STM8S001J3 (5)

Porty wejścia/wyjścia mikrokontrolera oraz przerwania zewnętrzne

W ramach serii artykułów dotyczących 8-pinowego mikrokontrolera STM8S001J3 przyszedł czas na zapoznanie się z podstawowym peryferium, jakim są porty wejścia/wyjścia oraz towarzyszącym im blokiem EXTI, który odpowiada za powiązane z portami przerwania zewnętrzne. W artykule przedstawiono opis tych dwóch zasobów mikrokontrolera, a ich działanie zilustrowano przykładowymi aplikacjami.

W każdym systemie elektronicznym mikrokontroler poprzez swoje wyprowadzenia, a dalej ścieżki na płytce PCB dołączony jest do pewnej liczby podzespołów, z którymi musi się komunikować. Komunikację należy tu rozumieć jako umiejętność wysyłania i odbierania informacji w postaci sygnału elektrycznego (napięciowego). Zadanie to realizują porty wejścia/wyjścia.

Porty wejścia/wyjścia w mikrokontrolerze STM8S001J3

Porty mikrokontrolera mogą pracować w dwóch trybach. Pierwszy z nich to GPIO (*General Purpose Input Output*). W trybie tym programista ma bezpośrednią kontrolę nad portami i może sterować nimi z poziomu aplikacji. Porty w trybie GPIO wykorzystywane są przede wszystkim do operacji wejścia wyjścia, np. odczytywania stanów przycisków lub sterowania stanem (włącz/wyłącz) przekaźników, kluczy tranzystorowych, LEDów itp.

Drugi tryb pracy portów wejścia/wyjścia nazywany jest trybem alternatywnym. W trybie ty porty są połączone z peryferiami mikrokontrolera, aby umożliwić ich komunikowanie się ze światem zewnętrznym. Przykładowo porty mogą być wykorzystane przez przetwornik A/C do odczytywania wartości napięcia z wyprowadzenia mikrokontrolera, przez timer do generowania sygnału PWM poza układ lub przez interfejsy komunikacyjne do realizacji transmisji danych między mikrokontrolerem i innymi układami.

Port wejścia/wyjścia w mikrokontrolerze STM8 składa się z następujących bloków:

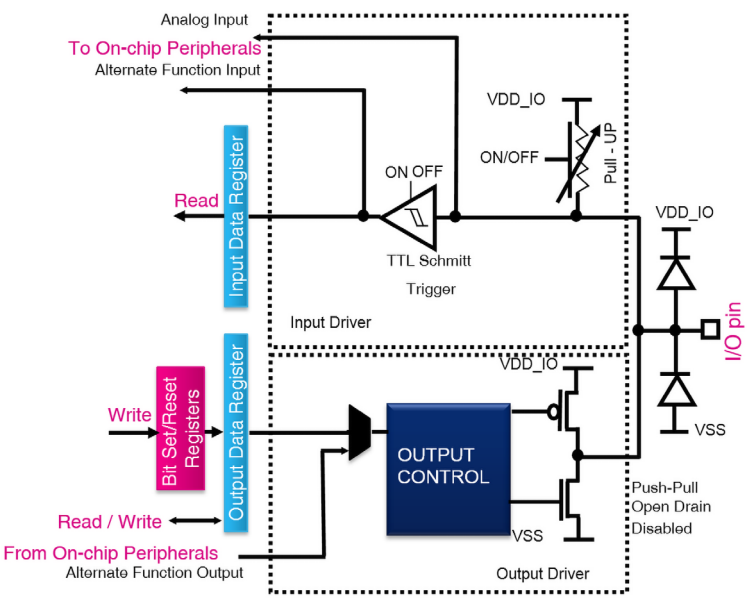
- Rejestrów (*Registers*) służących do konfiguracji tryby pracy portu, do przechowywania odczytanej z pinu wartości logicznej (w przypadku pracy w trybie wejściowym) i do przechowywania wartości, która na zostać wystawiona na pinie (w przypadku pracy w trybie wyjściowym).
- Kontrolera wejścia (*Input Driver*), który zbudowany jest z przerzutnika Schmitta i rezystora podciągającego do napięcia zasilania.
- Kontrolera wyjścia (*Output Driver*), który zbudowany jest z multipleksera sterowania (rejestry portu lub peryferia) oraz sterownika sygnału wyjściowego kontrolującego tranzystory.
- Dwóch diod zabezpieczających: jedna między pinem i dodatnim potencjałem napięcia zasilania, druga między pinem i ujemnym potencjałem napięcia zasilania.

