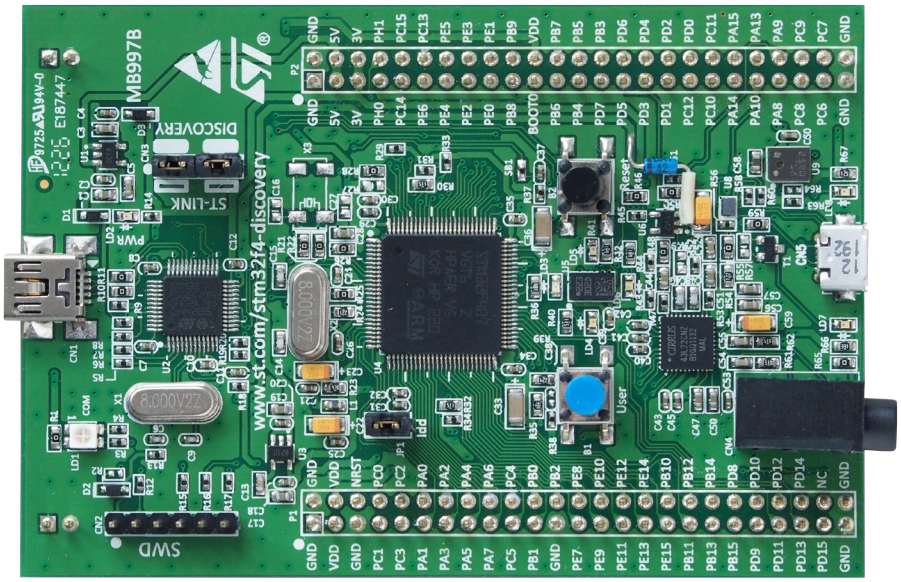
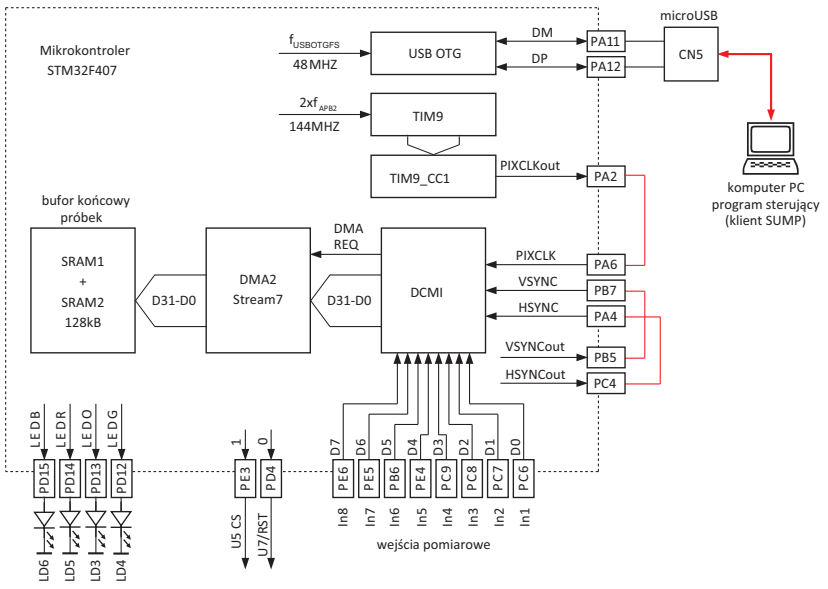
Analizator stanów logicznych z modułu STM32F4DISCOVERY

W artykule przedstawiono kolejny przykład użycia zestawu uruchomieniowego mikrokontrolera w roli analizatora stanów logicznych. Tym razem projekt analizatora powstał na bazie zestawu STM32F4DISCOVERY. Dzięki niestandardowemu użyciu układu DCMI w mikrokontrolerze STM32F407 uzyskano analizator stanów logicznych o godnych uwagi parametrach, przewyższający swymi osiągami popularny wśród hobbystów i studentów analizator Saleae.

Podstawowe parametry:
- Mikrokontroler typu STM32F407VGT6.
- Liczba wejść: 8.
- Poziom sygnałów wejściowych: +3 V/+5 V.
- Częstotliwość próbkowania sygnałów wejściowych: 1 kHz/2 kHz/5 kHz/10 kHz/20 kHz/50 kHz/100 kHz/200 kHz/500 kHz/1 MHz/2 MHz/6 MHz/12 MHz/24 MHz/48 MHz.
- Pojemność bufora pamięci próbek: 1 kB/2 kB/4 kB/8 kB/16 kB/32 kB/64 kB/128 kB.
- Wyzwalanie pomiaru dowolną kombinacją poziomów logicznych na wejściach.
- Opóźnienie wyzwolenia: od 0% do 100% pozycji bufora próbek.
- Protokół komunikacji: SUMP.



Fotografia 1. Zestaw uruchomieniowy STM32F4DISCOVERY



Rysunek 2. Układ analizatora stanów logicznych STM32F4DISCOVERY LA

Pokazany na **fotografii 1** zestaw uruchomieniowy STM32F4DISCOVERY jest oferowany przez STMicroelectronics jako tania platforma pozwalająca na zapoznanie się z mikrokontrolerem STM32F407 oraz prowadzenie prac rozwojowych. Ten zestaw pozwala przede wszystkim na prezentację możliwości mikrokontrolera STM32F407 w zakresie przetwarzania sygnałów akustycznych. W tym celu wyposażono go w mikrofon MEMS oraz w przetwornik audio DAC ze zintegrowanym wzmacniaczem akustycznym klasy D. Dodatkowo, na płytce modułu zamontowano też 3-osiowy akcelerometr, 8 diod LED i dwa przyciski. Jak każdy zestaw z rodziny Discovery, moduł wyposażono również w układ programatora-debuggera typu ST-LINK/V2 (starsze wersje płytki) lub ST-LINK/V2-A (nowsze wersje płytki zgodne z ARM mbed). Szczegółowe dane techniczne modułu można znaleźć w jego instrukcji użytkownika [1]. Zestaw STM32F4DISCOVERY był też szerzej opisywany na łamach „Elektroniki Praktycznej” [2].

Należy zwrócić uwagę na to, że firma STMicroelectronics oferuje również dwie inne płytki prototypowe, wyglądające na pierwszy rzut oka identycznie jak moduł STM32F4DISCOVERY. Są to: STM32F401C-DISCO oraz STM32F411E-DISCO. Płytki te są wyposażone w mikrokontrolery, odpowiednio: typu STM32F401VCT6 i STM32F411VET6. Niestety, pomimo bardzo dużego podobieństwa zewnętrznego do zestawu STM32F4DISCOVERY, oba wymienione moduły nie mogą być użyte w opisywanym projekcie. Główna przyczyna leży w tym, że mikrokontrolery STM32F401 i STM32F411 są pozbawione układu DCMI. Poza tym maksymalna

częstotliwość taktowania rdzenia w tych mikrokontrolerach jest znacznie niższa niż częstotliwość taktowania rdzenia w mikrokontrolerze STM32F407. Wynosi ona 80 MHz dla układu STM32F401 i 100 MHz dla STM32F411. Układ STM32F407 może być taktowany z maksymalną częstotliwością 168 MHz.

Budowa analizatora

Schemat blokowy układu analizatora stanów logicznych pokazano na **rysunku 2**. Analizator ma typową budowę. Składa się z modułu sprzętowego oraz uruchomionego na komputerze PC programu sterującego. Moduł sprzętowy próbkuje podawane na wejścia pomiarowe analizatora In1...In8 mierzone sygnały oraz zapisuje w lokalnym buforze uzyskane próbki, natomiast program sterujący odpowiada za konfigurowanie modułu sprzętowego, obrazowanie zarejestrowanych przez ten moduł sygnałów, ich analizę i pomiary. Sam moduł sprzętowy jest w całości zbudowany z odpowiednio połączonych i skonfigurowanych układów peryferyjnych mikrokontrolera STM32F407, z których najważniejszą funkcję pełni układ DCMI.

Układ DCMI (tj. Digital Camera Interface – **rysunek 3**) jest układem peryferyjnym w mikrokontrolerach STM32 przeznaczonym do obsługi kamer cyfrowych. Jest on implementowany w niektórych zaawansowanych modelach mikrokontrolerów STM32, głównie z rodziny STM32F4 i STM32F7. Dzięki elastycznej budowie może on obsługiwać wiele typów kamer o różnej szerokości szyny danych (od 8 do 14 bitów) i różnym formacie transmisji. W mikrokontrolerze STM32F407 dane te mogą być taktowane z maksymalną częstotliwością 54 MHz.

