STM8S001J3 (6)

Interfejs szeregowy UART

W ramach serii artykułów dotyczących 8-pinowego mikrokontrolera STM8S001J3 zostanie zaprezentowany interfejs komunikacyjny UART. Tak jak to miało miejsce w poprzednich artykułach, tutaj również materiał w pierwszej części udostępnia opis wybranego zasobu mikrokontrolera, a w drugiej części pokazuje krok po kroku przykład jego wykorzystania.

Porty wejścia/wyjścia w trybie GPIO mogą służyć do zrealizowania komunikacji między mikrokontrolerem i układami połączonymi z jego wyprowadzeniami. Sterowanie portami przez programistę z poziomu aplikacji dobrze sprawdza się w sytuacji, gdy komunikacja sprowadza się do wytwarzania i odczytywania pojedynczych stanów logicznych. W praktyce jednak wiele układów współtworzących system elektroniczny komunikuje się przez sekwencje bitów. Rozróżnia się dwa modele takiej komunikacji: transmisję szeregową (bity przesyłane są jeden po drugim przez jedną lub kilka linii sygnałowych) i transmisję równoległą (każdy bit słowa przesyłany jest przez oddzielną linię sygnałową). Realizacja takiej transmisji za pomocą portów w trybie GPIO nie jest optymalnym rozwiązaniem. Po pierwsze, wiąże się z określoną pracą implementacyjną, którą musi wykonać programista (mniej lub bardziej czasochłonną w zależności od poziomu skomplikowania transmisji). Po drugie, w wypadku, gdy aplikacja wymaga od mikrokontrolera częstego komunikowania się z układem lub układami, transmisja danych może okazać się niezbyt efektywna z punktu widzenia czasu wykonania (powód: implementacja ma charakter programowy, nie sprzętowy). Aby rozwiązać przedstawione problemy powszechnym rozwiązaniem stało się integrowanie w mikrokontrolerach kontrolerów odpowiedzialnych za sprzętową realizację komunikacji. Najpopularniejsze interfejsy komunikacyjne to I²C, SPI oraz UART będący przedmiotem tego artykułu.

Interfejs UART

W warstwie fizycznej komunikacja UART realizowana jest przy użyciu dwóch linii o nazwie TXD i RXD. Każdą z nich przewidziano do jednokierunkowego przesyłania danych: TXD jest wyjściem nadajnika, a RXD jest wejściem odbiornika. UART w podstawowej formie jest komunikacją asynchroniczną, a więc linia z sygnałem zegarowym nie jest używana. Jednak w szczególnych przypadkach UART może wykorzystywać linię zegarową i wtedy ma miejsce komunikacja synchroniczna (USART). Ponadto UART jest komunikacją typu punkt-punkt, co oznacza, że przy użyciu jednego łącza mogą komunikować się tylko dwa układy. Schemat połączenia elektrycznego właściwego dla transmisji UART pokazano na **rysunku 1**.

Rozróżnia się dwa tryby komunikacji UART. Pierwszy to full-duplex. Charakteryzuje się on tym, że komunikacja może zachodzić jednocześnie po linii TXD i RXD. Konsekwentnie drugim trybem jest half-duplex, dla którego wysyłanie i odbieranie danych może odbywać się tylko naprzemiennie.

Informacje w interfejsie UART przesyłane są zgodnie ze schematem przedstawionym na **rysunku 2**. Kolejno wyróżnić można: sygnalizujący rozpoczęcie ramki bit startu o stanie niskim, od 5 do 8 bitów danych, opcjonalny bit parzystości służący do sprawdzenia poprawności danych oraz sygnalizujący zakończenie ramki bit (1, 1,5 lub 2) stopu o stanie wysokim.