Schemat blokowy portu wejścia wyjścia mikrokontrolera STM8 pokazano na **rysunku 1**. Poprzez aktywowanie i konfigurowanie poszczególnych elementów portu można uzyskać różne konfiguracje pracy. Łącznie jest ich dziewięć. Konfiguracje wejściowe: *floating without interrupt*, *floating with interrupt*, *pull-up without interrupt* oraz *pull-up with interrupt*. Konfiguracje wyjściowe: *Open drain*, *Open drain fast*, *Push-pull*, *Push-pull fast* oraz *True open drain*. W praktyce jednak budowa portów wejścia/wyjścia nie zawsze jest taka sama, dlatego istotne jest sprawdzenie w dokumentacji, jakie tryby są dostępne dla danego portu mikrokontrolera.

Jak już wspomniano wcześniej, wybrane porty wejścia/wyjścia mogą pracować w trybie alternatywnym, co oznacza wykorzystanie portu przez inne peryferia mikrokontrolera. W **tabeli 1** przedstawiono zestawienie wszystkich portów mikrokontrolera STM8S001J3 wraz z dostępnymi dla nich konfiguracjami.

Struktura krzemowa mikrokontrolera dysponuje również portami wejścia/wyjścia, które nie są połączone z pinami obudowy. Są to: PA2, PB0, PB1, PB2, PB3, PB6, PB7, PC1, PC2, PC7, PD0, PD2, PD4, PD7, PE5 oraz PF4. Powinny one zostać skonfigurowane przez użytkownika w trybie pracy *output push-pull*, a ich wyjściowy poziom logiczny powinien zostać ustawiony jako niski. Ma to na celu obniżenie poboru prądu mikrokontrolera i zwiększenie odporności EMC układu.

Podczas konfigurowania portów wejścia/wyjścia należy zwrócić szczególną uwagę na port PD1, który jest współdzielony z interfejsem programowania i debugowania SWIM. Z uwagi na brak pinu NRST aplikacja mikrokontrolera STM8S001J3 musi sama zagwarantować, że interfejs SWIM jest dostępny w momencie, gdy sprzętowy programator debuger ST-Link nawiązuje połączone z mikrokontrolerem. Po rekonfiguracji portu PD1 interfejs ten nie jest dłużej dostępny, zatem aby mieć pewność, że ST-Link połączy się z mikrokontrolerem zaleca się, aby pierwszym etapem działania aplikacji była kilkusekundowa pętla opóźniająca, która da czas potrzebny użytkownikowi na zainicjowanie trybu debugowania. Po upłynięciu tego czasu aplikacja może realizować właściwe jej zadania, w tym przekonfigurować domyślny tryb portu PD1.



Rysunek 1. Budowa portu wejścia/wyjścia w mikrokontrolerze STM8

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

Numer pinu na obudowie SO8	Nazwa pinu	Domyślna funkcja	Funkcja alternatywna	Funkcja alternatywna „remap”
1	PD6/ AIN6/ UART1 _RX	Port PD6	Analog input 6/UART1 data receive	–
	PA1/ OSCIN	Port PA1	External clock input (HSE)	–
5	PA3/ TIM2_ CH3 SPI_ NSS/ UART1_TX	Port PA3	Timer 2 channel 3	SPI master/slave select / UART1 data transmit
	PB5/ I2C_ SDA/ TIM1_ BKIN	Port PB5	I2C data	Timer 1 break input
6	PB4/ I2C_ SCL/ ADC_ETR	Port PB4	I2C clock	ADC external trigger
7	PC3/ TIM1_CH3/ TLI/ TIM1_ CH1N	Port PC3	Timer 1 channel 3	Top Level Interrupt/ Timer 1 channel 1 inverted
	PC4/ CLK_CCO/ TIM1_ CH4/ AIN2/ TIM1_ CH2N	Port PC4	Configurable clock output/ Timer 1 channel 4	Analog input 2/ Timer 1 channel 2 inverted
	PC5/ SPI_SCK/ TIM2_ CH1	Port PC5	SPI clock	Timer 2 channel 1
8	PC6/ SPI_MOSI/ TIM1_ CH1	Port PC6	SPI master/slave in	Timer 1 channel 1
	PD1/ SWIM	Port PD1	SWIM debug interface	–
	PD3/ AIN4/ TIM2_ CH2/ ADC_ ETR	Port PD3	Analog input 4/ Timer 2 channel 2/ ADC external trigger	–
	PD5/ AIN5/ UART1 _TX	Port PD5	Analog input 5/ UART data transmit	–

Porty wejścia/wyjścia w STM8CubeMX

Narzędziem ułatwiającym pracę z portami mikrokontrolera STM8S001J3 jest program komputerowy STM8CubeMX. Dzięki niemu programista może w prosty sposób (za pomocą graficznego interfejsu użytkownika) sprawdzić, jakie są możliwe konfiguracje (tryby wejścia, wyjścia i alternatywne) dla wszystkich portów mikrokontrolera. Przykładowy scenariusz pokazano na **rysunku 2**.