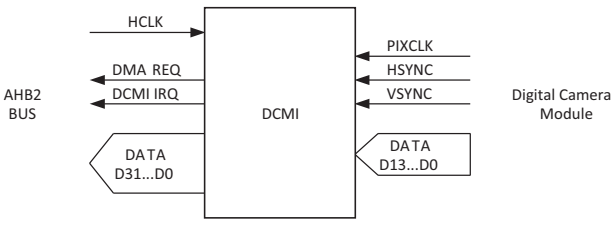
W jednym ze swoich trybów pracy układ DCMI może odbierać z kamery skompresowane obrazy w formacie JPEG. W takim przypadku dane obrazu są przesyłane do mikrokontrolera 8-bitową szyną danych D0…D7 (linie danych D8…D13 nie są używane). Taktowanie transmisji zapewnia linia sygnału zegarowego PIXCLK. W tym trybie linia VSYNC jest używana do sygnalizacji początku i końca obrazu JPEG, natomiast linia HSYNC jest używana do sygnalizacji ważności danych (**rysunek 4**). W układzie DCMI istnieje możliwość wyboru aktywnego zbocza sygnału zegarowego PIXCLK oraz aktywnych poziomów sygnałów HSYNC i VSYNC. W typowej konfiguracji układu DCMI źródłem sygnałów sterujących PIXCLK, HSYNC i VSYNC jest dołączona kamera. Nic jednak nie stoi na przeszkodzie, aby te sygnały były generowane lokalnie w mikrokontrolerze. W takim wypadku układ DCMI staje się dla mikrokontrolera szybkim, równoległym interfejsem wejściowym, mogącym obsługiwać odczyt dowolnych danych z dołączonych do linii D0…D7 układów zewnętrznych. Na tym właśnie pomyśle użycia układu DCMI opiera się działanie prezentowanego analizatora stanów logicznych.

Mierzone sygnały cyfrowe są podawane na linie D0…D7 układu DCMI, które są wyprowadzone na porty I/O PB6, PC6...PC9 i PE4...PE6 mikrokontrolera. Teoretycznie, w mikrokontrolerze STM32F407 porty te tolerują sygnały o napięciu do +5 V. **Ponieważ jednak w module STM32F4DISCOVERY do części portów I/O mikrokontrolera współdzielonych z układem DCMI są dołączone układy zewnętrzne, w wypadku linii D1 (port PC7) oraz linii D5 (port PB6) doprowadzane sygnały mogą mieć napięcie tylko +3 V. Oczywiście, w wypadku pozostałych wejść doprowadzane sygnały mogą mieć napięcie +3 V lub +5 V.** Układ DCMI jest zaprogramowany w ten sposób, iż dokonuje odczytu stanu linii D0...D7 na każdym narastającym zboczu sygnału zegarowego PIXCLK. Odczytane dane są gromadzone w wewnętrznym buforze układu DCMI o długości 4 B. Po zapełnieniu bufora układ DCMI generuje do kontrolera DMA2 żądanie transferu danych. W odpowiedzi kontroler DMA2 odczytuje z bufora układu DCMI dane w postaci jednego 32-bitowego słowa i umieszcza je w pamięci SRAM, pełniącej w układzie analizatora funkcję kołowego bufora próbek sygnału mierzonego. Cały proces gromadzenia próbek mierzonych sygnałów doprowadzonych do wejść In1... In8 jest więc realizowany sprzętowo, bez udziału jednostki centralnej. Trwa on od momentu aktywacji linii VSYNC i HSYNC aż do ich wyłączenia.

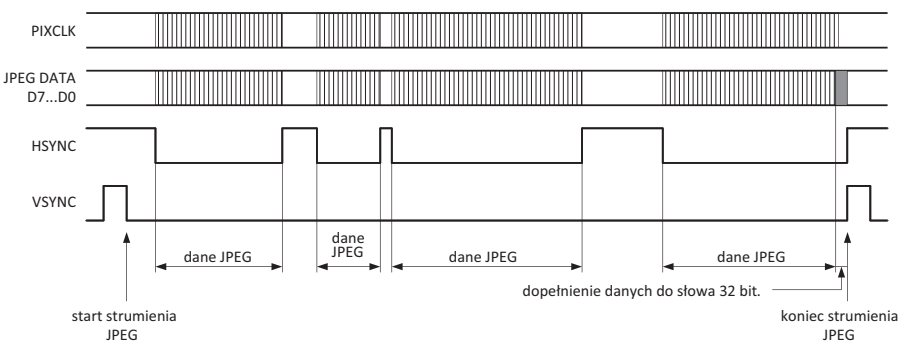
Linie sterujące VSYNC i HSYNC układu DCMI są kontrolowane w analizatorze sposób programowy. W tym celu zostały one dołączone, odpowiednio – do portów PB5 i PC4 mikrokontrolera, skonfigurowanych jako wyjścia. Program sterujący analizuje kolejno wartości zgromadzonych w pamięci SRAM próbek mierzonego sygnału, porównując je z wzorcem. Po wykryciu zaprogramowanej sekwencji wyzwalającej pomiar program oczekuje na zapis do bufora zadanej przed pomiarem liczby próbek, po czym zatrzymuje proces próbkowania, ustawiając porty PB5 i PC4, tym samym deaktywując linie VSYNC i HSYNC. W ten sposób ustalając pojemność bufora próbek, wzorzec wyzwolenia pomiaru oraz liczbę próbek sygnału po wyzwoleniu, można w analizatorze wpływać na czas pomiaru, moment jego wyzwolenia oraz opóźnienie wyzwolenia.

Sygnał zegara PIXCLK dla układu DCMI jest generowany przez układ czasowo-licznikowy TIM9, przy czym licznik układu TIM9 realizuje odmierzanie okresu sygnału PIXCLK. Kanał 1 obwodu próbkująco-porównującego układu TIM9 (tj. TIM9_CC1), pracując w trybie PWM2, generuje właściwy sygnał zegara o wypełnieniu 50%. Sygnał ten jest następnie wyprowadzony na port PA2 mikrokontrolera i podawany na wejście PIXCLK układu DCMI (tj. port PA6). Pożądaną częstotliwość sygnału zegarowego PIXCLK uzyskuje się, programując wartość podziału preskalera licznika TIM9 oraz okres zliczania licznika TIM9.

Układ czasowo-licznikowy TIM9 jest taktowany przebiegiem o częstotliwości 144 MHz (tj. 2×fAPB2). Jest to zarazem częstotliwość taktowania rdzenia mikrokontrolera STM32F407 i pozostałych układów dołączonych do magistral AHB. Taka wartość częstotliwości pozwala na generowanie przez układ TIM9 przebiegu zegarowego PIXCLK o maksymalnej częstotliwości 48 MHz. Co prawda rdzeń mikrokontrolera STM32F407 może być taktowany z maksymalną częstotliwością 168 MHz, ale paradoksalnie przy takiej częstotliwości na wyjściu układu TIM9 można uzyskać użyteczny sygnał zegara PIXCLK o maksymalnej częstotliwości równej 42 MHz, a więc znacząco niższej. Następna możliwa do uzyskania w tej konfiguracji wartość częstotliwości sygnału wyjściowego jest bowiem równa 56 MHz, co przekracza już dopuszczalną częstotliwość pracy układu DCMI.



Rysunek 3. Schemat funkcjonalny układu DCMI



Rysunek 4. Praca układu DCMI w trybie transferu obrazów w formacie JPEG

Niestety, w opisywanym projekcie w mikrokontrolerze STM32F407 nie można użyć innych częstotliwości taktowania rdzenia, które pozwoliłyby na uzyskanie równej siatki częstotliwości próbkowania sygnału pomiarowego (np. 50 MHz, 20 MHz, 10 MHz, itd.). Ograniczenie to jest spowodowane tym, że w mikrokontrolerze STM32F407 sygnały zegara dla rdzenia, układów peryferyjnych oraz układu USB OTG są generowane przez tę samą pętlę PLL. Ponieważ sygnał zegara układu USB OTG musi mieć stałą częstotliwość równą 48 MHz, częstotliwość taktowania rdzenia mikrokontrolera i pozostałych układów peryferyjnych musi być wielokrotnością tej liczby.

Połączenie analizatora stanów logicznych z komputerem PC jest obsługiwane przez układ USB OTG. Jest to w zasadzie jedyny możliwy sposób bezpośredniego podłączenia płytki STM32F4DISCOVERY do komputera PC. W przeciwieństwie bowiem do zestawów rodziny Nucleo, wyposażonych w układ programatora-debuggera ST-LINK/V2-1, ten dostępny w module STM32F4DISCOVERY nie umożliwia połączenia z mikrokontrolerem modułu przez wirtualny port szeregowy. W omawianym projekcie oprogramowanie układu USB OTG w mikrokontrolerze STM32F407 obsługuje klasę CDC interfejsu USB. Analizator stanów logicznych jest więc rozpoznawany w systemie komputera PC jako wirtualny port szeregowy.

Stan pracy analizatora stanów logicznych jest obrazowany przez cztery znajdujące się na płytce STM32F4DISCOVERY diody LED: LD3, LD4, LD5 i LD6. Opis pełnionych przez nie funkcji przedstawiono w **tabeli 1**.

W układzie analizatora używane są dodatkowo dwa porty I/O mikrokontrolera: PD4 i PE3. Są one skonfigurowane jako wyjścia i ustawione na stałe odpowiednio w stanie niskim (port PD4) i wysokim (port PE3). Ich zadaniem jest zablokowanie na płytce STM32F4DISCOVERY zewnętrznych układów, które są dołączone do mikrokontrolera do tych samych wyprowadzeń, których używa układ DCMI. I tak ustawiony w stanie niskim port PD4 utrzymuje w stanie ciągłego resetu układ przetwornika audio DAC i wzmacniacza w klasie D typu CS43L22 (U7), dzięki czemu zwolniona zostaje linia PA4 mikrokontrolera, stanowiąca wejście sygnału HSYNC układu DCMI. Z kolei ustawiony w stanie wysokim port PE3 przełącza układ akcelerometru LIS302DL (U5) w tryb pracy z interfejsem i2c, dzięki czemu zostaje uwolniona linia PA6 mikrokontrolera, stanowiąca wejście D5 układu DCMI.