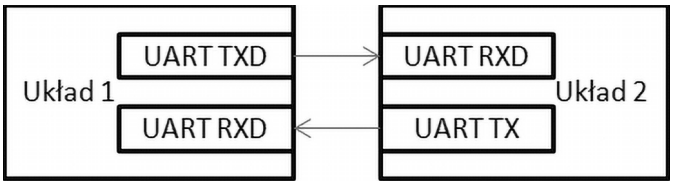
Popularnymi podzespołami używającymi interfejsu UART są moduły Bluetooth, GSM i GPS. Warto pamiętać, że zastosowanie transmisji UART nie ogranicza się tylko do komunikacji mikrokontrolera z komponentami otaczającymi go w systemie. Rozwiązanie to stanowi również podstawę standardów komunikacji między urządzeniami: standardu komputerowego RS-232 i sieciowych standardów przemysłowych RS-422 i RS-485 o topologii magistrali.

Interfejs UART w mikrokontrolerze STM8S001J3

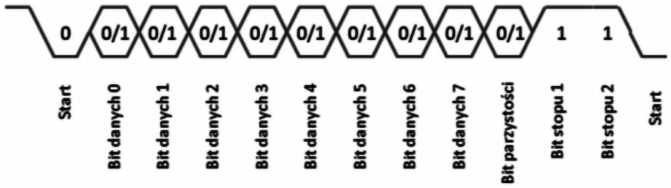
Schemat blokowy interfejs UART w mikrokontrolerze STM8 pokazano na **rysunku 3**. Wyróżnić w nim należy następujące elementy:

- Linie komunikacyjne: nadawczą UART_TX, odbiorczą UART_RX oraz zegarową UART_CK.
- Rejestry dla danych: nadawczy rejestr przesuwny i nadawczy rejestr danych, odbiorczy rejestr przesuwny i odbiorczy rejestr danych.
- Rejestry konfiguracyjne.
- Taktowany przez sygnał fMASTER moduł odpowiedzialny za generowanie podstawy czasu dla wysyłanych i odbieranych danych (innymi słowy dbanie o poprawną prędkość transmisji, tak zwany baudrate).
- Blok przerwań.

Dzięki budowie zgodnej z zaprezentowanym schematem interfejs UART oferuje wszystkie podstawowe funkcjonalności takie jak konfiguracja prędkości transmisji (do 1 Mbit/s), wybór liczby bajtów danych i stopu, wykrywanie błędów komunikacji oraz obsługę przerwań. Ponadto interfejs UART daje możliwość korzystania z różnych trybów transmisji (**rysunek 4**): asynchronicznej full-duplex (w użyciem



Rysunek 1. Schemat połączenia dwóch układów komunikujących się poprzez interfejs UART



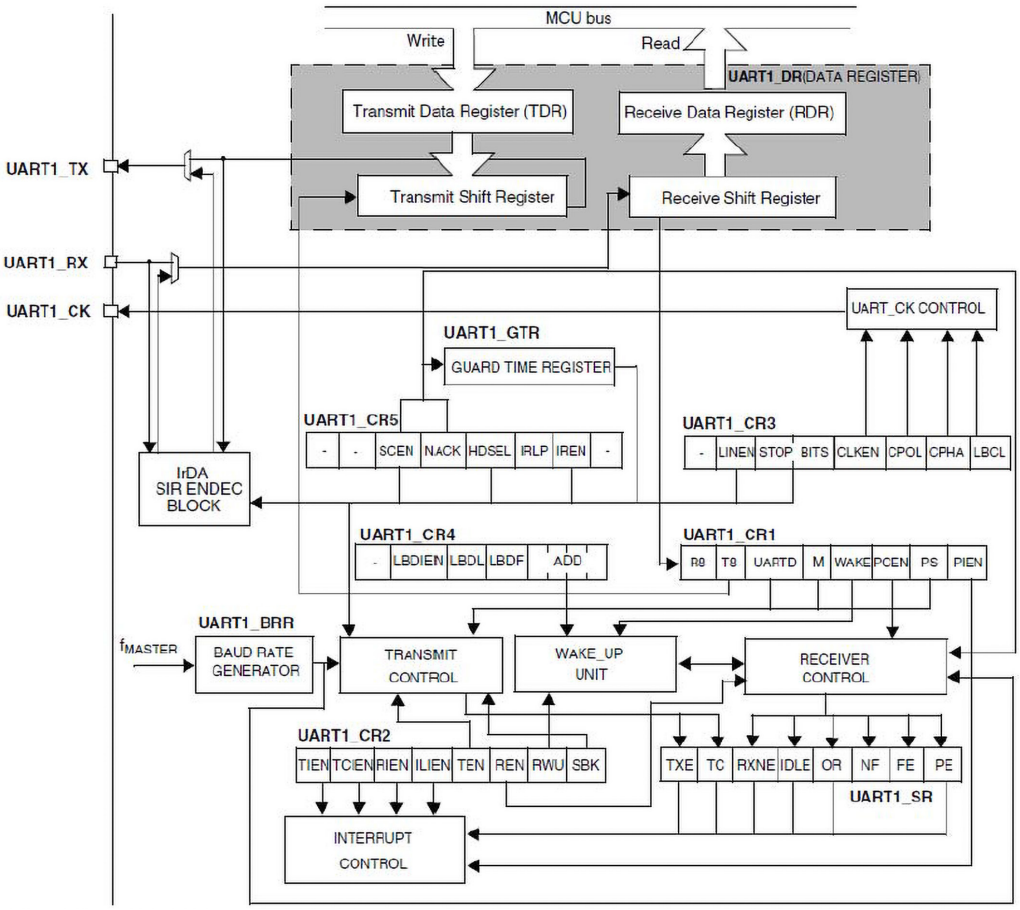
Rysunek 2. Format komunikacji w interfejsie UART

