniejsze objętościowo i łatwiejsze do przesłania. Jednak dane bez kodowania są wygodniejsze do wykorzystania przez układy sterowników i detektorów, gdyż przyśpieszają dostęp do konkretnych obszarów obrazu i ułatwiają śledzenie zmian, co jest istotne np. przy detekcji ruchu.

Można jeszcze dokonać kilku podziałów między innymi ze względu na szczegóły budowy, matrycę, cenę ale jednym z najważniejszych jest dostępność lub brak dokumentacji technicznej. Przetwornik o najciekawszych możliwościach, lecz bez precyzyjnej dokumentacji zawierającej opisy wewnętrznych rejestrów i sposobu sterowania, jest praktycznie nieprzydatny. Niestety, to sytuacja typowa w wypadku większości kamerek stosowanych w telefonach komórkowych.

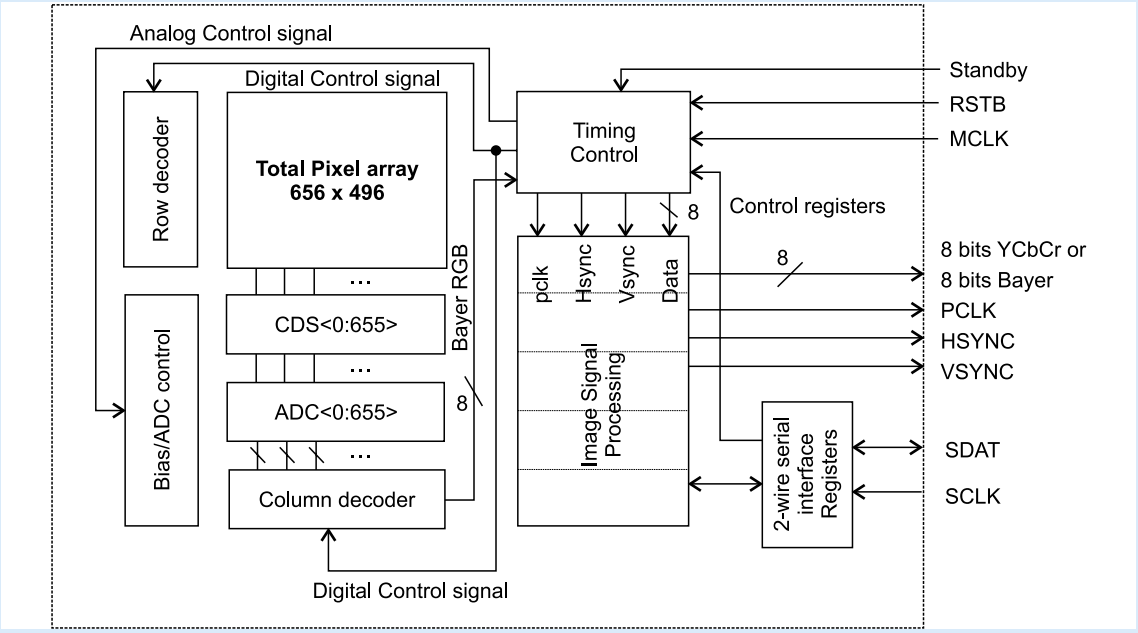
Ten opis zastosowania przetwornika obrazu w systemie mikroprocesorowym jest oparty na elemencie PPV401 z wewnętrznym sterownikiem PO6030 wyprodukowanym przez firmę Pixelplus. To względnie prosty w działaniu element dostarczający niekodowany strumień danych obrazowych. Można go zastosować w systemie z mikrokontrolerem o przeciętnych możliwościach. Wadą elementu jest jego trudna dostępność, ponieważ dystrybutor krajowy wycofał się z jego sprzedaży. Jednak sposób działania przetwornika jest bardzo podobny do innych układów dostępnych na rynku. Znając ogólne zasady łatwiej szukać kluczowych danych w dokumentacji bardziej dostępnych przetworników np. firm Aptina czy OmniVision.

Ogólnie o budowie

Pragnąc wykorzystać w swoim urządzeniu przetwornik obrazu warto znać jego ogólną budowę żeby wiedzieć jakie dane można z niego uzyskać oraz jak nim sterować.