Zastosowana w układzie analizatora stanów logicznych STM32F4Discovery LA metoda gromadzenia próbek sygnału mierzonego z użyciem układu DCMI ma szereg zalet w porównaniu do najczęściej spotykanego w podobnych projektach bezpośredniego odczytu stanu portu wejściowego mikrokontrolera w sposób programowy, przez jednostkę centralną, lub też sprzętowy, przez kontroler DMA. Po pierwsze, w układzie DCMI próbkowanie linii wejściowych następuje w ściśle określonym momencie – w prezentowanym projekcie jest ono wykonywane na narastającym zboczu przebiegu zegarowego PIXCLK. W razie bezpośredniego odczytu portu wejściowego mikrokontrolera występują natomiast fluktuacje momentu odczytu. Ma to miejsce nawet w sytuacji wykonywania tej operacji przez kontroler DMA, ponieważ ten układ konkuruje z jednostką centralną o dostęp do pamięci SRAM i bez specjalnych zabiegów typu unikanie w programie odczytu lub zapisu zmiennych w chwili dokonywania transferu DMA, niemożliwe jest dokładne określenie momentu wykonania przez kontroler DMA tej operacji. Po drugie, układ DCMI wstępnie gromadzi próbki mierzonego sygnału w lokalnym buforze o rozmiarze 4 B, dzięki czemu mogą one być następnie przesłane do pamięci SRAM przez kontroler DMA w postaci jednego słowa o długości 32 bitów. Zmniejsza to czterokrotnie częstotliwość transferów DMA, a więc i obciążenie magistrali pamięci SRAM, w stosunku do częstotliwości próbkowania sygnału. Dodatkowo dane wyjściowe układu DCMI są buforowane w 4-pozycyjnym rejestrze FIFO, co jeszcze bardziej poprawia płynność transferów DMA. W przypadku bezpośredniego odczytu portu wejściowego mikrokontrolera liczba transferów danych do pamięci SRAM musi być natomiast zawsze równa częstotliwości próbkowania wejść pomiarowych.

Zasilanie

Oryginalny zestaw STM32F4DISCOVERY może być zasilany albo z portu USB zewnętrznego komputera przez gniazdo miniUSB programatora – debuggera ST-LINK/V2 (tj. złącze CN1), albo z zewnętrznego zasilacza napięciem o wartości +5 V doprowadzonym do styków 3 i 4 złącza szpilkowego P2. W przypadku używania modułu STM32F4DISCOVERY w roli analizatora stanów logicznych żadna z powyższych metod zasilania układu nie jest jednak zbyt wygodna. W pierwszym wypadku jest bowiem konieczne dołączenie modułu Discovery do dwóch portów USB w komputerze, z których jeden będzie służył do zasilania układu, natomiast drugi – do komunikacji z programem sterującym analizatorem. W drugim wypadku jest natomiast niezbędny zewnętrzny zasilacz o napięciu wyjściowym +5 V, ewentualnie zasilanie modułu analizatora z mierzonego układu, w którym napięcie +5 V może jednak nie być dostępne. Najwygodniejsze wydaje się zasilanie analizatora stanów logicznych ze złącza USB zewnętrznego komputera tym samym przewodem, który jest używany do komunikacji z programem sterującym, czyli przez gniazdo microUSB interfejsu USB OTG (tj. złącze CN5). Proponowaną modyfikację obwodu zasilania płytki STM32F4DISCOVERY, pozwalającą na zasilanie modułu przez gniazdo CN5, przedstawiono na **rysunku 5**. Zmiana ta polega na dołączeniu diody DS pomiędzy pin nr 1 złącza CN5 a obwód napięcia +5 V na płytce Discovery. Dzięki temu napięcie +5 V z portu USB zewnętrznego komputera trafiające na pin 1 gniazda CN5 będzie przez dodaną diodę DS podawane na linię zasilania +5 V modułu STM32F4DISCOVERY. Ze względu na minimalizację spadku napięcia zasilania dodana dioda powinna być diodą Schottk'yego. Najlepiej, gdyby jej maksymalny prąd przewodzenia był nie mniejszy niż 0,5 A. W wykonanym prototypie, ze względu na dostępność, zastosowano jednak diodę BAT42 o maksymalnym ciągłym prądzie przewodzenia 0,2 A, co okazało się zupełnie wystarczające. W takim wypadku nie należy jednak zasilać z modułu analizatora stanów logicznych innych układów zewnętrznych, ponieważ prąd przewodzenia diody BAT42 może okazać się niewystarczający.

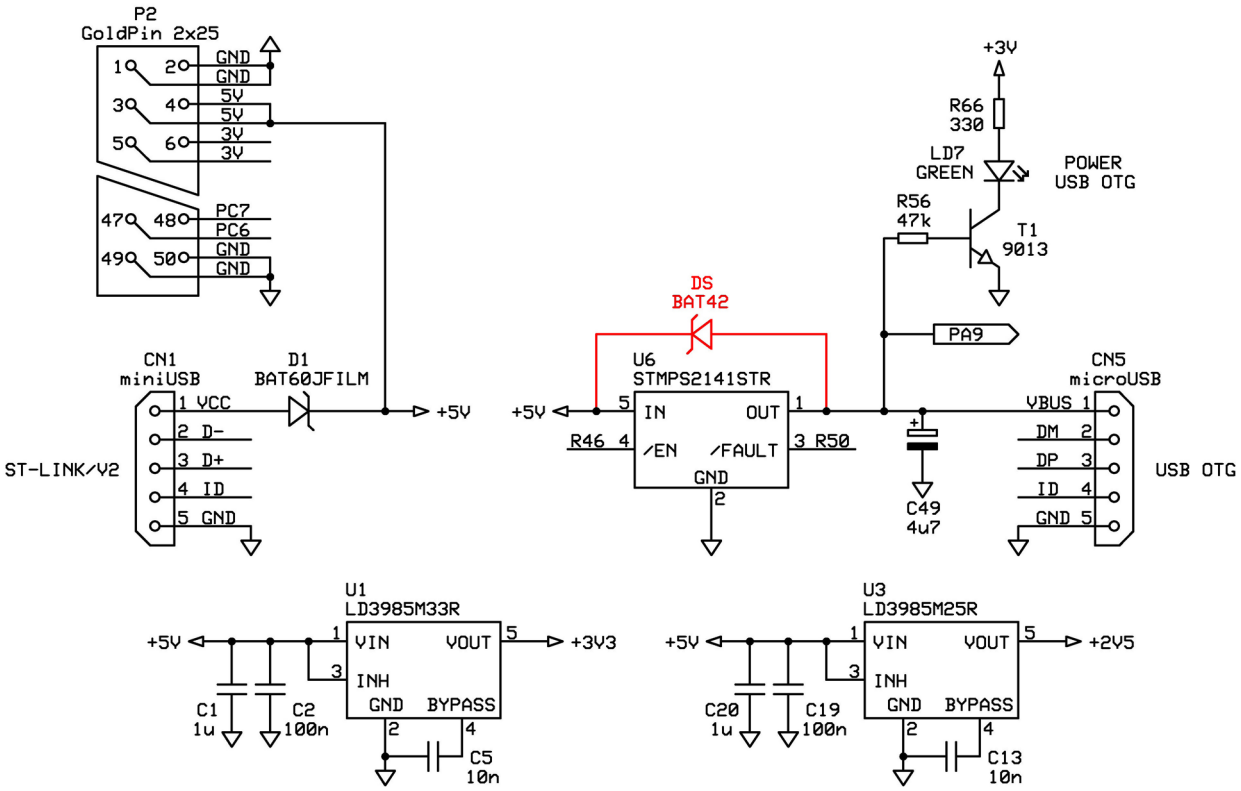
Proponowana modyfikacja obwodu zasilania płytki STM32F4DISCOVERY w żaden sposób nie zmienia ani też nie upośledza oryginalnych funkcji interfejsu USB OTG oraz układu kontroli jego zasilania (tj. układu U6). Przerobiona płytka nadal może służyć do prac rozwojowych związanych z obsługą urządzeń peryferyjnych USB podłączanych do gniazda CN5.