Tabela 1. Lista portów mikrokontrolera STM8S001J3 wraz z dostępnymi dla nich konfiguracjami UART

Numer pinu na obudowie SO8	Nazwa pinu	Funkcja alternatywna	Funkcja alternatywna „remap”
1	PD6/AIN6/UART1_RX	UART1 data receive	
5	PA3/TIM2_CH3/SPI_NSS/UART1_TX		UART1 data transmit
8	PD5/AIN5/UART1_TX	UART1 data transmit	

linii UART_TX i UART_RX), synchronicznej full-duplex (linie UART_TX, UART_RX oraz UART_CK), Single-wire (tylko linia UART_TX do nadawania i odbierania w trybie half-duplex), smartcard, IrDA, Multiprocessor oraz LIN.

W mikrokontrolerze STM8S001J3 dostępny jest jeden interfejs UART o nazwie UART1. Interfejs ten zachował większość wymienionych wcześniej cech. Głównym ograniczeniem jest brak linii UART_CK, co uniemożliwia realizację komunikacji synchronicznej, a więc z sygnałem zegarowym. W **tabeli 1** przedstawiono zestawienie wszystkich portów mikrokontrolera STM8S001J3, które mogą zostać wykorzystane na potrzeby interfejsu UART1. Analizując informacje z tej tabeli można zauważyć, że linia UART1_RX jest dostępna tylko poprzez pin numer 1 mikrokontrolera. Z kolei linia UART1_TX domyślnie jest przypisana do portu PD5 (pin numer 8), jednak użytkownik ma możliwość przeniesienia (remapowania) jej na port PA3 (pin numer 5). W przypadku wykorzystania trybu Single-wire komunikacja realizowana może być za pomocą portu PD5 lub PA3 (do wyboru użytkownika).