PPV401 jest nieskomplikowanym przetwornikiem dostarczającego ciągłego strumienia niekodowanej informacji o kolejnych pikselach obrazu. Ma rozdzielone magistrale dla danych obrazu i sygnałów sterujących oraz kilka dodatkowych linii sygnałów synchronizujących. Jego schemat blokowy pokazano na **rysunku 1**. Zanim przejdziemy do bardziej szczegółowego opisu sygnałów, najpierw kilka słów o budowie matrycy rejestrującej przetwarzającej obraz optyczny, ponieważ sposób jej funkcjonowania wpływa na budowę przetwornika, jego właściwości i sterowanie.

Matryca składa się z mozaiki miniaturowych elementów światłoczułych rejestrujących informacje o natęże-



Rysunek 1. Schemat blokowy przetwornika PPV401

niu światła pojedynczych punktów (pikseli) składających się na przetwarzany obraz. Dla uzyskania informacji o kolorze przed światłoczułymi elementami umieszczone są kolorowe filtry dla trzech podstawowych barw na które reaguje ludzkie oko: czerwonej, zielonej, niebieskiej czyli w skrócie RGB. Na rysunku 2 przedstawiona została kolorowa szachownica stanowiąca wycinek filtru pokrywającego całą aktywną przestrzeń matrycy przetwornika. Tego typu filtr nosi nazwę Bayer i jest często spotykanym rozwiązaniem w wielu typach przetworników. Budowa matrycy powoduje, że pojedynczy element światłoczuły dla jednego koloru podstawowego obsługuje kilka sąsiednich „wirtualnych" pikseli obrazu. Drobne oszustwo nieznacznie pogarsza jakość obrazu, jednak istotną korzyścią jest redukcja liczby elementów światłoczułych potrzebnych do konwersji o założonej rozdzielczości. Wewnętrzny sterownik przetwarza informacje z sąsiednich detektorów kolorów podstawowych i na wyjściu przetwornika użytkownik otrzymuje dane w taki sposób, jakby każdy piksel obrazu składał się z pojedynczego elementu RGB. Po uformowaniu dane są wysyłane bezpośrednio przez równoległą 8-bitową magistralę wyjściową. W innych przetwornikach w zależności od formatu danych wyjściowych i precyzji przetwarzania składowych koloru rozmiar magistrali może być inny np. 12-bitowy lub 16-bitowy. Z reguły jednak wszędzie będzie to oddzielna równoległa magistrala wyjściowa dla szybkiego przesyłu dużych bloku danych o obrazie.

R	G	R	G	R	G
G	B	G	B	G	B
R	G	R	G	R	G
G	B	G	B	G	B
R	G	R	G	R	G
G	B	G	B	G	B

Rysunek 2. Wycinek filtra pokrywającego matrycę

Użytkownik aby móc wykorzystać ten nieprzerwany strumień danych potrzebuje jeszcze sygnałów synchronizujących. Sygnały te informują o przesyłaniu danych treści obrazu. W przetworniku PPV401 są to: VSYNC o poziomie aktywnym w czasie gdy wysyłane są dane kolejnych linii tworzących obraz, HSYNC o poziomie aktywnym w czasie wysyłania pojedynczej linii obrazu, PCLK taktujący pojawienie się na magistrali danych kolejnego piksela. Te trzy sygnały w zupełności wystarczą do prawidłowego odebrania danych obrazu z przetwornika. Takie sygnały, być może oznaczone nieco innymi nazwami będą obecne w każdym typowym przetworniku.

Osobną magistralą najczęściej szeregową przesyłane są dane przeznaczone do zapisu i odczytywane z wewnętrznych rejestrów sterujących przetwornika. Rozdzielenie magistrali ma sens, ponieważ po stronie użytkownika upraszcza budowę interfejsów pomiędzy przetwornikiem a mikrokontrolerem. Zastosowanie magistrali szeregowej do przesyłu danych sterujących pozwala zredukować liczbę niezbędnych linii, a ograniczona szybkość przepływu komend nie ma wielkiego znaczenia. W przetworniku PPV401 magistrala sterująca działa w formacie bardzo podobnym do standardowego I²C. Występują bity *Startu* i *Stopu*, urządzenie jest adresowane DCh (11011100b) dla zapisu i DDh (11011101b) przy odczycie z rejestrów. Przesyłane po sekwencji *Start* adresy rejestrów określają punkt początkowy dla pojedynczego lub grupowego zapisu lub odczytu.