Budowa

Do wykonania działającego analizatora stanów logicznych wystarczy sama płytka STM32F4DISCOVERY. Wyprowadzenia sygnałów PIXCLKout, HSYNCout i VSYNCout kontrolujących moduł DCMI zostały dobrane

Tabela 1. Sygnalizatory stanu pracy analizatora stanów logicznych STM32F4Discovery LA

Lp	Dioda	Kolor	Sygnalizowany stan
1	LD4	zielony	gotowość przyrządu do rozpoczęcia pomiaru
2	LD3	pomarańczowy	pomiar (gromadzenie próbek sygnału w oczekiwaniu na wyzwolenie)
3	LD5	czerwony	pomiar (gromadzenie próbek sygnału po wyzwoleniu) lub transfer danych do komputera PC
4	LD6	niebieski	aktywność programu analizatora



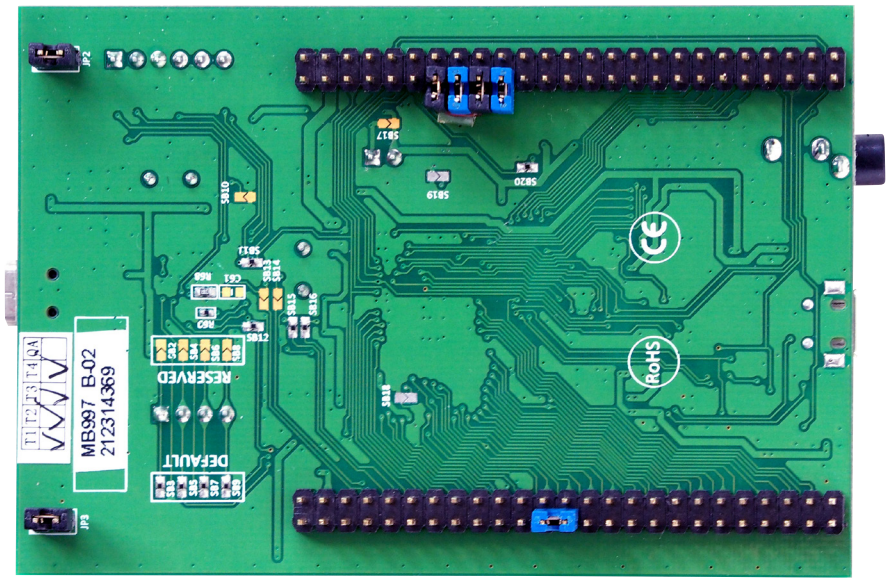
Rysunek 5. Obwód zasilania +5 V płytki STM32F4DISCOVERY

w układzie tak, aby do minimum ograniczyć długości przewodów krosujących i ich krzyżowanie się na płytce. Do wykonania wszystkich połączeń w układzie wystarczy pięć zworek. Połączenie sygnału VSYNCout z VSYNC realizuje zworka założona bezpośrednio na piny PB5 i PB7 złącza szpilkowego P2 modułu Discovery. Niestety, wyprowadzenia sygnałów PIXCLKout (pin PA2 złącza P1) i PIXCLK (pin PA6 złącza P1) nie mogą być bezpośrednio połączone zworką, ponieważ te wyprowadzenia rozdziela pin PA4 będący wejściem sygnału HSYNC. Najprościej połączenie pinu PA2 z PA6 można wykonać za pomocą dwóch zworek nałożonych z jednej strony na łączone piny, a z drugiej strony na wygięty w kształcie litery U odcinek srebrzanki o średnicy ok. 0,8…1,0 mm. W taki sam sposób należy zrealizować połączenie sygnału HSYNCout (pin PC4 złącza P1) z HSYNC (pin PA4 złącza P1). W celu uniknięcia zwarć na płytce dobrze jest zabezpieczyć odsłonięte odcinki srebrzanki kawałkami koszulki izolacyjnej. Widok płytki STM32F4DISCOVERY z wykonanymi połączeniami sygnałów sterujących modułem DCMI pokazano na **fotografii 6**.

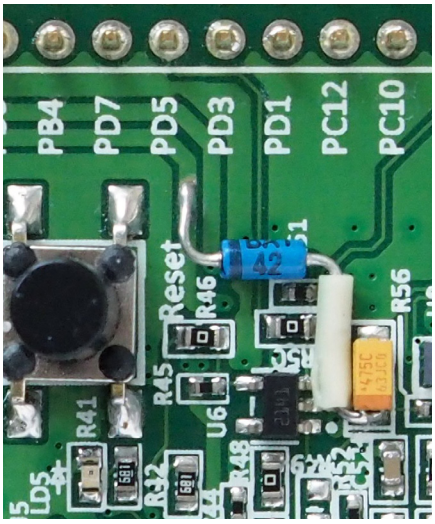
Diodę DS, pozwalającą na zasilanie modułu STM32F4DISCOVERY przez gniazdo portu USB OTG (tj. złącze CN5), najłatwiej jest zamontować na górnej stronie płytki, lutując wyprowadzenie anody do pinu + kondensatora tantalowego C49 oraz lutując wyprowadzenie katody do ścieżki +5 V. Odsłonięte wyprowadzenia montowanej diody dobrze jest zabezpieczyć odcinkami koszulki izolacyjnej. Szczegóły montażu diody DS przedstawia **fotografia 7**.

Oprogramowanie

Strukturę projektu programu analizatora stanów logicznych pokazuje **rysunek 5**. Program napisano w języku C i skompilowano za pomocą GCC. Ze względu na dążenie do uzyskania maksymalnej szybkości wykonywania programu kompilację wykonano z włączoną opcją **–O3**, tj. maksymalnej optymalizacji pod kątem szybkości wykonywania kodu programu.



Fotografia 6. Krosowanie sygnałów sterujących modułem DCMI na płytce STM32F4DISCOVERY



Fotografia 7. Sposób montażu diody pozwalającej na zasilanie modułu STM32F4DISCOVERY przez gniazdo USB OTG

Program analizatora ma typową budowę. W pierwszym kroku po starcie mikrokontrolera inicjalizowane są wszystkie układy peryferyjne używane w projekcie, tj.:

- Układ RCC generujący przebieg zegarowy jest konfigurowany do wytwarzania przebiegów zegarowych o następujących częstotliwościach: f_{SYSCLK}=144 MHz, f_{AHB}=144 MHz, f_{APB1}=36 MHz, f_{APB2}=72 MHz, $f_{USBOTGFS}$=48 MHz.
- Timer SysTick jest konfigurowany do generowania przerwań zegarowych co 100 ms.
- Układ USB OTG jest konfigurowany do pracy w trybie Full Speed z wyprowadzeniem linii DP i DM na porty PA12 i PA11, a także inicjalizowany jest programowy stos USB i ładowany jest sterownik klasy CDC.
- Układ czasowo-licznikowy TIM9 jest konfigurowany do pracy jako automatyczny licznik taktowany podwojonym sygnałem zegarowym magistrali APB2.
- Układ próbkująco-porównujący TIM9_CC1 jest konfigurowany do pracy w trybie generowania sygnału PWM2 o wypełnieniu 50%, z wyprowadzeniem wyjściowego sygnału na port PA2.
- Porty PD4 i PE3, wyłączające nieużywane układy na płytce Discovery, są konfigurowane jako wyjścia push-pull, low speed.
- Porty PB5 i PC4, sterujące liniami VSYNC i HSYNC, są konfigurowane jako wyjścia push-pull, high speed.
- Układ DCMI jest konfigurowany do pracy w trybie ciągłego odbioru obrazów JPEG, z 8-bitową szyną danych, odczytem danych na narastającym zboczu sygnału PIXCLK i aktywnym stanem wysokim sygnałów HSYNC i VSYNC, z sygnałami kontrolnymi wyprowadzonymi na porty PA6, PA4 i PB7 oraz liniami danych wyprowadzonymi na porty PB6, PC6...PC9 i PE4...PE6.
- Kontroler DMA2 jest konfigurowany do przesyłania strumieniem nr 7 32-bitowych słów z rejestru danych układu DCMI do pamięci SRAM.
- Porty PD12...PD15, sterujące diodami LED, są konfigurowane jako wyjścia push-pull, low speed.

W programie inicjalizacja poszczególnych układów peryferyjnych jest wykonywana w drodze bezpośredniego zapisu ich rejestrów konfiguracyjnych. Szczegóły można poznać, analizując kod źródłowy programu znajdujący się w materiałach dodatkowych (plik *LA_hrdwr.c*).

Listing 1. Pętla główna programu analizatora stanów logicznych

```
// Program main loop
while (1)
{
    // Listen to SUMP client
    if (TM_USB_VCP_Getc(&c) == TM_USB_VCP_DATA_OK)
    {   // if data byte received
        SUMP_service(c);  // process SUMP client command
    }
    // Service logic analyzer state machine
    switch (SUMP_LAcfg.state)
    {
        case LA_IDLE: // wait for SUMP client commands
            break;
        case LA_START:// start sampling inputs
            LA_Start();  // start analyzer hardware
            SUMP_LAcfg.state = LA_PRETRIG;  // collect pretrigger samples
            LED_OFF(LED_G);  // switch on orange LED
            LED_ON(LED_O);
            break;
        case LA_PRETRIG: // collect samples preceding trigger event
            if (LA_CollectSamples() == TRUE) SUMP_LAcfg.state = LA_TRIGGER; // wait for trigger
            break;
        case LA_TRIGGER: // detect trigger
            if (LA_DetectTrigger() == TRUE)
            {
                SUMP_LAcfg.state = LA_POSTTRIG;  // collect posttrigger samples
                LED_OFF(LED_O);  // switch on red LED
                LED_ON(LED_R);
            }
            break;
        case LA_POSTTRIG: // collect samples after trigger event
            if (LA_CollectSamples() == TRUE)
            {
                LA_Stop();  // stop analyzer hardware
                SUMP_sendSamples();  // send captured samples to SUMP client
                SUMP_LAcfg.state = LA_IDLE;  // wait for new command
                LED_OFF(LED_R);  // switch on green LED
                LED_ON(LED_G);
            }
            break;
        case LA_RESET:// reset logic analyzer
        default:  // or unknown state
            LA_Stop();  // stop analyzer hardware
            SUMP_LAcfg.state = LA_IDLE;  // return to known state
            LED_OFF(LED_O);  // switch on green LED
            LED_OFF(LED_R);
            LED_ON(LED_G);
            break;
    }
}
```