Funkcje SPL do sterowania portami wejścia/wyjścia

Aby w prosty sposób skonfigurować porty wejścia/wyjścia układu STM8S001J3, a następnie nimi sterować, warto w aplikacji użyć bibliotek SPL (*Standard Peripheral Library*) przygotowanych dla mikrokontrolerów STM8S. Pliki *stm8s_gpio.h* oraz *stm8s_gpio.c* udostępniają szereg funkcji do tego celu. Ich zestawienie zaprezentowano w **tabeli 2**.

Przykładowa aplikacja sterująca portami wejścia/wyjścia

W celu wykonania przykładowej aplikacji zostało użyte środowisko programistyczne STVD (ST Visual Develop) oraz kompilator Cosmic CXSTM8. Opis tych narzędzi, jak również instrukcja jak wykonać za ich pomocą szablon nowego projektu wraz z dodaniem bibliotek SPL są dostępne w trzecim artykule z tej serii. Korzystając ze wspomnianego szablonu projektu należy edytować kod pliku *main.c*, w którym umieszczony zostanie cały kod aplikacji.

Przykładowa aplikacja demonstruje podstawowy sposób wykorzystania portów wejścia/wyjścia, a więc odczytywanie wartości logicznej z wejścia portu oraz wystawianie wartości logicznej na wyjściu portu. Na potrzeby generowania sygnału wejściowego użyty został przycisk (dołączony do portu PA3), natomiast w celu

Tabela 2. Zestawienie wybranych funkcji

Nazwa funkcji	Opis działania funkcji
GPIO_DeInit(GPIO_TypeDef* GPIOx)	Konfiguracja domyślna wybranego portu
GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_Pin_TypeDef GPIO_Pin, GPIO_Mode_TypeDef GPIO_Mode)	Konfiguracja wybranego portu według ustawień użytkownika
GPIO_Write(GPIO_TypeDef* GPIOx, uint8_t PortVal)	Ustawienie wartości logicznej na wyjściu portów według wskazania użytkownika
GPIO_WriteHigh(GPIO_TypeDef* GPIOx, GPIO_Pin_TypeDef PortPins)	Ustawienie wartości logicznej wysokiej na wyjściu wybranych przez użytkownika portach
GPIO_WriteLow(GPIO_TypeDef* GPIOx, GPIO_Pin_TypeDef PortPins)	Ustawienie wartości logicznej niskiej na wyjściu wybranych przez użytkownika portach
PIO_WriteReverse(GPIO_TypeDef* GPIOx, GPIO_Pin_TypeDef PortPins)	Ustawienie wartości logicznej przeciwnej do aktualnej na wyjściu wybranych przez użytkownika portach
GPIO_ReadInputData(GPIO_TypeDef* GPIOx)	Odczytanie wartości logicznej na wejściu portów
GPIO_ReadOutputData(GPIO_TypeDef* GPIOx)	Odczytanie wartości logicznej na wyjściu portów
GPIO_ReadInputPin(GPIO_TypeDef* GPIOx, GPIO_Pin_TypeDef GPIO_Pin)	Odczytanie wartości logicznej na wybranym przez użytkownika porcie
GPIO_ExternalPullUpConfig(GPIO_TypeDef* GPIOx, GPIO_Pin_TypeDef GPIO_Pin, FunctionalState NewState)	Włączenie lub wyłączenie rezystora podciągającego do dodatniego napięcia zasilania na wybranym porcie

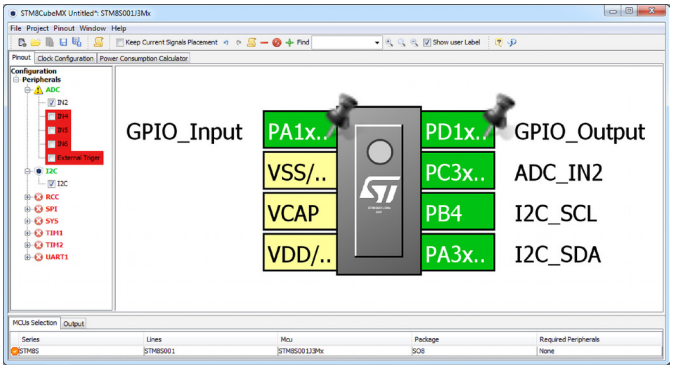
wizualizacji sygnału wyjściowego wykorzystano diodę LED (dołączony do portu PC3). Program działa według schematu: jeśli przycisk nie jest wciśnięty, dioda LED miga, a gdy przycisk jest wciśnięty, dioda LED nie miga. W tym celu wykonane zostaną następujące kroki:

