Zintegrowany przetwornik obrazu w systemie z mikrokontrolerem

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

Poco stosować przetworniki obrazu? Pierwsza odpowiedź jest najprostsza: 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. Obraz można wykorzystać jako źródło danych dla sterowników, detektorów ruchu, przy pomiarach obiektów, detekcji zdarzeń itp. Możliwości jest sporo jednak żeby 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 takimi angielskimi określeniami kryją się elementy które nas interesują. Przetworniki obrazu mogą być wykonane w formie układów scalonych współpracujących z zewnętrznym systemem optycznym czyli obiektywem. Takie przetworniki stosowane są w klasowym sprzęcie jak cyfrowe aparaty czy kamery. Przetworniki mogą być także zintegrowane z optyką 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.

Drugim ważnym rozróżnieniem jest format danych wyjściowych jakie 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.

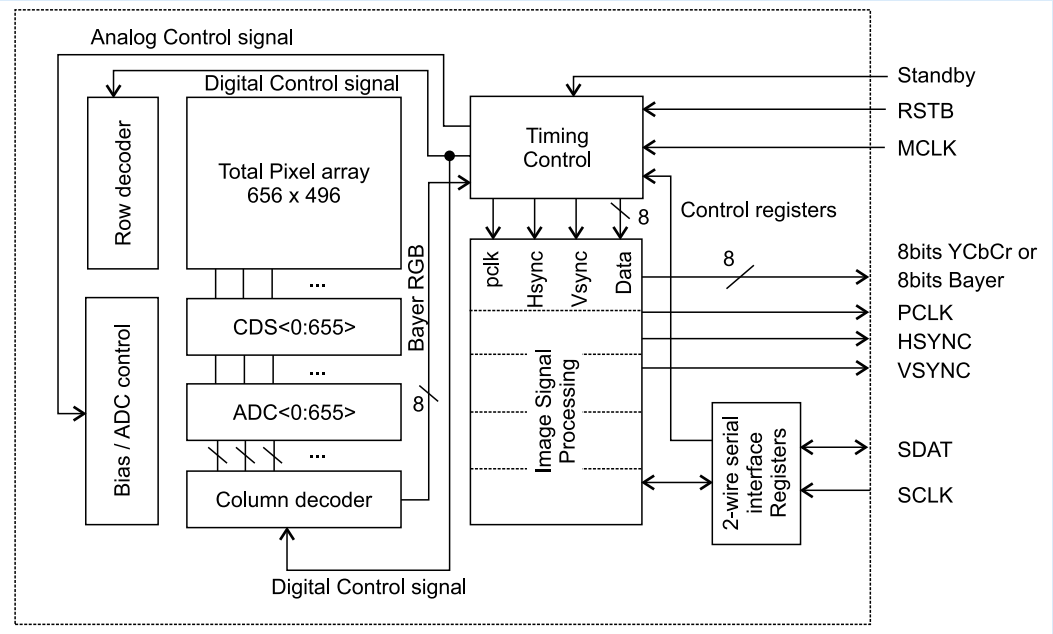
Można jeszcze dokonać kilku podziałów między innymi ze względu na szczegóły budowy, matrycę, cenę ale jednym z najważniejszych jest dostępność lub brak dokumentacji technicznej. Przetwornik o najciekawszych możliwościach bez precyzyjnej dokumentacji z opisem wewnętrznych rejestrów i sposobu sterowania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

Niniejszy opis wykorzystania przetwornika obrazu w systemie mikroprocesorowym oparty został na elemencie PPV401 z wewnętrznym sterownikiem PO6030 wyprodukowanym przez firmę PIXELPLUS. To względnie prosty w działaniu element dostarczający nie kodowany strumień danych obrazowych. Można go zastosować w systemie z mikrokontrolerem o przeciętnych możliwościach. Wadą elementu jest jego trudna dostępność. Krajowy dystrybutor wycofał się z jego sprzedaży. Jednak sposób działania przetwornika jest bardzo podobny do innych dostępnych na rynku układów. Znając ogólne zasady łatwiej szukać kluczowych danych w dokumentacji bardziej dostępnych przetworników np. firm APTINA cz OmniVision czy oferowanych przez dalekowschodnich producentów modułach kamer.

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 prostym przetwornikiem dostarczającego ciągłego strumienia nie kodowanej informacji o kolejnych pikselach obrazu. Ma rozdzielone magistrale dla danych obrazu i sygnałów sterujących oraz kilka dodatkowych linii sygnałów synchronizujących. Jego ogólna budowa pokazana została 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 sposób sterowania.

Matryca składa się z mozaiki miniaturowych elementów światłoczułych rejestrujących informacje o natężeniu światła pojedynczych punktów (pikseli) składających się na przetwarzany obraz (**rysunek 2**). Dla uzyskania informacji o kolorze przed światłoczułymi elementami umieszczone są kolorowe filtry dla trzech podstawowych barw na które reaguje ludzkie oko: czerwonej, zielonej, niebieskiej



Rysunek 1. Schemat blokowy przetwornika obrazu PPV401

czyli w skrócie RGB. Na Rys2 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 ilości 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 wysyłane są 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 ilości danych o obrazie.