Działanie analizatora stanów logicznych jest kontrolowane przez maszynę stanów obsługiwaną w pętli głównej programu (**listing 1**). W stanie spoczynkowym LA_IDLE praca programu ogranicza się jedynie do analizy komend przesyłanych z komputera PC. Odbiór komendy startu pomiaru powoduje uruchomienie układów peryferyjnych mikrokontrolera, realizujących próbkowanie wejść (tj. układu TIM9, DCMI i DMA2 – **listing 2**) i przejście analizatora do wstępnego gromadzenia próbek w buforze (tj. stanu

Listing 2. Funkcja inicjalizacji pomiaru

```
/*Name         :  LA_Start
  Description  :  Start sampling LA inputs at sampling frequency defined in LA
                  configuration
  Argument(s)  :  none
  Return value :  none */
void LA_Start(void)
{
    // Init sample buffor index and set number of pretrigger samples
    SUMP_LAcfg.smplBufIdx = 0;
    SUMP_LAcfg.smplCntr = SUMP_LAcfg.smplQtty - SUMP_LAcfg.posttrigQtty;
    // Set sampling frequency and start sampling clock generator
    // - set timer prescaler
    SMPL_TIM->PSC = SUMP_LAcfg.freqDiv / 0x10000;
    // - set timer period
    SMPL_TIM->ARR = SUMP_LAcfg.freqDiv / (SMPL_TIM->PSC + 1) - 1;
    // - set pulse width to 0.5
    SMPL_TIM->CCR1 = (SMPL_TIM->ARR + 1) / 2;
    // - start timer
    SET_BIT(SMPL_TIM->CR1, TIM_CR1_CEN);
    // Program and enable DCMI DMA transfers
    // - clear interrupt requests from previous DMA transfer
    DMA2->HIFCR = DMA_HIFCR_CTCIF7 | DMA_HIFCR_CHTIF7 | DMA_HIFCR_CTEIF7 |
                  DMA_HIFCR_CDMEIF7 | DMA_HIFCR_CFEIF7;
    // - set number of DMA transfered 32-bit data words
    DMA2_Stream7->NDTR = SUMP_SMPL_BUF_SIZE;
    // - set peripheral register address
    DMA2_Stream7->PAR = (uint32_t)&DCMI->DR;
    // - set memory address
    DMA2_Stream7->M0AR = (uint32_t)SUMP_smplBuf;
    // - enable DMA channel
    SET_BIT(DMA2_Stream7->CR, DMA_SxCR_EN);
    // Enable inputs capture by DCMI unit
    // - clear interrupt requests from previous inputs capture
    DCMI->ICR = DCMI_ICR_LINE_ISC | DCMI_ICR_VSYNC_ISC | DCMI_ICR_ERR_ISC |
                DCMI_ICR_OVF_ISC | DCMI_ICR_FRAME_ISC;
    // - enable DCMI
    SET_BIT(DCMI->CR, DCMI_CR_CAPTURE);
    // - wait 10ms for DCMI start
    dwt_Delay(10000);
    // Generate „Start of the JPEG frame" event (i.e. falling edge of VSYNC)
    SMPL_VSYNC_GPIO->BSRRH = 1 << SMPL_VSYNC_PIN;
    // Set „Data valid" (i.e. HSYNC = LOW)
    SMPL_HSYNC_GPIO->BSRRH = 1 << SMPL_HSYNC_PIN;
}
```

Listing 3. Funkcja kontroli liczby zgromadzonych w buforze próbek sygnału mierzonego

```
/*Name         :  LA_CollectSamples
  Description  :  Collect requested number of samples
                  Quantity of samples to capture is defined in SUMP_LAcfg.smplCntr
  Argument(s)  :  none
  Return value :  TRUE if all requested samples captured, otherwise FALSE */
uint8_t LA_CollectSamples(void)
{
    uint32_t smplQtty;                  // number of new samples in buffer

    // Process new input samples in buffer
    while ((smplQtty = (2*SUMP_SMPL_BUF_SIZE - DMA2_Stream7->NDTR - SUMP_LAcfg.smplBufIdx)
                       %SUMP_SMPL_BUF_SIZE))
    {
        // Still collecting samples
        if (smplQtty < SUMP_LAcfg.smplCntr)
        {
            // Update sample buffer index and samples counter
            SUMP_LAcfg.smplBufIdx = (SUMP_LAcfg.smplBufIdx + smplQtty) % SUMP_SMPL_BUF_SIZE;
            SUMP_LAcfg.smplCntr -= smplQtty;
        }
        // All requested samples has been collected
        else
        {
            // Update sample buffer index and samples counter
            SUMP_LAcfg.smplBufIdx = (SUMP_LAcfg.smplBufIdx + SUMP_LAcfg.smplCntr) % SUMP_SMPL_BUF_SIZE;
            SUMP_LAcfg.smplCntr = 0;
            return TRUE;
        }
    }
    return FALSE;
}
```

LA_PRETRIG). W tym czasie w buforze zapisywane są próbki sygnału poprzedzające moment wyzwolenia pomiaru (**listing 3**).

Po zgromadzeniu zadanej przez program sterujący liczby próbek, kolejne próbki sygnału wejściowego zapisywane w buforze są analizowane pod kątem wykrycia warunku wyzwolenia pomiaru (stan LA_TRIGGER). Ze względu na dążenie do uzyskania maksymalnej szybkości analizy, sprawdzanie warunku wyzwolenia jest wykonywane na słowach 32-bitowych (**listing 4**). Jednocześnie przetwarzane są więc cztery kolejne próbki sygnału mierzonego. Taka metoda wykrywania warunku wyzwolenia, chociaż szybka, skutkuje jednak niewielkimi fluktuacjami momentu wyzwolenia pomiaru w zakresie od 0 do +3 próbek, w zależności od tego, w którym bajcie analizowanego 32-bitowego słowa został spełniony warunek wyzwolenia pomiaru. W praktyce jednak nie stanowi to wielkiego problemu. Po wykryciu wyzwolenia pomiaru program oczekuje na zgromadzenie w buforze ustawionej przed pomiarem liczby próbek (list. 3), po czym wyłącza układy peryferyjne mikrokontrolera realizujące próbkowanie wejść (**listing 5**) i przesyła zgromadzone dane do programu sterującego (stan LA_POSTTRIG). Niezależnie od ustawionego w programie sterującym rozmiaru bufora próbek, w analizatorze próbki są gromadzone zawsze w buforze o pełnym rozmiarze 128 kB. Tylko do programu sterującego jest wysyłany blok danych o zadanej wielkości. Zakończenie transmisji powoduje przejście analizatora do stanu spoczynkowego LA_IDLE i oczekiwanie na komendę rozpoczęcia nowego pomiaru.

Listing 4. Funkcja detekcji warunku wyzwolenia pomiaru

```
/*Name         :  LA_DetectTrigger
  Description  :  Detect trigger event
  Argument(s)  :  none
  Return value :  TRUE if trigger condition detected, otherwise FALSE */
uint8_t LA_DetectTrigger(void)
{
    uint32_t smplWrd;                   // four consecutive 8-bit samples

    // Process new input samples in buffer
    while ((SUMP_SMPL_BUF_SIZE - DMA2_Stream7->NDTR) != SUMP_LAcfg.smplBufIdx)
    {
        // Load four input samples and check trigger bits
        smplWrd = SUMP_smplBuf[SUMP_LAcfg.smplBufIdx];
        smplWrd ^= SUMP_LAcfg.trigValue;
        smplWrd &= SUMP_LAcfg.trigMask;
        // Detect trigger event
        if (((smplWrd & 0x000000FF) == 0) || ((smplWrd & 0x0000FF00) == 0) ||
            ((smplWrd & 0x00FF0000) == 0) || ((smplWrd & 0xFF000000) == 0))
        {
            // Trigger detected - set number of samples to collect after trigger
            SUMP_LAcfg.smplCntr = SUMP_LAcfg.posttrigQtty;
            return TRUE;
        }
        // No trigger - move to the next sample
        SUMP_LAcfg.smplBufIdx = (SUMP_LAcfg.smplBufIdx + 1) % SUMP_SMPL_BUF_SIZE;
    }
    return FALSE;
}
```

