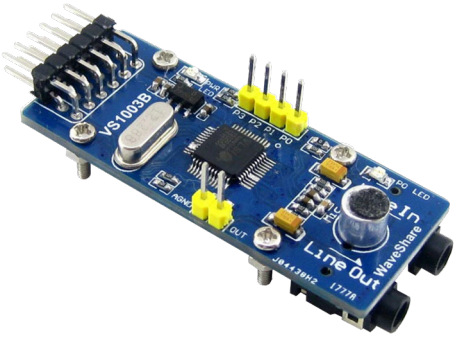
Moduł VS1003B jako automat informacyjny

Głównym elementem modułu VS1003B jest scalony dekoder i koder plików dźwiękowych. Modułu można użyć jako części składowej odtwarzacza „empetrójek". Może też służyć do nagrywania dźwięku mowy z jakością dyktafonu. Jednak jego możliwości pozwalają na coś ciekawszego – zbudowanie urządzenia generującego na żądanie odpowiednie komunikaty słowne, ostrzeżenia lub informacje.



Na rynku dostępne są podobne moduły różniące się szczegółami budowy. Bazują na tym samym układzie scalonym VS1003B, więc ich możliwości są podobne. Zastosowany w projekcie moduł firmy WaveShare ma następujące parametry:

- wyjście dźwięku stereofonicznego wyprowadzone na gniazdo typu mini jack. Wyjście można obciążyć głośnikiem lub słuchawkami o oporności 30 Ω,
- linię wejściową dla dźwięku monofonicznego o amplitudzie w zakresie od 0 do 2200 mVpp,
- zamontowany mikrofon piezoelektryczny jako alternatywne źródło dźwięku monofonicznego,
- złącze szpilkowe 12-pinowe do podłączenia cyfrowych sygnałów sterujących pracą modułu,
- możliwość odtwarzania plików dźwiękowych w formatach MP3, WAV, WMA, MIDI,
- kodowanie dźwięku z linii wejściowej albo mikrofonu w trybie ADPCM z możliwością zapisu do pliku WAV,
- zasilanie pojedynczym napięciem 3,3 V,
- maksymalny pobór prądu około 50 mA (zależny od ustawionej głośności).

Komunikacja z modułem

Wymiana danych z modułem jest możliwa poprzez magistralę SPI. Moduł pracuje jako urządzenie podrzędne. Do wyprowadzeń SPI podłączone są dwie wewnętrzne magistrale układu VS1003B: SCI (*Serial Control Interfejs*) i SDI (*Serial Data Interface*). Pierwsza jest używana do zapisu i odczytu z wewnętrznych rejestrów sterujących pracą układu, druga służy do przesyłania danych dekodowanego pliku muzycznego przekształcanych na dźwięki. O tym, do której magistrali będą podłączane wyprowadzenia SPI, decydują stany niskie dwóch sygnałów: xCS aktywuje magistralę SCI, a xDCS aktywuje magistralę SDI.

Na **listingu 1** pokazana została procedura odczytu zawartości 16-bitowego rejestru poprzez magistralę SCI. Stan wysoki sygnału DREQ informuje o gotowości układu VS1003B. Po ustawieniu stanu niskiego na wyprowadzeniu xCS, można przesłać magistralą SPI z kontrolera sterującego komendę odczytu (wartość 0x03) i bajt adresu odczytywanego rejestru. Następnie czytane są dwa bajty ze starszymi i młodszymi bitami rejestru. Na koniec magistrala SCI jest zamykana ustawieniem wysokiego stanu na wyprowadzeniu xCS.

Na **listingu 2** pokazana została procedura zapisu do 16-bitowego rejestru magistralą SCI. Po otwarciu magistrali, przesłaniu komendy zapisu (wartość 0x02) i adresu rejestru, wysyłane są bajty ze starszymi i młodszymi bitami danej. Na koniec magistrala powinna zostać zamknięta.

Z kolei na **listingu 3** pokazana została procedura przesłania przez wewnętrzną magistralę SDI fragmentu odtwarzanego pliku MP3 o rozmiarze do 32 bajtów. Tak jak poprzednio należy poczekać na stan wysoki sygnału DREQ i dopiero wtedy można wysłać stan niski na wyprowadzenie xDCS otwierające dostęp do wewnętrznej magistrali SDI. Rozpoczyna się przesłanie bajtów danych, które są umieszczane w wewnętrznym buforze VS1003B. Po przesłaniu danych należy podać stan wysoki na xDCS, zamykając magistralę SDI.