- Zostanie wykonana funkcja opóźniająca *delay()*.
- Na początku aplikacji wywołana zostanie funkcja opóźniająca, co przeciwdziała przez kilka sekund ewentualnemu późniejszemu wyłączeniu interfejsu programowania i debugowania SWIM będącemu efektem rekonfiguracji portów.
- Wykonany zostanie kod 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 *GPIO_Init()* skonfiguruje port PC3 jako wejście i PA3 jako wyjście.
- W nieskończonej pętli:
  - Wywołanie funkcji *GPIO_ReadInputPin()* pozwoli określić poziom logiczny na wejściu portu PC3. W zależności od wyniku zwróconego przez tą funkcję Instrukcja warunkowa *if* zdecyduje o dalszym kroku.
  - Jeśli na wejściu portu odczytany zostanie niski stan logiczny, wywołane zostaną kolejno funkcje *GPIO_WriteHigh()*, *delay()*, *GPIO_WriteLow()* i ponownie *delay()*, co spowoduje mignięcie diody LED.
  - Jeśli na wejściu portu odczytany zostanie wysoki stan logiczny, nie zostanie wykonane żadne zadanie.

Kod zgodny z zaprezentowanym opisem pokazano w **listingu 1**.

Moduł EXTI w mikrokontrolerze STM8S001J3

Obsługę portów wejścia/wyjścia można usprawnić poprzez użycie przerwań zewnętrznych (EXTI – External Interrupts). Blok EXTI jest częścią kontrolera przerwań o nazwie ITC (*Interrupt controller*). Kontroler ten udostępnia do 32 przerwań sprzętowych z możliwością konfiguracji 4 poziomów priorytetów oraz 3 niemaskowalne przerwania: RESET, TRAP oraz TLI (*Top Level Interrupt*). W odniesieniu do portów mikrokontrolera STM8S001J3 możliwe jest wykorzystanie przerwań od grupy portów PA (wyprowadzenia PA1, PA3), PB (PB4, PB5), PC (PC3 współdzielony z TLI, PC4, PC5, PC6) oraz PD (PD1, PD3, PD5, PD6). Aby port mógł generować przerwanie, musi być skonfigurowany do pracy w trybie wejściowym z włączonym przerwaniem. Każda grupa portów ma wybieralny typ zbocza sygnału, które aktywuje przerwanie (opadające, rosnące lub oba).



Rysunek 2. Przykładowa konfiguracja portów wejścia/wyjścia mikrokontrolera STM8S001J3 w programie STM8CubeMX

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

```
#include "stm8s.h"

void delay(unsigned long int how_long);

main()
{
    //delay to avoid 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;
    //configuration of used pins: PC3 as input (button), PA3 as output (LED)
    GPIO_Init(GPIOC, GPIO_PIN_3, GPIO_MODE_IN_FL_NO_IT);
    GPIO_Init(GPIOA, GPIO_PIN_3, GPIO_MODE_OUT_PP_LOW_FAST);

    while (1)
    {
        if(GPIO_ReadInputPin(GPIOC, GPIO_PIN_3) == FALSE)
        {
            GPIO_WriteHigh(GPIOA, GPIO_PIN_3);
            delay(2000);
            GPIO_WriteLow(GPIOA, GPIO_PIN_3);
            delay(2000);
        }
    }
}

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

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

Listing 2. Przykładowa aplikacja sterująca EXTI

```
#include "stm8s.h"

unsigned char interrupt = 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;
    //configuration of used pins: PC3 as input (button), PA3 as output (LED)
    GPIO_Init(GPIOC, GPIO_PIN_3, GPIO_MODE_IN_FL_IT);
    GPIO_Init(GPIOA, GPIO_PIN_3, GPIO_MODE_OUT_PP_LOW_FAST);

    EXTI_SetExtIntSensitivity(EXTI_PORT_GPIOC, EXTI_SENSITIVITY_RISE_ONLY);
    enableInterrupts();
    while (1)
    {
        if(interrupt == 1)
        {
            GPIO_WriteHigh(GPIOA, GPIO_PIN_3);
            delay(2000);
            GPIO_WriteLow(GPIOA, GPIO_PIN_3);
            delay(2000);
            interrupt = 0;
        }
    }
}