Rysunek 3. Schemat blokowy interfejsu UART w mikrokontrolerze STM8

Interfejs UART w STM8CubeMX

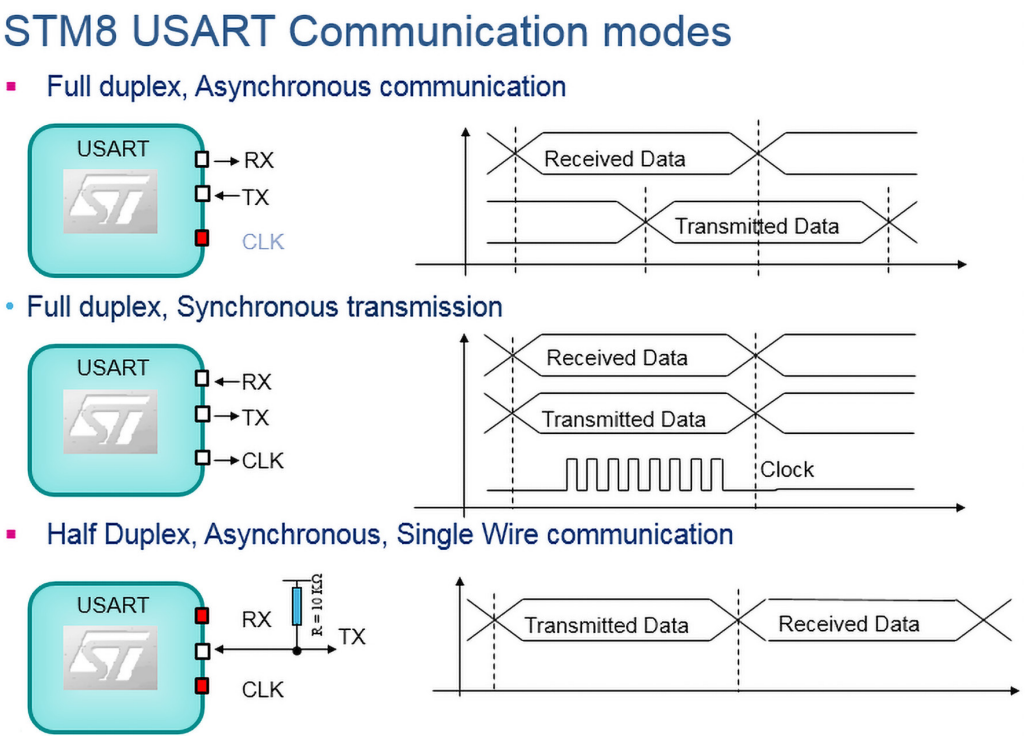
Narzędziem ułatwiającym pracę z portami wejścia/wyjścia mikrokontrolera STM8S001J3 jest program komputerowy STM8CubeMX. Dzięki niemu programista może w prosty sposób (za pomocą graficznego interfejsu użytkownika) sprawdzić jakie są możliwe konfiguracje dla wszystkich portów oraz peryferiów mikrokontrolera. Tak jest również w przypadku interfejsu UART1. Przykładowy scenariusz pokazano na **rysunku 5**.

Funkcje SPL do sterowania interfejsem UART

Aby w prosty sposób skonfigurować interfejs UART układu STM8S001J3, a następnie wysyłać i odbierać dane za jego pomocą, warto w aplikacji użyć bibliotek SPL (*Standard Peripheral Library*) przygotowanych dla mikrokontrolerów z rodziny STM8S. Pliki *stm8s_uart1.h* oraz *stm8s_uart1.c* udostępniają szereg funkcji do tego celu. Zestawienie najważniejszych funkcji zaprezentowano w **tabeli 2**.

Przykładowa aplikacja sterująca interfejsem UART

W celu stworzenia przykładowej aplikacji użyte zostało środowisko programistyczne STVD (ST Visual Develop) oraz kompilator Cosmic CXSTM8. Opis tych narzędzi, jak również instrukcja jak stworzyć za ich pomocą szablon nowego projektu wraz z dodaniem bibliotek



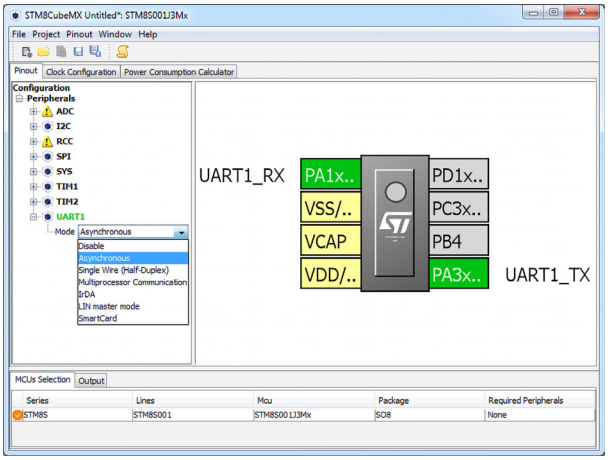
Rysunek 4. Wybrane tryby komunikacji, które wspiera kontroler komunikacyjny UART mikrokontrolera STM8S001J3