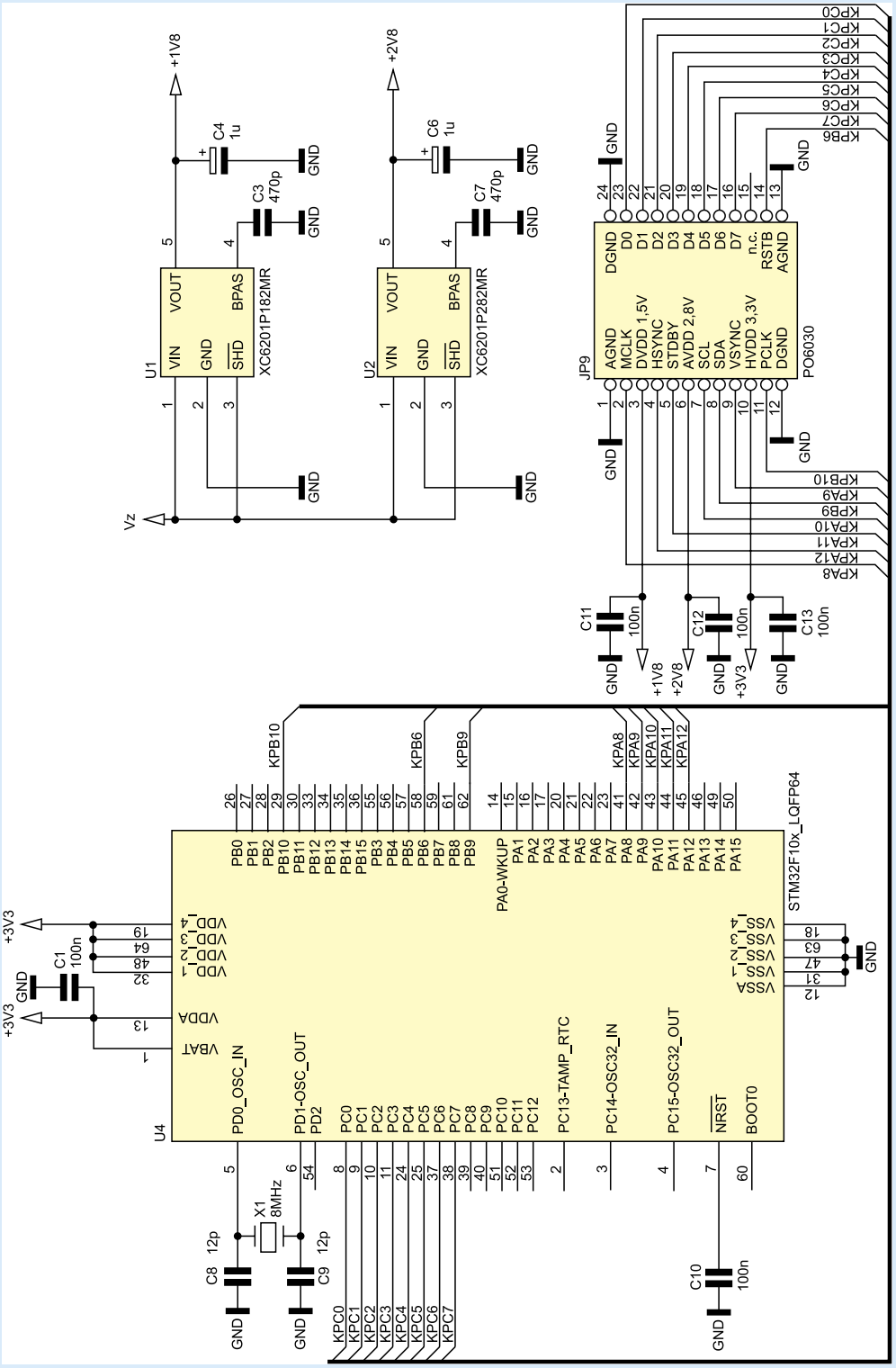
Podobną magistralę można spotkać w innych przetwornikach z tym, że może ona pracować w innym formacie np. SPI. W dokumentacji magistrala sterująca i sposób dotarcia do rejestrów powinny być dokładnie opisane. Zazwyczaj po zerowaniu do rejestrów przetwornika wpisywany jest zestaw sensownych wartości umożliwiających jego standardową pracę. Jednak zmiana parametrów takich jak rozdzielczość wymaga już ingerencji w zawartość rejestrów.