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

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

Tabela 3. Zestawienie wybranych funkcji

Nazwa funkcji	Opis działania funkcji
EXTI_DeInit(void)	Konfiguracja modułu EXTI według ustawień użytkownika
EXTI_SetExtIntSensitivity(EXTI_Port_TypeDef Port, EXTI_Sensitivity_TypeDef SensitivityValue)	Wybranie zbocza sygnału, które aktywuje przerwanie dla portu wejścia/wyjścia
EXTI_SetTLISensitivity(EXTI_TLISensitivity_TypeDef SensitivityValue)	Wybranie zbocza sygnału, które aktywuje przerwanie TLI
EXTI_GetExtIntSensitivity(EXTI_Port_TypeDef Port)	Odczytanie aktualnie wybranego zbocza sygnału, które aktywuje przerwanie dla portu wejścia/wyjścia
EXTI_GetTLISensitivity(void)	Odczytanie aktualnie wybranego zbocza sygnału, które aktywuje przerwanie TLI

Funkcje SPL do sterowania modułem EXTI

Biblioteki SPL udostępniają funkcje dla przerwań zewnętrznych, których użycie ułatwia pisanie kodu aplikacji. Funkcje te znajdują się w plikach *stm8s_exti.h* oraz *stm8s_exti.c*. Ich zestawienie zaprezentowano w **tabeli 3**.

Listing 3. Procedura obsługi przerwania

```
/* Private variables ---------------------------------------------------------*/
extern unsigned char interrupt;
…
/**
  * @brief  External Interrupt PORTC Interrupt routine
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(EXTI_PORTC_IRQHandler, 5)
{
    interrupt = 1;
}
```

Przykładowa aplikacja sterująca modułem EXTI

Łatwo zauważyć, że przygotowana wcześniej aplikacja sterującą portami wejścia/wyjścia nie jest zbyt efektywna ze względu na programową obsługę przycisku, z czego wynika konieczność nieustanego (czasochłonnego) sprawdzania przez mikrokontroler stanu logicznego na wejściu portu. Zadanie to można zoptymalizować przez dodanie obsługi przerwania (EXTI), dzięki czemu zmiana poziomu logicznego na wejściu portu (naciśnięcie przycisku) będzie sygnalizowane sprzętowo. Modyfikacja pliku main.c polega na:

- Dodaniu zmiennej *interrupt*, która przechowuje informację o przerwaniu.
- Zmianie argumentu funkcji *GPIO_Init()* z *GPIO_MODE_IN_FL_NO_IT* na *GPIO_MODE_IN_FL_IT*.
- Wywołaniu funkcji *EXTI_SetExtIntSensitivity()* ustawiającej aktywne zbocze dla przerwania.
- Wywołanie funkcji *enableInterrupts()* włączającej przerwania.
- Wewnątrz instrukcji warunkowej *if*:
  - Zamianie wywołania funkcji *GPIO_ReadInputPin()* na zmienną *interrupt*.
  - Wyzerowanie zmiennej *interrupt*.

Kod zgodny z zaprezentowanym opisem pokazano w **listingu 2**. Obsługa przerwań zaimplementowana jest w pliku *stm8s_it.c*. W pliku tym należy:

- Zasygnalizować przez dyrektywę *extern* istnienie zmiennej *interrupt*.
- Przypisać wartość 1 zmiennej *interrupt* w funkcji obsługi przerwania od grupy portów PC.

Kod pokazano w **listingu 3**.

Podsumowanie

W artykule przekazano podstawowe informacje o portach wejścia/wyjścia i przerwaniach zewnętrznych mikrokontrolera STM8S001J3 wraz z opisem przykładowych aplikacji. Osoby chcące dowiedzieć się bardziej szczegółowych informacji powinny sięgnąć do dokumentacji technicznej producenta. Parametry i charakterystyka obu zasobów mikrokontrolera 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). Informacje uzupełniające dostępne są w nocie aplikacyjnej AN5047: *Getting started with the STM8S001J3 microcontroller*. Dodatkowo aplikacje testowe dostępne są w podkatalogu bibliotek SPL: …\STM8S_StdPeriph_Lib\Project\STM8S_StdPeriph_Examples\GPIO oraz …\STM8S_StdPeriph_Lib\Project\STM8S_StdPeriph_Examples\EXTI.

Szymon Panecki
szymon.panecki@st.com