Tabela 2. Zestawienie wybranych funkcji z biblioteki dla interfejsu UART

Nazwa funkcji	Opis działania funkcji
UART1_DeInit(void)	Konfiguracja domyślna interfejsu UART1
UART1_Init(uint32_t BaudRate, UART1_WordLength_TypeDef WordLength, UART1_StopBits_TypeDef StopBits, UART1_Parity_TypeDef Parity, UART1_SyncMode_TypeDef SyncMode, UART1_Mode_TypeDef Mode)	Konfiguracja interfejsu UART1 według ustawień użytkownika
UART1_Cmd(FunctionalState NewState);	Włączenie lub wyłączenie interfejsu UART1
UART1_ITConfig(UART1_IT_TypeDef UART1_IT, FunctionalState NewState)	Włączenie lub wyłączenie wybranych przerwań dla interfejsu UART1
UART1_HalfDuplexCmd(FunctionalState NewState)	Włączenie lub wyłączenie trybu half-duplex dla interfejsu UART1
UART1_ReceiveData8(void)	Odczytanie zawartości rejestru odebranych danych interfejsu UART1
UART1_SendData8(uint8_t Data)	Wpisanie danych do rejestru wysyłania danych interfejsu UART1 i wysłanie danych
UART1_ClearFlag(UART1_Flag_TypeDef UART1_FLAG)	Wyzerowanie wybranej flagi interfejsu UART1
UART1_GetITStatus(UART1_IT_TypeDef UART1_IT)	Odczytanie stanu wybranej flagi interfejsu UART1
UART1_ClearITPendingBit(UART1_IT_TypeDef UART1_IT)	Wyzerowanie wybranego bitu związanego z przerwaniami dla interfejsu UART1

SPL dostępne są w artykule numer 3 z tej serii (EP 2/2018). Korzystając ze wspomnianego szablonu projektu należy edytować kod plików *main.c* oraz *stm8s_it.c*, w których umieszczony zostanie kod aplikacji.

Przykładowa aplikacja jest demonstracją dwukierunkowej komunikacji UART. Odbiornik kontrolera UART oczekuje na dane z zewnątrz. Nadejście danych sygnalizuje przerwanie, które inicjuje wykonanie funkcji obsługi przerwania. W funkcji tej każdorazowo odbierany jest jeden bajt danych. Dodatkowo ustawiana jest wartość zmiennej. Zmienna ta monitorowana jest w pętli głównej programu i jeśli jej wartość została zmieniona w funkcji obsługi przerwania, wtedy nadajnik interfejsu UART wysyła informację zwrotną.

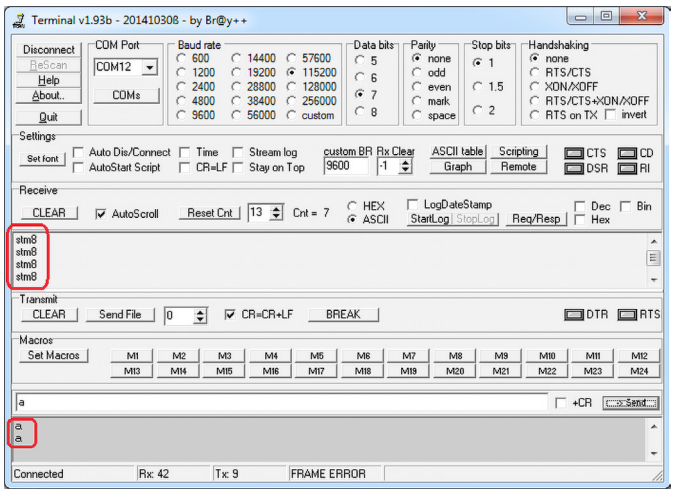


Rysunek 5. Przykładowa konfiguracja mikrokontrolera STM8S001J3 w programie STM8CubeMX