Połączenia pomiędzy modułem i kontrolerem sterującym

Moduł VS1003B pracuje pod nadzorem zewnętrznego kontrolera wysyłającego sekwencje rozkazów sterujących lub kolejne paczki danych odtwarzanego pliku muzycznego. Linie magistrali SPI, sygnałów sterujących i zasilania wyprowadzone są na 12-stykowe gniazdo grzebieniowe modułu. Na **rysunku 1**

Listing 1. Procedura odczytu zawartości 16-bitowego rejestru poprzez magistralę SCI

```
//odczyt z rejestru 16-bitowego
uint16_t Player1003::ReadSci(uint8_t addr){
uint16_t res;
uint8_t bufor;

    // czekaj na stan wysoki sygnału DREQ
    while (HAL_GPIO_ReadPin(dreq_port, dreq_pin) ==GPIO_PIN_RESET);
    HAL_GPIO_WritePin(cs_port, cs_pin, GPIO_PIN_RESET);//xCS stan niski
    bufor =0x03;                                    //komenda odczytu
    SPI_WysylanieDoVS1003B(&bufor, 1);          //wysłanie kodu komendy do VS1003B
    bufor =addr;                                    //adres rejestru do odczytu
    SPI_WysylanieDoVS1003B(&bufor, 1);          //wysłanie adresu rejestru SCI
    SPI_OdbieranieZVS1003B(&bufor, 1);          //odbiór H bajtu
    res =(uint16_t)bufor<<8;
    SPI_OdbieranieZVS1003B(&bufor, 1);          //odbiór L bajtu
    res =res + bufor;
    HAL_GPIO_WritePin(cs_port, cs_pin, GPIO_PIN_SET);  //xCS stan wysoki
    return res;//procedura zwraca 16 bitową wartość rejestru
}
```

Listing 2. Procedura zapisu do 16-bitowego rejestru magistralą SCI

```
//zapis do rejestru 16-bitowego
void Player1003::WriteSci(uint8_t addr, uint16_t data){
uint8_t bufor;

    // czekaj na stan wysoki sygnału DREQ
    while (HAL_GPIO_ReadPin(dreq_port, dreq_pin) ==GPIO_PIN_RESET);
    HAL_GPIO_WritePin(cs_port, cs_pin, GPIO_PIN_RESET);//xCS stan niski
    bufor =0x02;                                    //komenda zapisu
    SPI_WysylanieDoVS1003B(&bufor, 1);          //wysłanie kodu komendy do VS1003B
    bufor =addr;                                    //adres rejestru do zapisu
    SPI_WysylanieDoVS1003B(&bufor, 1);          //wysłanie adresu rejestru SCI
    bufor =data >>8;
    SPI_WysylanieDoVS1003B(&bufor, 1);          //wysłanie 8 starszych bitów
    bufor =0xff & data;
    SPI_WysylanieDoVS1003B(&bufor, 1);          //wysłanie 8 młodszych bitów
    HAL_GPIO_WritePin(cs_port, cs_pin, GPIO_PIN_SET);   //xCS stan wysoki
}
```

Listing 3. Procedura przesłania poprzez wewnętrzną magistralę SDI

```
//przesłanie fragmentu danych pliku MP3
int Player1003::WriteSdi(const uint8_t *data, uint8_t bytes){
uint8_t i, bufor;

    if (bytes >32) return -1;//błąd, za długa transmisja
    // czekaj na stan wysoki sygnału DREQ
    while (HAL_GPIO_ReadPin(dreq_port, dreq_pin) ==GPIO_PIN_RESET);
    HAL_GPIO_WritePin(dcs_port, dcs_pin, GPIO_PIN_RESET);  //xDCS stan niski
    for (i=0; i<bytes; i++){
        bufor =*data;
        data++;
        SPI_WysylanieDoVS1003B(&bufor,1);
    }
    HAL_GPIO_WritePin(dcs_port, dcs_pin, GPIO_PIN_SET);    //xDCS stan wysoki
    return 0;
}
```

zaznaczono wszystkie niezbędne połączenia pomiędzy modułem a kontrolerem sterującym na płytce NUCLEO-F091. W tym projekcie można zastosować dowolny inny kontroler o pamięci FLASH nie mniejszej niż 256 kB. Do sterowania pracą automatu posłużą łańcuchy tekstowe przesyłane portem USB do płytki NUCLEO. Port służący do łączności ze zintegrowanym na płytce programatorem można także wykorzystać jako interfejs połączony z UART2 kontrolera STM32F091 zamontowanego na płytce NUCLEO. Widoczny na rysunku 1 potencjometr posłuży do demonstracji odczytu przez automat poziomu napięcia podawanego na wejście przetwornika ADC kontrolera. W **tabeli 1** zostały zebrane połączenia pomiędzy wszystkimi elementami projektu.

Działanie automatu informacyjnego

Automat ma za zadanie przetworzyć odebrany portem USB tekst z zapisem liczb dziesiętnych na odpowiednie komunikaty słowne. Automat jest w stanie prawidłowo czytać liczby w zakresie od 0 do 999999. Zapis liczb powinien być przesłany jako ciąg ASCII składający się ze znaków cyfr „0…9” zakończony znakiem nowej linii LF. Odbiór znaku LF rozpoczyna przetwarzanie. Ciąg cyfr tworzących liczbę kończy pierwszy znak niebędący cyfrą lub liczba cyfr przekraczająca 6 kolejnych znaków.