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 ilość niezbędnych linii a ograniczona szybkość przepływu magistralą tego typu nie ma wielkiego znaczenia. W przetworniku PPV401 magistrala sterująca działa w formacie bardzo podobnym do standardowego I²C. Występują bity Statu 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.

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

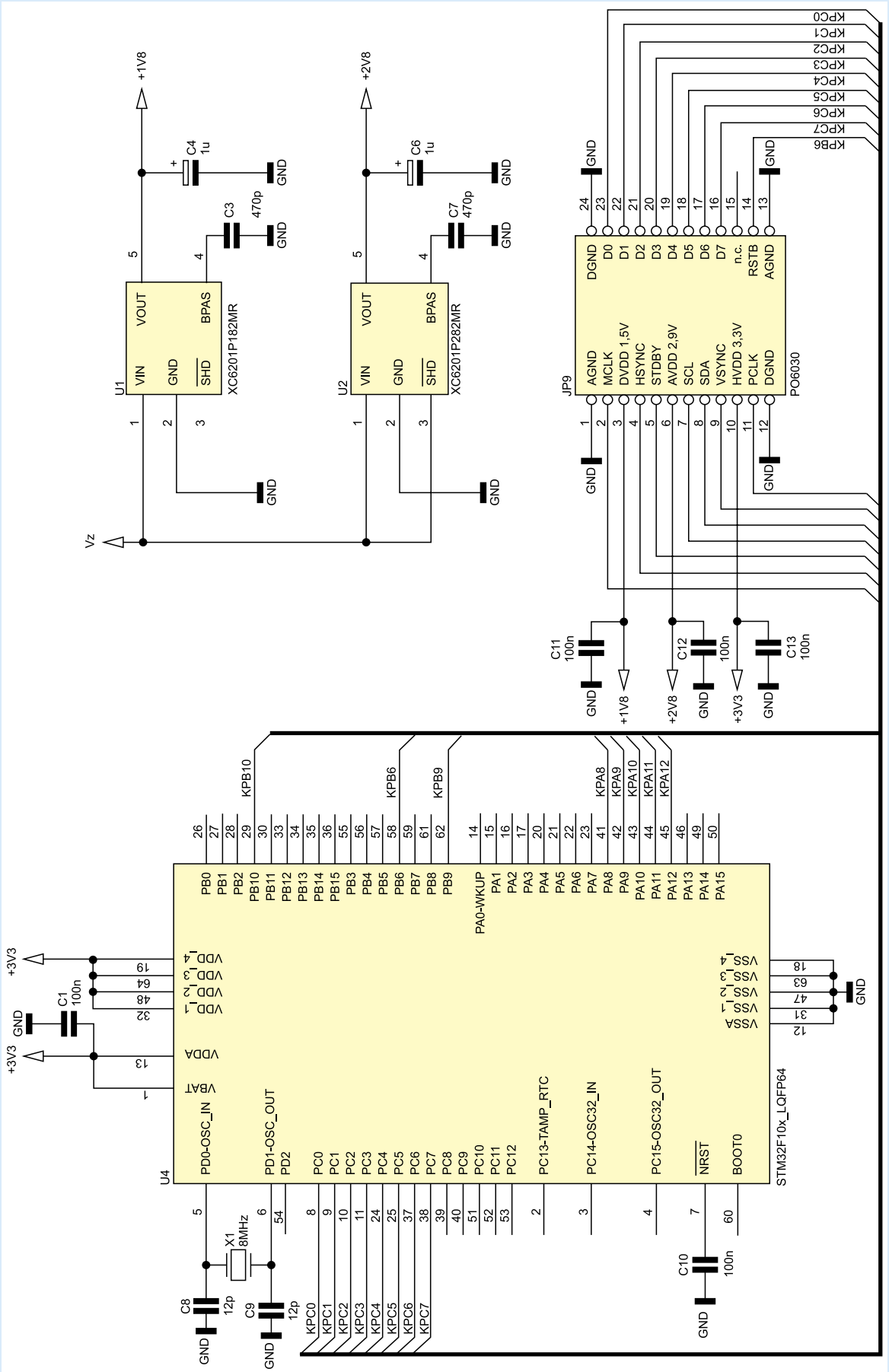
Rysunek 2. Budowa matrycy RGB przetwornika obrazu

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

Dołączenie przetwornika obrazu z mikrokontrolerem

Zanim zostanie przedstawiony przykład konkretnego rozwiązania sprzętowego najpierw przyjrzyjmy się zestawieniu najważniejszych warunków które należy spełnić:

Dopasowanie poziomów. Większość modułów wyposażonych jest w magistrale pracujące z poziomami logicznymi 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



Rysunek 3. Schemat dołączenia przetwornika do systemu z mikrokontrolerem STM32

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 najprostszym 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.