Listing 5. Funkcja zakończenia pomiaru

```
/*Name         :  LA_Stop
  Description  :  Stop sampling input signals
  Argument(s)  :  none
  Return value :  none */
void LA_Stop(void)
{
    // Set „DCMI: Data not valid” (i.e. HSYNC = HIGH)
    SMPL_HSYNC_GPIO->BSRRL = 1 << SMPL_HSYNC_PIN;
    // Disable DCMI DMA transfers
    CLEAR_BIT(DMA2_Stream7->CR, DMA_SxCR_EN);
    while (READ_BIT(DMA2_Stream7->CR, DMA_SxCR_EN));
    // Update index of the last captured sample
    SUMP_LAcfg.smplBufIdx = SUMP_SMPL_BUF_SIZE - DMA2_Stream7->NDTR;
    // Disable inputs capture by DCMI unit
    // - clear CAPTURE bit
    CLEAR_BIT(DCMI->CR, DCMI_CR_CAPTURE);
    // - set „DCMI: End of the JPEG frame” (i.e. rising edge of VSYNC)
    SMPL_VSYNC_GPIO->BSRRL = 1 << SMPL_VSYNC_PIN;
    // - wait for CAPTURE bit reset
    while (READ_BIT(DCMI->CR, DCMI_CR_CAPTURE));
    // Stop sampling clock generator
    // - stop timer
    CLEAR_BIT(SMPL_TIM->CR1, TIM_CR1_CEN);
    // - clear timer
    SMPL_TIM->CNT = 0;
}
```

Bufor próbek analizatora stanów logicznych jest ulokowany w pamięci SRAM mikrokontrolera, gdzie zajmuje obszar 128 kB. Jak łatwo sprawdzić, jest to cała dostępna pamięć SRAM1 i SRAM2 mikrokontrolera STM32F407. Wszystkie zmienne oraz stos są z kolei umieszczone w pamięci CCM-RAM mikrokontrolera (tj. Core Coupled Memory RAM). Jest to specjalny blok pamięci RAM o rozmiarze 64 kB implementowany w niektórych zaawansowanych układach rodziny STM32F4. Pamięć ta jest dołączona tylko do szyny D-bus i dostępna jedynie dla rdzenia. Inne układy peryferyjne, takie jak np. kontrolery DMA, nie mają do tej pamięci dostępu. Z pamięci CCMRAM nie może też być wykonywany kod programu. Taka metoda rozmieszczenia danych programu analizatora w pamięci RAM wymusiła konieczność modyfikacji standardowego

skryptu linkera, jak również zmian w domyślnym kodzie startowym mikrokontrolera STM32F407. Polegały one głównie na rozbudowie kodu startowego o inicjalizację zmiennych w pamięci CCMRAM. Osoby zainteresowane szczegółami tych modyfikacji powinny przeanalizować skrypt linkera oraz kod startowy mikrokontrolera znajdujące się w materiałach dodatkowych (pliki *stm32f4xx_flash.ld* oraz *startup_stm32f4xx.s*).

Obsługę interfejsu USB OTG w mikrokontrolerze STM32F407 wykonano na bazie biblioteki *tm_stm32f4_usb_vcp* wirtualnego portu szeregowego dla płytek STM32F4 Discovery i STM32F426 Discovery, której autorem jest Tilen Majerle [3]. Dla potrzeb projektu analizatora stanów logicznych oryginalna biblioteka została dość znacznie zmodyfikowana. Zmiany dotyczyły przede wszystkim dostosowania kodu biblioteki do kompilacji przez kompilator GCC oraz objęły usunięcie zależności tej biblioteki od wychodzącej już z użycia biblioteki *STM32F4xx Standard Peripheral Library*. Przy okazji okazało się, że oryginalna biblioteka *tm_stm32f4_usb_vcp* ma problemy z transmisją dużych bloków danych, co również zostało naprawione, łącznie z kilkoma innymi wykrytymi drobnymi błędami. Po inicjalizacji biblioteki *tm_stm32f4_usb_vcp* przy starcie programu mikrokontrolera cała obsługa interfejsu USB jest realizowana w przerwaniach.

Sam analizator stanów logicznych po podłączeniu do portu USB komputera PC enumeruje się w systemie jako wirtualny port szeregowy. Do jego obsługi konieczne jest

Tabela 2. Komendy protokołu SUMP obsługiwane przez analizator stanów logicznych STM32F4Discovery LA v.1.0

Lp	Komenda	Kod	Dane	Odpowiedź analizatora
1	Reset ustawień analizatora	0x00	Brak	Brak
2	Start pomiaru	0x01	Brak	Brak
3	Identyfikacja analizatora	0x02	Brak	1ALS
4	Odczyt metadanych analizatora	0x04	Brak	0x01 – identyfikator pola nazwy urządzenia STM32F4Discovery LA v.1.0 – nazwa 0x00 – znacznik końca nazwy
				0x02 – identyfikator pola wersji 1.0 – numer wersji 0x00 – znacznik końca wersji
				0x21 – identyfikator pola rozmiaru dostępnej pamięci próbek 0x00, 0x00, 0x02, 0x00 – rozmiar pamięci w bajtach (tj. 128kB
				0x23 – identyfikator pola maksymalnej częstotliwości próbkowania 0x00, 0x44, 0x95, 0x08 – maksymalna częstotliwość próbkowania w Hz (tj. 144 MHz)
				0x40 – identyfikator pola liczby wejść analizatora 0x08 – liczba wejść
				0x41 – identyfikator pola skróconego numeru wersji 0x02 – skrócony numer wersji
				0x00 – znacznik końca metadanych
5	Maska wyzwalania pomiaru (stan 0)	0xC0	Bajt 1 – maski bitowe dla wejść In8...In1 Bajt 2 – ignorowany (oryginalnie maski bitowe dla wejść In16...In9) Bajt 3 – ignorowany (oryginalnie maski bitowe dla wejść In24...In17) Bajt 4 – ignorowany (oryginalnie maski bitowe dla wejść In32...In25)	Brak
6	Wartość wyzwalająca pomiar (stan 0)	0xC1	Bajt 1 – wartości bitowe dla wejść In8...In1 Bajt 2 – ignorowany (oryginalnie wartości bitowe dla wejść In16...In9) Bajt 3 – ignorowany (oryginalnie wartości bitowe dla wejść In24...In17) Bajt 4 – ignorowany (oryginalnie wartości bitowe dla wejść In32...In25)	Brak
7	Wartość dzielnika częstotliwości	0x80	Bajt 1 – LSB stopnia podziału dzielnika częstotliwości Bajt 2 – drugi bajt stopnia podziału dzielnika częstotliwości Bajt 3 – MSB stopnia podziału dzielnika częstotliwości Bajt 4 – ??? – ignorowany	Brak
8	Liczba próbek i opóźnienie wyzwolenia pomiaru	0x81	Bajt 1 – LSB liczby próbek N Bajt 2 – MSB liczby próbek N Bajt 3 – LSB opóźnienia wyzwolenia Bajt 4 – MSB opóźnienia wyzwolenia	Brak
9	Zarejestrowane stany logiczne wejść In8...In1	Brak	Brak	Blok N bajtów zawierających kolejne próbki stanów logicznych wejść In8...In1

Listing 6. Plik konfiguracyjny analizatora STM32F4Discovery LA dla programu OLS

```
# Configuration for STM32F4 Discovery Logic Analyzer profile
# The short (single word) type of the device described in this profile
device.type = STM32F4DISCO
# A longer description of the device
device.description = STM32F4 Discovery Logic Analyzer
# The device interface, SERIAL only
device.interface = SERIAL
# The device's native clockspeed, in Hertz.
device.clockspeed = 144000000
# The clockspeed used in the divider calculation, in Hertz. Defaults to 100MHz as most devices appear to use this.
device.dividerClockspeed = 144000000
# Whether or not double-data-rate is supported by the device (also known as the „demux"-mode).
device.supports_ddr = false
# Supported sample rates in Hertz, separated by comma's
device.samplerates = 1000, 2000, 5000, 10000, 20000, 50000, 100000, 200000, 500000, 1000000, 2000000, 6000000, 12000000, 24000000,
48000000
# What capture clocks are supported
device.captureclock = INTERNAL
# The supported capture sizes, in bytes
device.capturesizes = 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072
# Whether or not the noise filter is supported
device.feature.noisefilter = false
# Whether or not Run-Length encoding is supported
device.feature.rle = false
# Whether or not a testing mode is supported
device.feature.testmode = false
# Whether or not triggers are supported
device.feature.triggers = true
# The number of trigger stages
device.trigger.stages = 1
# Whether or not „complex" triggers are supported
device.trigger.complex = false
# The total number of channels usable for capturing
device.channel.count = 8
# The number of channels groups, together with the channel count determines the channels per group
device.channel.groups = 1
# Whether the capture size is limited by the enabled channel groups
device.capturesize.bound = false
# Which numbering does the device support
device.channel.numberingschemes = DEFAULT
# Is a delay after opening the port and device detection needed? (0 = no delay, >0 = delay in milliseconds)
device.open.portdelay = 0
# The receive timeout for the device (in milliseconds, 100 = default, <=0 = no timeout)
device.receive.timeout = 100
# Does the device need a high or low DTR-line to operate correctly? (high = true, low = false)
device.open.portdtr = true
# Which metadata keys correspond to this device profile? Value is a comma-separated list of (double quoted) names...
device.metadata.keys = „STM32F4Discovery LA v.1.0"
# In which order are samples sent back from the device? false = last sample first, true = first sample first
device.samples.reverseOrder = true
###EOF###
```

zainstalowanie w systemie komputera PC sterownika *VCP_Vxxx_Setup.exe*, który jest dostępny do pobrania na stronie firmy ST-Microelectronics. Widok prawidłowo zainstalowanego w systemie MSWindows analizatora STM32F4DISCOVERY LA przedstawia **rysunek 6**.