Drugim sposobem generowania komunikatu słownego jest naciśnięcie niebieskiego przycisku użytkownika na płytce NUCLEO. Program odczytuje poziom napięcia z suwaka potencjometru, konwertuje go na wartość w miliwoltach i wypowiada komunikat, dodając na końcu miano.

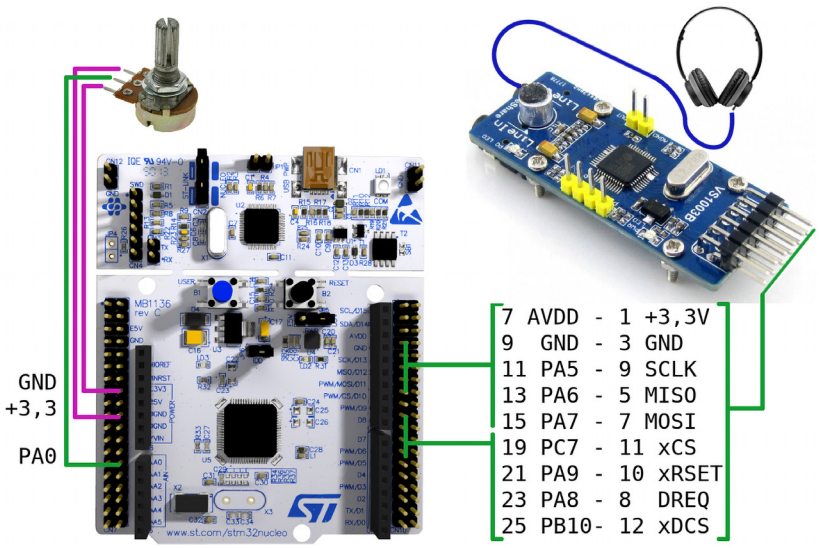
Jak to jest zrobione?

Generowane przez automat komunikaty miały jak najbardziej przypominać mowę naturalną wraz z właściwą odmianą liczebników w języku polskim. Jednocześnie cały program i próbki dźwięków musiały się zmieścić w dostępnej pamięci FLASH kontrolera. Odtwarzanie wartości liczby poprzez wokalizowanie kolejnych pozycji cyfr okazało się mało zrozumiałe w odbiorze i sztuczne. Z kolei tworzenie komunikatu z poszczególnych fonemów odpowiadających kolejnym głoskom czytanej liczby jest zbyt skomplikowane. Ostatecznie okazało się, że najlepsze jest rozwiązanie pośrednie. Nagrano próbki dźwięku liczb, z których można utworzyć większe całości dla liczb z podanego zakresu działania automatu. Są to próbki dla liczb: od 0 do 19, liczebników dziesiętnych w zakresie 10…90, 100…900 oraz słów „tysiąc” i „miliwolty” we wszystkich potrzebnych odmianach. Logika programu musi sobie poradzić z doborem odpowiedniej próbki dźwięku do odtworzenia w zależności od pozycji cyfry w przesłanej liczbie, a także pomijać nieznaczące zera na początku czy w środku odtwarzanej liczby.

Próbki dźwięków liczb można przygotować na dwa sposoby. Albo nagrać samodzielnie z lektorem, tworząc oddzielne małe pliki MP3 dla każdej liczby i liczebników oddzielnie. Albo zrobić to samo, korzystając z darmowych programów typu TTS (Text-To-Speech) w polskiej wersji językowej. Potem każdą próbkę binarną pliku MP3 należy przekonwertować na plik tekstowy z zapisem heksadecymalnym, który można dołączyć do pliku źródłowego oprogramowania kontrolera.

Prezentowany projekt automatu można oczywiście rozwinąć, dostosowując do swoich potrzeb. Odtwarzane pliki komunikatów można przechowywać w dołączanej pamięci zewnętrznej. Można odtwarzać nagrane różne komunikaty, na przykład informacyjne, dodając inne sygnały dźwiękowe. Wykorzystując dodatkowe porty kontrolera, można wyzwalać odtwarzanie komunikatu przez podłączone dodatkowe czujniki itp.

Ryszard Szymaniak
biuro@ars.info.pl



Rysunek 1. Niezbędne połączenia pomiędzy modułem a kontrolerem sterującym na płytce NUCLEO-F091

Tabela 1. Połączenia pomiędzy wszystkimi elementami projektu

NUCLEO		VS1003B	
Złącze	**Sygnał**	**Złącze**	**Sygnał**
CN10-7	AVDD (+3,3 V)	1	+3,3 V
CN10-9	GND	3	GND
CN10-11	PA5	9	SCLK
CN10-13	PA6	5	MISO
CN10-15	PA7	7	MOSI
CN10-19	PC7	11	xCS
CN10-21	PA9	10	xRSET
CN10-23	PA8	8	DREQ
CN10-25	PB10	12	xDCS
NUCLEO		**POTENCJOMETR**	
Złącze	**Sygnał**	**Sygnał**	
CN7-16	+3,3 V	1 skrajny styk	
CN7-20	GND	3 skrajny styk	
CN7-28	PA0 (IN0 ADC)	2 suwak	