Rozmiar dostępnego bufora danych. Dane obrazu w stanie surowym zajmują sporo miejsca. Najczęściej nie jest możliwa obróbka danych już na etapie ich odczytu z modułu i potrzebny jest bufor, do którego będą zapisywane. Dla przykładu, obraz o rozdzielczości 640×480 pikseli w formacie 5:6:5 (2 bajty danych dla składowych koloru dla 1 piksela) potrzebuje bufora mieszczącego 614400 bajtów. Dla obrazka o rozdzielczości 160×120 pikseli w formacie 5:6:5 jest potrzebny bufor o pojemności 38400 bajtów.

Na **rysunku 3** pokazano przetestowany schemat podłączenia modułu przetwornika PPV401 do mikrokontrolera z rodziny STM32. Sterowniki z tej grupy mogą być zasilane napięciem z przedziału 2…3,2 V przy wewnętrznym taktowaniu do 72 MHz. Liczba dostępnych portów zależy od wybranego typu mikrokontrolera i jego obudowy. Porty są w stanie pracować z sygnałami o częstotliwości do 50 MHz. W niektórych układach np. STM32F103RC, STM32F103RE czy STM32F103RG, wewnętrzna pamięć RAM jest na tyle duża, że można jej część wykorzystać jako bufor dla danych obrazka o rozdzielczości 160×120 pikseli 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. Pominięte zostały niektóre obwody niezbędne w pracy rzeczywistego urządzenia takie jak interfejs JTAG, elementy resetu czy 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 podłączone są do wyjścia magistrali danych obrazu wysyłanych z przetwornika jako ciągły strumień danych. Można wybrać dowolny port mikrokontrolera, jednak korzystne jest 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 mają mikrokontrolery z rodziny STM32 dla transmisji I^2C, obie linie, tak jak poprzednio, mogą być dowolnie wybrane.

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

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

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

DMA sprzętowy zapis danych obrazu do buforu

DMA polega na automatycznym przepisywaniu danych z jednego miejsca do innego przy pomocy mechanizmów sprzętowych mikrokontrolera. W tym wypadku będzie chodziło o odczyt danych z magistrali obrazu i ich zapis do buforu w wewnętrznej pamięci RAM mikrokontrolera. Mechanizm DMA, poza jego zaprogramowaniem oraz wystartowaniem i zatrzymaniem, nie wymaga ingerencji CPU, które w tym czasie może wykonywać inne operacje. W dodatku jest to mechanizm szybki i może być jedynym rozwiązaniem przy odbiorze danych z przetwornika. Sterownik DMA może funkcjonować na dwa sposoby. Raz uruchomiony będzie stale odczytywał dane z magistrali przesyłając je do bufora. Nie jest to właściwe rozwiązanie w wypadku odczytu z przetwornika obrazu. Dane wysyłane są co prawda nieprzerwanie, ale nie w sposób nie ciągły, tylko linia po linii i 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 przepisuje dane z magistrali KPC0-KPC7 do bufora w pamięci RAM, zwiększa swój wewnętrzny licznik i oczekuje na kolejne zdarzenie. 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 buforu w sposób uporządkowany tylko dane pikseli obrazu. W przypadku przetworników w których impulsy PCLK są generowane bez przerwy także 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

Do obsługi odbioru danych z przetwornika obrazu oprócz fizycznego podłączenia przetwornika do mikrokontrolera potrzebne jest wewnętrzne oprogramowanie a dokładniej procedury obsługi przetwornika. Procedury powinny zainicjować porty mikrokontrolera połączone

z modułem, DMA i sterujący Timer, zainicjować wewnętrzne rejestry przetwornika, a potem w odpowiednim momencie wystartować DMA i zatrzymać je po zakończeniu przesłania danych obrazka.

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 magistrali danych obrazu jako wejściowych z wewnętrznym podciąganiem do 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);
}.
```

Ustawienie trybu pracy 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 trybu pracy 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:
//inicjacja DMA dla danych przepisywanych z magistrali modułu do bufora w 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  RAM  o  rozmiarze 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);
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 Timer 2
void TIMER2_Inicjacja(void)
{
  /* ------------------------------------
-------------------------
       TIM2  Configuration:  kanał  TIM2_CH3 w połączeniu z DMA obsługuje zapis danych z  magistrali  danych  obrazu  do  bufora 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 filtra
  TIM_ICInitStructure.TIM_ICFilter =0;
  TIM_ICInit(TIM2, &TIM_ICInitStructure);
  /* TIM2 enable counter */
  TIM_Cmd(TIM2, ENABLE);
}
```

Po wymuszeniu odpowiedniego poziomu linii sterujących modułem RSTB i STDBY, system jest gotowy do przechwycenia ramki pojedynczego obrazu. Procedura, która kontroluje ten proces może wyglądać następująco (słowo „ramka" w komentarzu oznacza kompletny jeden obraz w ciągłym strumieniu danych transmitowanych magistralą):

```
//oczekiwanie na wyzerowanie 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
```

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.

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