Protokół komunikacyjny

Analizator stanów logicznych komunikuje się z komputerem PC za pomocą protokołu SUMP [4, 5]. Pierwotnie ten protokół był używany w analizatorze stanów logicznych o nazwie *Sump Logic Analyzer*, lecz od tej pory jest bardzo często implementowany w analizatorach stanów logicznych tworzonych na podstawie licencji open source. Główną zaletą takiego podejścia jest możliwość użycia w charakterze programu sterującego analizatorem, i obrazującego zarejestrowane sygnały, jednego z wielu programów klienckich, napisanych dla analizatora *Sump* lub też późniejszych, wywodzących się z niego konstrukcji.

Sterowanie analizatorem za pomocą protokołu SUMP jest realizowane w nim za pomocą 1-bajtowych komend przesyłanych z programu sterującego do analizatora łączem szeregowym. Komendom ustawiającym parametry pracy analizatora towarzyszą dodatkowo 4 bajty danych, zawierające programowane ustawienia. Zwrotnie analizator odpowiada tylko na komendy żądania identyfikacji, przesyłając do programu sterującego swoje dane identyfikacyjne. Poza tym analizator po zakończeniu pomiaru przesyła do programu sterującego blok danych zawierający zarejestrowane stany wejść pomiarowych.

Ponieważ prezentowany analizator ma skromniejsze możliwości niż analizator *Sump*, nie było potrzeby implementacji pełnego protokołu komunikacyjnego oryginalnego analizatora. **Tabela 2** zawiera listę komend protokołu SUMP obsługiwanych przez analizator STM32F4Discovery LA v.1.0.

Program dla komputera PC

W roli programu sterującego analizatorem STM32F4Discovery LA użyto programu o nazwie *OpenBench LogicSniffer* (w skrócie OLS), którego autorem jest Jan Willem Janssen [6]. Jest to chyba najlepszy program klienta SUMP dostępny na licencji open source. Ostatnia opublikowana wersja tego programu nosi numer 0.9.7.2.

Program OLS był już prezentowany na łamach „Elektroniki Praktycznej" w ramach opisu projektu analizatora stanów logicznych XMC2Go LA [7]. Warto jednak przypomnieć, że program OLS został napisany w języku Java, dzięki czemu może być uruchamiany w wielu systemach operacyjnych, także w systemach MS Windows i Linux. Jak wszystkie programy napisane w Javie, do poprawnej pracy program OLS wymaga zainstalowanej w systemie komputera PC wirtualnej maszyny Javy (tzw. JRE – Java Runtime Environment).

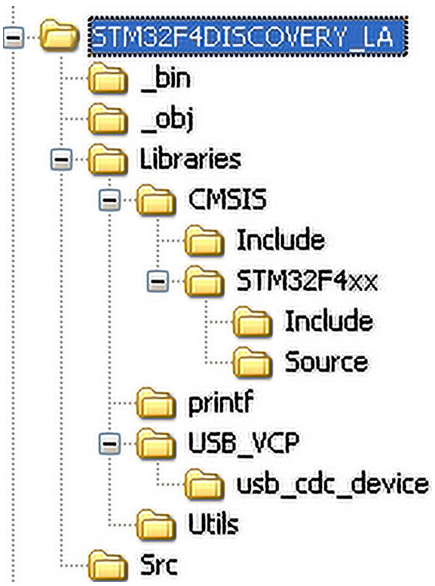
Sam program OLS nie wymaga instalacji. Po pobraniu programu ze strony projektu OLS i zapisaniu go na dysku w wybranym katalogu należy jedynie umieścić w folderze */plugins* programu plik konfiguracyjny o nazwie *ols.profile-stmf4discovery.cfg* (**listing 6**), zawierający dane identyfikacyjne analizatora STM32F4Discovery LA i deklaracje dozwolonych parametrów pracy tego przyrządu. Dzięki tym informacjom program OLS jest w stanie automatycznie rozpoznać dołączony moduł analizatora i poprawnie go skonfigurować. Przy okazji

interfejs graficzny programu OLS dostosowuje się do możliwości dołączonego modułu, udostępniając użytkownikowi tylko te funkcje, które są obsługiwane przez podłączony analizator.

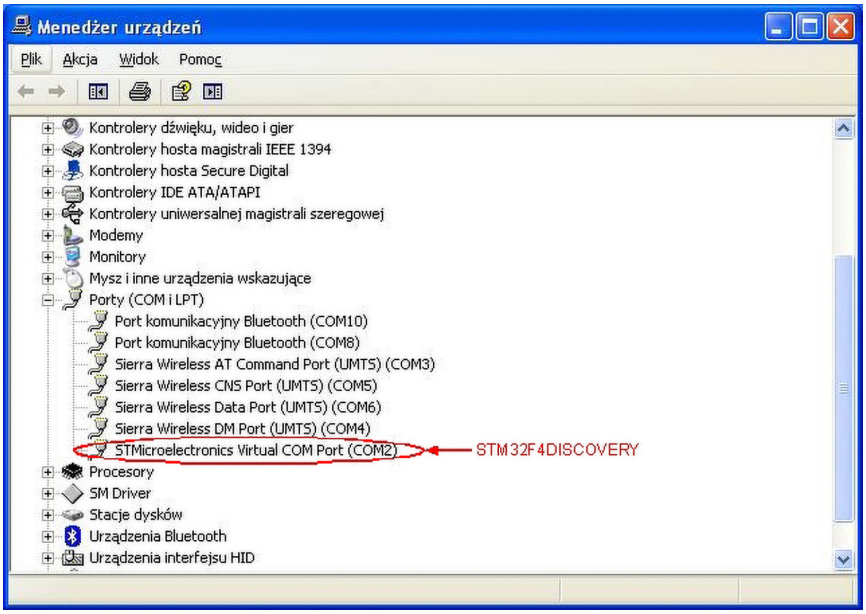
Program OLS jest uruchamiany za pośrednictwem pliku *run.bat*, znajdującego się w głównym folderze programu. Po starcie aplikacji konieczne jest skonfigurowanie połączenia z modułem STM32F4DISCOVERY. W tym celu w zakładce *Connection* okna *OLS Capture settings* (**rysunek 7**) należy wybrać numer wirtualnego portu COM, pod którym jest widoczny w systemie analizator STM32F4Discovery LA (rys. 6). Ponieważ komunikacja programu OLS z analizatorem fizycznie odbywa się przez łącze USB, prędkość transmisji portu COM w konfiguracji połączenia nie ma znaczenia. Może być ona ustawiona na dowolną wartość.

Sprawność połączenia programu OLS z analizatorem może zostać zweryfikowana przez wciśnięcie przycisku *Show device metadata*. W odpowiedzi w zakładce *Connection* powinny zostać wyświetlone dane podłączonego modułu, tj. nazwa urządzenia, numer wersji jego oprogramowania oraz numer obsługiwanego protokołu komunikacyjnego (rys. 7). Dodatkowo pole typu analizatora powinno zostać automatycznie ustawione na wartość *STM32F4 Discovery Logic Analyzer*. W przypadku błędnego rozpoznania przez program OLS podłączonego modułu analizatora typ obsługiwanego modułu może zostać zmieniony przez użytkownika ręcznie.

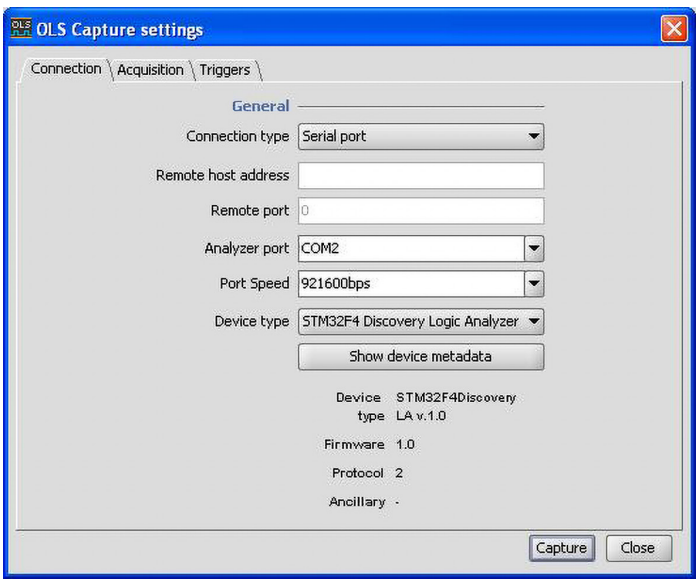
Możliwości programu OLS oraz jego obsługa były już omawiane we wspomnianym wcześniej opisie analizatora XMC 2Go [7]. Czytelnicy zainteresowani tym tematem powinni więc sięgnąć do tego artykułu. **Rysunki 8**, **9** i **10** przedstawiają zrzuty ekranu