Oprócz opisanych linii moduł przetwornika PPV401 ma jeszcze linie zerującą (*Reset*), wprowadzającą układ w stan uśpienia (*Standby*) oraz wejście zewnętrznych sygnałów taktujących. Inne przetworniki mogą mieć dodatkowe linie sygnałowe, jednak nie powinny one mieć większego znaczenia przy dołączaniu sterownika do systemu z mikrokontrolerem.

Dołączanie przetwornika obrazu do mikrokontrolera

Zanim zostanie przedstawiony przykład konkretnego rozwiązania sprzętowego najpierw przyjrzyjmy się jakie najważniejsze warunki należy spełnić:

- **Dopasowanie poziomów.** Większość modułów jest wyposażonych w magistrale pracujące z poziomami logicznymi ok. 0 V dla poziomu niskiego i 2,8…3,3 V dla poziomu wysokiego. Można oczywiście posłużyć się dwukierunkowymi konwerterami poziomów, jednak najłatwiej jest zastosować mikrokontroler, którego porty mogą bezpośrednio współpracować z wyprowadzeniami modułu.
- **Liczba linii sterujących.** Dokładna liczba potrzebnych portów zależy od typu zastosowanego przetwornika; przeciętnie będzie to do 16 dwukierunkowych linii.
- **Szybkość przetwarzania danych.** Nawet w podstawowym wariancie odczytu z przetwornika jednej ramki statycznego obrazu mikrokontroler i jego porty powinny być w stanie przechwycić strumień danych o szybkości ok. 20 Mb/s
- **Wielkość dostępnego bufora danych.** Surowe dane obrazu zajmują liczą wiele bajtów. Najczęściej nie jest możliwa obróbka danych już na etapie ich odczytu z modułu i w pamięci operacyjnej jest potrzebny, do którego będzie zapisywany obraz. Dla przykładu, obraz o rozdzielczości 640×480 pikseli w formacie 5:6:5 (2 bajty danych dla składowych koloru dla 1 piksela) potrzebuje bufora o pojemności 614400 bajtów. W przypadku obrazu o rozdzielczości 160×120 pikseli w formacie 5:6:5 potrzebny jest bufor o pojemności 38400 bajtów.



Rysunek 3. Schemat rozwiązania opisywanego w artykule

Na **rysunku 3** pokazano sprawdzony schemat dołączenia modułu przetwornika PPV401 do mikrokontrolera z rodziny STM32. Mikrokontrolery z tej rodziny mogą być zasilane napięciem z przedziału 2…3,2 V przy wewnętrznym taktowaniu sygnałem o częstotliwości do 72 MHz. Liczba dostępnych portów zależy od wybranego typu mikrokontrolera i rodzaju jego obudowy. Porty mogą pracować z sygnałami o częstotliwości do 50 MHz. W niektórych układach np. STM32F103RC, STM32F103RE lub STM32F103RG wewnętrzna pamięć RAM jest na tyle duża, że można jej część wykorzystać jako bufor dla danych obrazu o rozdzielczości 160×120 pik-

seli bez konieczności dołączania dodatkowej, zewnętrznej pamięci RAM. Stosując układy z rodziny STM lub podobne można także skorzystać z mechanizmu DMA pozwalającego na niemal automatyczny zapis danych z magistrali modułu do pamięci RAM mikrokontrolera.

Na **rys. 3** pokazane są połączenia pomiędzy modułem przetwornika obrazu a mikrokontrolerem. Pominięto niektóre obwody niezbędne w pracy rzeczywistego urządzenia, takie jak interfejs JTAG, elementy obwodu zerowania i interfejsu UART. Przyporządkowanie portów mikrokontrolera najczęściej jest dowolne, jednak wybór kilku konkretnych portów wiązał się z wykorzystaniem w pracy oprogramowania sterującego mechanizmu DMA.