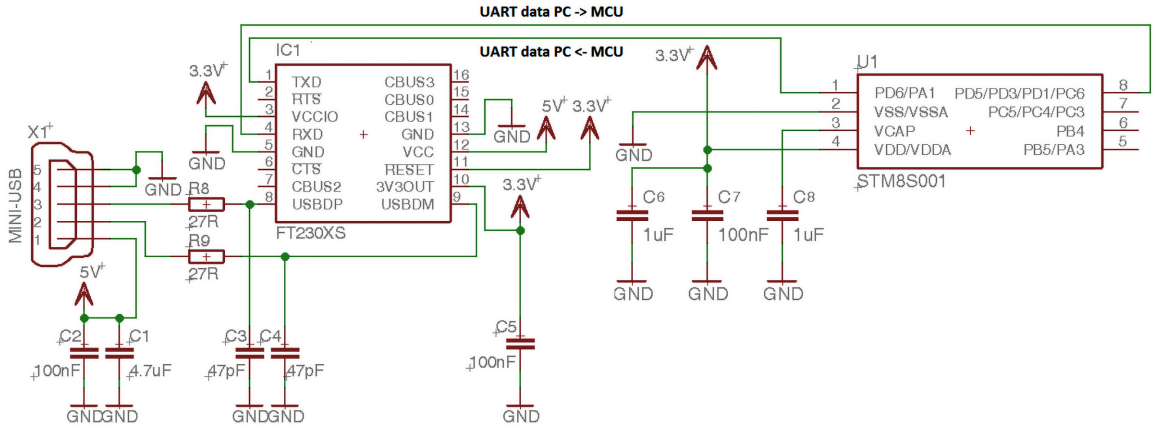
W warstwie sprzętowej interfejs UART mikrokontrolera STM8S001J3 połączony jest z układem FTDI będącym konwerterem UART-USB, widzianym po stronie komputera jako wirtualny port COM. Odpowiedni schemat pokazano na **rysunku 6**.

W celu zaimplementowania opisanej aplikacji wykonane zostaną następujące kroki w pliku main.c:

- Utworzenie tablicy *TX_data*, która przechowuje dane do wysłania interfejsem UART.
- Utworzenie zmiennej *i*, która pomaga przy indeksowaniu bajtów, które zostaną wysłane interfejsem UART.



Rysunek 7. Terminal komputerowy pokazujący dane wysłane do i odebrane z mikrokontrolera STM8S001J3



Rysunek 6. Schemat połączeń między mikrokontrolerem STM8S001J3, układem FTDI oraz komputerem PC

- Utworzenie zmiennej *interrupt*, która przechowuje informację o przerwaniu.
- Utworzenie zmiennej *RX_data*, która przechowuje dane odebrane interfejsem UART
- Utworzenie funkcji opóźniającej *delay()*.
- Wywołanie funkcji opóźniającej *delay()*, co przeciwdziała przez kilka sekund ewentualnemu późniejszemu wyłączeniu interfejsu programowania i debugowania SWIM będącego efektem rekonfiguracji portów.
- Wykonanie kodu konfiguracji portów wejścia/wyjścia, które nie są połączone z wyprowadzeniami mikrokontrolera (kod wzięty z noty aplikacyjnej AN5047: *Getting started with the STM8S001J3 microcontroller*).
- Wywołanie funkcji *UART1_Deinit()* w celu wykonania konfiguracji domyślnej interfejsu UART.
- Wywołanie funkcji *UART1_Init()* w celu wykonania konfiguracji interfejsu UART zgodnej z parametrami wybranymi przez użytkownika.
- Wywołanie funkcji *UART1_ITConfig ()* włączającej wybrane przerwania dla interfejsu UART.
- Wywołanie funkcji *enableInterrupts()* włączającej działanie przerwań.
- Wewnątrz nieskończonej pętli *while()*:
  - Wykonanie instrukcji warunkowej *if()* sprawdzającej wartość zmiennej *interrupt*.
  - Jeśli zmienna *interrupt* ma wartość 1, co oznacza sygnalizację przerwania i nowych odebranych danych, wywoływana jest w pętli *for()* funkcja *UART1_SendData8()* wysyłająca dane przez interfejs UART. Zakończenie wysyłania każdego bajtu sprawdzane jest funkcją *UART1_GetFlagStatus()* w pętli *while()*. Na koniec zerowana jest zmienna *interrupt*.