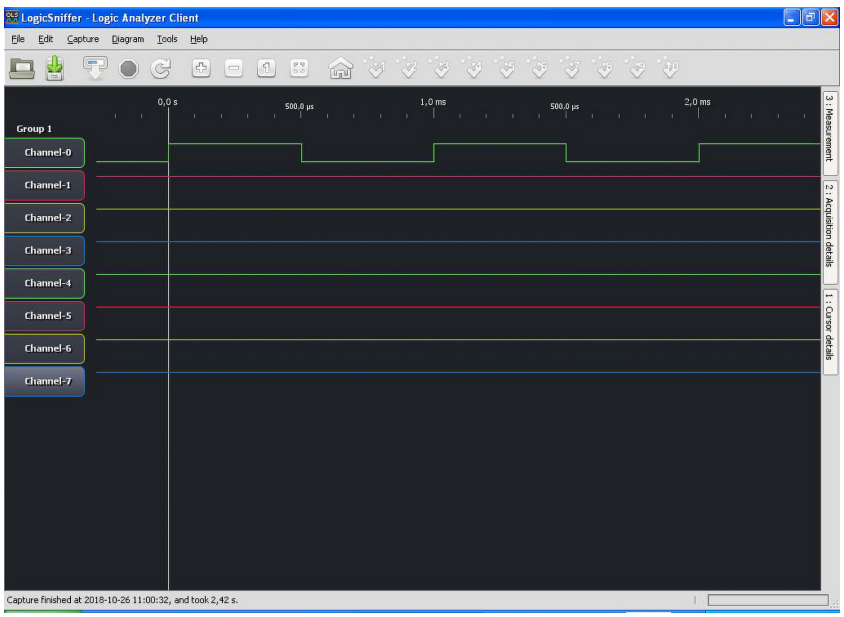
Rysunek 8. Struktura projektu programu analizatora



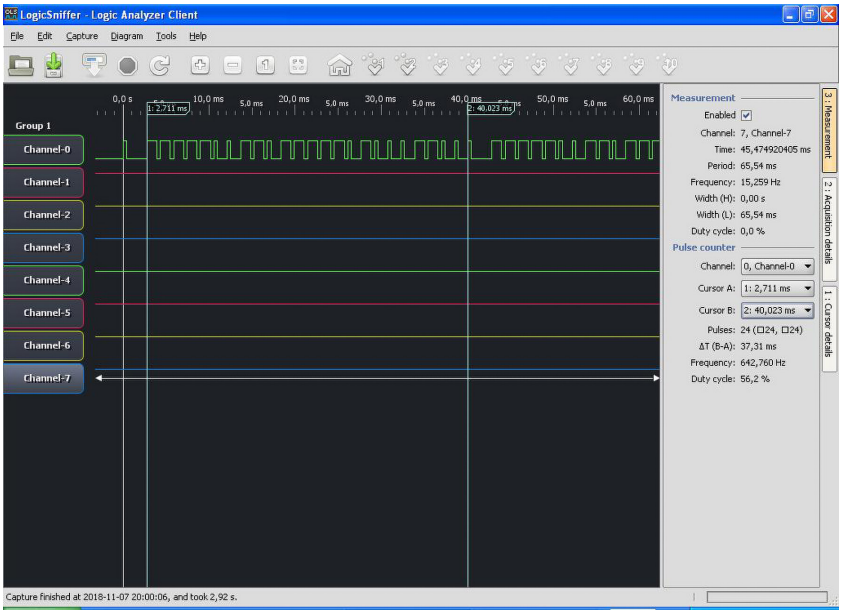
Rysunek 9. Identyfikacja analizatora STM32F4DISCOVERY LA w systemie MS Windows



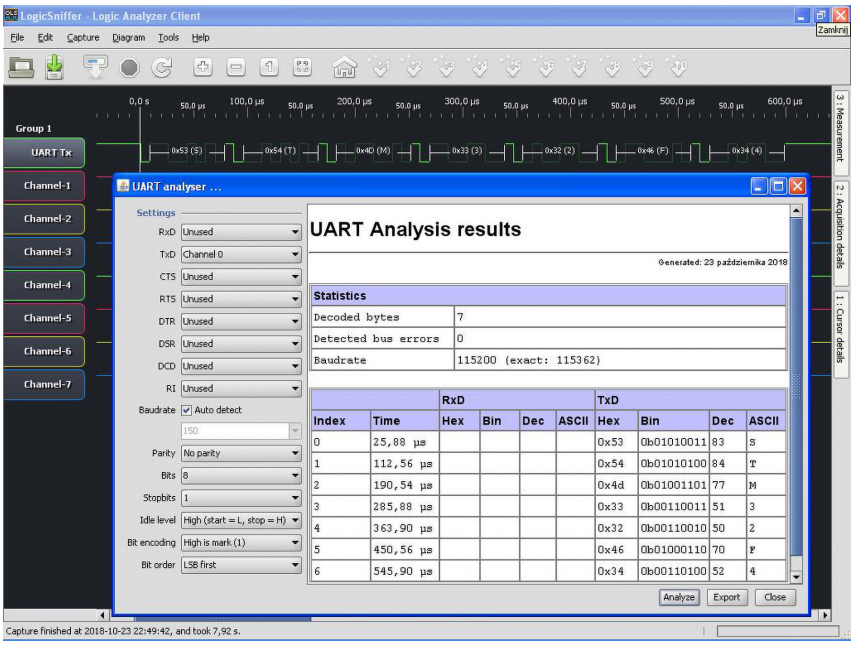
Rysunek 10. Zakładka konfiguracji połączenia z modułem analizatora stanów logicznych



Rysunek 11. Sygnał o częstotliwości 1 kHz i wypełnieniu 50% (fsample = 48 MHz, Nbuf = 128 kB)



Rysunek 12. Sygnał pilota radiowego sterującego gniazdkiem sieciowym (fsample = 1 MHz, Nbuf = 64 kB)



Rysunek 13. Automatyczna analiza sygnału nadajnika UART (fsample = 24 MHz, Nbuf = 128 kB)

programu OLS podczas prezentacji i analizy przykładowych sygnałów zarejestrowanych przez analizator STM32F4Discovery LA v.1.0.

Alternatywne rozwiązania

Przedstawiony projekt nie jest jedynym możliwym sposobem użycia zestawu STM32F4DISCOVERY w roli analizatora stanów logicznych. W Internecie można znaleźć opisy innych układów tego typu. Zazwyczaj próbkowanie wejść jest w nich wykonywane albo przez rdzeń mikrokontrolera STM32F407, odczytujący stany portu wejściowego w pętli lub w przerwaniach i zapisujący je w pamięci SRAM, albo przez kontroler DMA, bezpośrednio przesyłający stany portu wejściowego do pamięci SRAM. Analizatory te osiągają maksymalną częstotliwość próbkowania wejść na poziomie kilku – kilkunastu MHz i rzadko przekraczają barierę 20 MHz.

Niewątpliwie najbardziej interesującym przykładem wykorzystania zestawu uruchomieniowego mikrokontrolera w roli analizatora stanów logicznych jest produkt firmy Sysprogs o nazwie *Analyzer2Go* [8]. Jest to komercyjne oprogramowanie przekształcające zestawy demonstracyjne mikrokontrolerów STM32 w analizatory stanów logicznych. Aktualnie obsługuje ono 8 rodzajów płytek z rodzin Nucleo i Discovery. W przypadku użytego w prezentowanym projekcie modułu STM32F4DISCOVERY pozwala ono na uzyskanie następujących maksymalnych częstotliwości próbkowania mierzonych sygnałów:

- Próbkowanie bez kompresji danych: 42 MHz@1,6 ms (rozmiar bufora próbek równy 64 kB).
- Próbkowanie z kompresją danych w buforze: 8,4 MHz.
- Próbkowanie z ciągłą transmisją danych z bufora do komputera PC: 6,5 MHz (max. transfer równy 631 kB/s).

Dla porównania, wspomniany we wstępie popularny analizator stanów logicznych *Saleae* pozwala na próbkowanie wejść z maksymalną częstotliwością 24 MHz.

Jak więc widać, przedstawiony w artykule analizator STM32F4DISCOVERY LA v.1.0 ze swoją maksymalną częstotliwością próbkowania wejść równą 48 MHz i maksymalnym rozmiarem bufora próbek równym 128 kB prezentuje się na tle analogicznych rozwiązań bardzo korzystnie i tym samym może stanowić cenne uzupełnienie pracowni elektronika zajmującego się układami cyfrowymi.

Aleksander Borysiuk
alex_priv@wp.pl

Bibliografia

1. Discovery kit with STM32F407VG MCU. User manual UM1472, STMicroelectronics, 2017
2. Tomasz Starak, *Tanio, taniej. STM32F4Discovery… Cortex-M4 w pełnej okazałości*, EP 11/2011
3. Tilen Majerle, USB virtual COM port library for STM32F4 Discovery and STM32F429 Discovery boards, ***http://bit.ly/2Gqu0Uq***
4. SUMP Communications Protocol, ***http://bit.ly/2SMh6RG***
5. The Logic Sniffer's extended SUMP protocol, ***http://bit.ly/2EmjXwI***
6. L'XTREME, OLS Client, ***http://bit.ly/2PC2EK5***
7. Aleksander Borysiuk, *Zestaw demonstracyjny XMC 2Go jako analizator stanów logicznych*, EP 3/2015
8. Ivan Shcherbakov, How we turned 8 popular STM32 boards into powerful logic analyzers, Sysprogs, 2017, ***http://bit.ly/2SMmqEL***