Linie KPC0…KPC7 są dołączone do wyjścia magistrali danych obrazu wysyłanych z przetwornika jako ciągły strumień danych. W tym celu można wybrać dowolny port mikrokontrolera, jednak jest korzystne wybranie linii portu od najmłodszej wzwyż. Dzięki temu możliwy będzie bezpośredni zapis do buforu RAM bez konieczności dodatkowych manipulacji związanych z formowaniem bajtu lub słowa danych.

Linie KPA10 i KPB9 obsługują magistralę szeregową, za pośrednictwem której przesyłane są rozkazy sterujące pracą modułu. O ile nie zamierzamy wykorzystać wsparcia sprzętowego, jakie oferują mikrokontrolery z rodziny STM32 dla transmisji I²C obie linie, tak jak poprzednio, mogą być wybrane dowolnie.

Sygnały *RSTB* (zerowania) i *STDBY* (uśpienia) ustawiane są zwykle na początku pracy lub sporadycznie w tych przypadkach, gdy należy uśpić przetwornik lub wyłączyć go, zwykle dla obniżenia poboru mocy. Wybór linii wyjściowych KPB6 i KPA11 sterujących tymi funkcjami jest całkowicie dowolny.

Linie wejściowe KPA9 (*VSYNC*) i KPA12 (*HSYNC*) są dołączone do sygnałów synchronizujących, za pomocą których przetwornik informuje mikrokontroler o rozpoczęciu wysyłania danych obrazu oraz przesyłaniu kolejnej jego linii. Dla ułatwienia, kontrolę tych sygnałów można zautomatyzować dołączając je do przerwań uruchamiających odpowiednie procedury obsługi. Ponieważ w mikrokontrolerach STM32 każdy port może być źródłem przerwania, więc sygnały synchronizacji można dołączyć do dowolnej linii portu.

Podłączenie sygnału *PCLK*, którego zbocze narastające sygnalizuje pojawienie się na magistrali kolejnego bajtu danych piksela obrazu nie może być zupełnie dowolnie i wiąże się z mechanizmem DMA.

Listing 1. Ustawienie linii magistrali danych obrazu jako wejściowych z wewnętrznym podciąganiem do plusa napięcia zasilania

```
//procedura inicjacji linii magistrali danych kamery
void KAM_Magistrala_danych_inicjacja()
{
  GPIO_InitTypeDef GPIO_InitStructure;
/* Enable Clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
/* Configure pin */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0; //inicjacja linii KPC0
magistrali
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOC, &GPIO_InitStructure);
}
```

Listing 2. Inicjowanie kanału DMA

```
/* inicjacja DMA dla danych przepisywanych z magistrali modułu do
bufora RAM */
void Inicjacja_DMA_modul_RAM()
{
  uint16_t rozmiar_buforu=38400;
/* Enable DMA1 clock */
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
/* DMA1 channel1 configuration */
  DMA_DeInit(DMA1_Channel1);
  DMA_InitStructure.DMA_PeripheralBaseAddr = KAM_DAT_Address;
  DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)&bufor_RAM_
danych[POCZATEK_BUFORU_16_LINII_OBRAZU];
  DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
  DMA_InitStructure.DMA_BufferSize = rozmiar_buforu;//(bufor w RAM
38400 bajtów
  DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
  DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
  DMA_InitStructure.DMA_PeripheralDataSize = DMA_
PeripheralDataSize_Byte;
  DMA_InitStructure.DMA_MemoryDataSize = DMA_PeripheralDataSize_
Byte;
  DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
  DMA_InitStructure.DMA_Priority = DMA_Priority_High;
  DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
  DMA_Init(DMA1_Channel1, &DMA_InitStructure);
/* Enable DMA1 channel1 współpraca z TIMER2 CC3 wejście PB10*/
  DMA_Cmd(DMA1_Channel1, ENABLE);
}
```