Kod zgodny z zaprezentowanym opisem pokazano w **listingu 1**. Obsługa przerwań zaimplementowana jest w pliku *stm8s_it.c*. W pliku tym należy:

- Zasygnalizować przez dyrektywę *extern* istnienie zmiennych *interrupt* oraz *RX_data*).
- W funkcji obsługi przerwania od interfejsu UART wykonać funkcję *UART1_ReceiveData8()* odbierającą dane i wpisanie zwracanej przez nią wartości do zmiennej *RX_data* (**listing 2**).

Po stronie komputera do komunikacji wystarczy dowolny program będący terminalem komunikacyjnym do obsługi portu COM. Test komunikacji z mikrokontrolerem można wykonać poprzez wybranie portu COM odpowiadającego układowi FTDI, ustawienie odpowiedniej prędkości transmisji i wysłanie pojedynczego znaku.

Listing 1. Stworzony w oparciu o funkcje SPL kod pliku main.c

```
#include "stm8s.h"

unsigned char TX_data[] = {'s','t','m','8',0xd, 0xa};
unsigned char i;
unsigned char interrupt;
unsigned char RX_data = 0;

void delay(unsigned long int how_long);

main()
{
    //delay to avoid irreversible blocking of SWIM
    delay(100000);
    //configuration of unused pins
    GPIOA->DDR |= GPIO_PIN_2;
    GPIOB->DDR |= GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_6 | GPIO_PIN_7;
    GPIOC->DDR |= GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_7;
    GPIOD->DDR |= GPIO_PIN_0 | GPIO_PIN_2 | GPIO_PIN_4 | GPIO_PIN_7;
    GPIOE->DDR |= GPIO_PIN_5;
    GPIOF->DDR |= GPIO_PIN_4;
    //UART configuration
    UART1_DeInit();
    UART1_Init(115200, UART1_WORDLENGTH_8D, UART1_STOPBITS_1, UART1_PARITY_NO,
               UART1_SYNCMODE_CLOCK_DISABLE, UART1_MODE_TXRX_ENABLE);
    UART1_ITConfig(UART1_IT_RXNE, ENABLE);
    enableInterrupts();
    while (1)
    {
        if(interrupt == 1)
        {
            for (i = 0; i <= 5; i++)
            {
                UART1_SendData8(TX_data[i]);
                while (UART1_GetFlagStatus(UART1_FLAG_TXE) == RESET);
            }
            interrupt = 0;
        }
    }
}

void delay(unsigned long int how_long)
{
    unsigned long int i;

    for (i = 0; i< how_long; i++)
    {
    }
}
```

Listing 2. Obsługa przerwania

```
/* Private variables */
extern unsigned char interrupt;
extern unsigned char RX_data;
…

/**
  * @brief  UART1 RX Interrupt routine
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(UART1_RX_IRQHandler, 18)
{
    RX_data = UART1_ReceiveData8();
    interrupt = 1;
}
```

W odpowiedzi należy spodziewać się informacji zwrotnej. Pokazano to na **rysunku 7**.

Podsumowanie

W artykule przekazano podstawowe informacje o interfejsie UART mikrokontrolera STM8S001J3 wraz z opisem przykładowej aplikacji. Osoby chcące dowiedzieć się bardziej szczegółowych informacji powinny sięgnąć do dokumentacji technicznej producenta. Parametry i charakterystyka interfejsu UART dostępne są w nocie katalogowej mikrokontrolera (*datasheet*). Z kolei schematy oraz opis wszystkich funkcjonalności i rejestrów znajduje się w podręczniku użytkownika mikrokontrolera (*user manual* RM0016). Dodatkowo szereg aplikacji testowych dostępny jest w dedykowanym podkatalogu bibliotek SPL: …\STM8S_StdPeriph_Lib\Project\STM8S_StdPeriph_Examples\UART1.

Szymon Panecki
szymon.panecki@st.com

REKLAMA