Listing 3. Inicjowanie Timera 2

```
void TIMER2_Inicjacja(void)
{
/* ----------------------------------------------------------------
TIM2 Configuration: kanał TIM2_CH3 w połączeniu z DMA obsługuje
zapis danych z magistrali danych obrazu do buforu w pamięci RAM
mikrokontrolera TIM2CLK = 36 MHz, Prescaler = 36, TIM2 counter
clock = 1MHz(1us)
---------------------------------------------------------------- */
/* TIM2 disable counter */
  TIM_Cmd(TIM2, DISABLE);
/* Time base configuration */
  TIM_TimeBaseStructure.TIM_Period = 65535;
  TIM_TimeBaseStructure.TIM_Prescaler = 1;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
/* Prescaler configuration */
  TIM_PrescalerConfig(TIM2, 36, TIM_PSCReloadMode_Immediate);
  TIM_ICStructInit(&TIM_ICInitStructure);
//kanał 3 „input capture”
  TIM_ICInitStructure.TIM_Channel =TIM_Channel_3;
//reakcja na zbocze narastające sygnału PCLK
  TIM_ICInitStructure.TIM_ICPolarity =/*TIM_ICPolarity_Falling*/
TIM_ICPolarity_Rising;
  TIM_ICInitStructure.TIM_ICSelection =TIM_ICSelection_DirectTI;
//bez peskalera zdarzeń, każde zdarzenie (zbocze) uwzględniane
  TIM_ICInitStructure.TIM_ICPrescaler =TIM_ICPSC_DIV1;
//bez filtru
  TIM_ICInitStructure.TIM_ICFilter =0;
  TIM_ICInit(TIM2, &TIM_ICInitStructure);
/* TIM2 enable counter */
  TIM_Cmd(TIM2, ENABLE);
}
```

DMA sprzętowy zapis danych obrazu do bufora

Praca DMA polega na automatycznym przepisywaniu danych z jednego miejsca do innego za pomocą mechanizmów sprzętowych mikrokontrolera, bez konieczności angażowania CPU. W tym przypadku będzie chodziło o odczyt danych z magistrali obrazu i ich zapis do bufora w pamięci RAM mikrokontrolera. Mechanizm DMA poza zaprogramowaniem, uruchomieniem i zatrzymaniem nie wymaga ingerencji jednostki centralnej, która w tym czasie może wykonywać inne operacje. Dodatkowo, jest to mechanizm szybki i może być jedynym rozwiązaniem przy odbiorze danych z przetwornika.

DMA może funkcjonować na dwa sposoby. Raz uruchomiony będzie stale odczytywał dane z magistrali przesyłając je do bufora. Nie jest to właściwe rozwiązanie w przypadku odczytu z przetwornika obrazu. Dane wysyłane są co prawda nieprzerwanie, ale nie w sposób ciągły, tylko linia po linii, piksel po pikselu. Dlatego aby mogły być prawidłowo odebrane i zapisane w buforze w pamięci RAM, praca DMA powinna być synchronizowana.

W przyjętym rozwiązaniu kanał DMA jest powiązany z wewnętrznym Timerem2 mikrokontrolera, a dokładnie z jego kanałem CH3. Timer2 pracuje w trybie reakcji na zdarzenie *input–capture*. W momencie zaistnienia zdarzenia, kanał 3 Timera uruchamia DMA, które przepisu-

je dane z magistrali KPC0…KPC7 do bufora w pamięci RAM, zwiększa wewnętrzny licznik i oczekuje na kolejne zdarzenia. Tymi zdarzeniami są narastające zbocza impulsów sygnalizujące pojawienie się na magistrali stabilnych danych kolejnego piksela linii obrazu. Impulsy taktujące, czyli PCLK, wysyłane są z przetwornika do mikrokontrolera linią KPB10. Ponieważ impulsy PCLK nie są wysyłane w przerwach pomiędzy kolejnymi liniami obrazu, kanał DMA prześle do bufora w sposób uporządkowany tylko dane pikseli obrazu. W przetwornikach, w których impulsy PCLK są generowane nieprzerwanie również w przerwach pomiędzy liniami obrazu, należy użyć przerwania wywoływanego zboczami sygnału HSYNC do chwilowego wyłączania działania DMA w przerwach pomiędzy liniami.

Podstawowe procedury programu sterującego

Oprócz fizycznego połączenia przetwornika z mikrokontrolerem do obsługi odbioru danych z przetwornika obrazu jest potrzebne wewnętrzne oprogramowanie, a dokładniej – procedury obsługi przetwornika. Procedury powinny zainicjować porty mikrokontrolera połączone z modułem, DMA i Timer sterujący, zainicjować wewnętrzne rejestry przetwornika, a potem, w odpowiednim momencie, uruchomić DMA i zatrzymać je po zakończeniu przesłania danych obrazu.

Inicjacja wewnętrznych rejestrów przetwornika obrazu zależy od jego typu. O tym, które rejestry zainicjować i jakimi wartościami, można dowiedzieć się z dokumentacji przetwornika. Inicjację układów mikrokontrolera dla konfiguracji pokazanej na rys. 3 można opisać w kilku kolejnych krokach:

- Ustawienie trybu pracy linii dołączonych do magistrali danych obrazu jako wejściowych z wewnętrznym podciąganiem do plusa napięcia zasilania (**listing 1**).
- Ustawienie linii KPA10 i KPB9 szeregowej magistrali sterującej jako wyjściowych w trybie otwartego drenu. Dzięki temu można zarówno ustawiać poziom linii jak i go odczytywać co jest niezbędne w przypadku linii SDA. Należy pamiętać o zastosowaniu zewnętrznych oporników podciągających poziomy obu linii do napięcia zasilającego.
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_OD; //open drain
- Ustawienie linii sterujących sygnałami RSTB i STDBY jako wyjściowych a linii sygnałów synchronizujących HSYNC, VSYNC, PCLK jako wejściowych z wewnętrznym podciąganiem do napięcia zasilania.
  GPIO_InitStructure.GPIO_Mode =GPIO_Mode_Out_PP; // RSTB, STDBY
  GPIO_InitStructure.GPIO_Mode =GPIO_Mode_IPU; // HSYNC, VSYNC, PCLK
- Zainicjowanie kanału DMA (**listing 2**).
- Remapowanie portu PB10 (sygnał PCLK) dla podłączenia do TIMER2 CH3.
  //podłączenie zegara do funkcji alternatywnych
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
  GPIO_PinRemapConfig(GPIO_PartialRemap2_TIM2, ENABLE);
- Inicjowanie Timera 2 (**listing 3**).

Po ustawieniu linii RSTB i STDBY sterujących modułem na odpowiednim poziomie, system jest gotowy do przechwycenia ramki pojedynczego obrazu. Procedura, która kontroluje ten proces może wyglądać (słowo „ramka" w komentarzu oznacza kompletny jeden obraz w ciągłym strumieniu danych transmitowanych magistralą) jak na **listingu 4**.

W podanym przykładzie zakończenie przesyłania bloku danych ramki obrazu sygnalizuje wewnętrzna flaga mikrokontrolera powiązana z kanałem DMA DMA1_FLAG_TC1. Inną metodą jest zliczanie impulsów HSYNC sygnalizujących przesyłanie kolejnych linii ramki obrazu. Po odliczeniu oczekiwanej ilości linii można wyłączyć DMA.

Listing 4. Przechwycenia ramki pojedynczego obrazu

```
/* oczekiwanie na stan niski VSYNC -przerwę pomiędzy kolejnymi
ramkami danych obrazu */
while (VSYNC) !=0);
/* oczekiwanie na stan wysoki VSYNC -start przesyłu danych kolejnej
ramki obrazu */
while (VSYNC) ==0);
/* uruchomienie DMA sterowanego przez TIM2 Capture Compare 3 */
TIM_DMACmd(TIM2, TIM_DMA_CC3, ENABLE);
/* oczekiwanie aż flaga DMA1_FLAG_TC1 zasygnalizuje zakończenie
przesyłu DMA bloku 38400 bajtów */
while(!DMA_GetFlagStatus(DMA1_FLAG_TC1));
DMA_ClearFlag(DMA1_FLAG_TC1); //zerowanie flagi
TIM_DMACmd(TIM2, TIM_DMA_CC3, DISABLE); //zatrzymanie DMA
DMA_Cmd(DMA1_Channel1, DISABLE); //wyłączenie DMA
```

Podsumowanie

Przedstawiony opis dotyczy najprostszego rozwiązania gdy obrazek ma minimalną rozdzielczość. Przy większych obrazkach ilość danych wymaga dołączenia do systemu zewnętrznych pamięci RAM dla danych obrazu.

Ryszard Szymaniak, EP