
NobleProg

Mikrokontrolery STM32 – wprowadzenie – porty, timery, przerwania

Grzegorz Mazur

Plan warsztatów

- Podstawowe bloki mikrokontrolerów STM32
- Włączanie modułów peryferyjnych - RCC
- Porty GPIO
- Timer systemowy SysTick
- Reakcja na stan przycisków i innych elementów stykowych
- Timery wielofunkcyjne

Podstawowe bloki STM32

- Sterownik systemu SYS
- Sterownik przerwań NVIC
- Sterowanie taktowaniem RCC
- Porty wejścia-wyjścia GPIO
- Timer rdzenia SysTick

RCC – włączanie modułów

- We współczesnych uC, w celu ograniczenia poboru mocy, bloki funkcjonalne są włączane przez oprogramowanie wtedy, gdy są potrzebne
- Blok wyłączony nie może być konfigurowany ani używany
- W STM32 zarządzaniem stanem bloków zajmuje się moduł RCC

RCC – włączanie modułów

- Rejestry o nazwach xxxxENR zawierają bity sterujące włączeniem modułów peryferyjnych
- W celu włączenia modułu należy ustawić odpowiadający mu bit w stan 1
  - Odpowiednie maski bitowe mają zdefiniowane nazwy symboliczne
- Przykład – włączenie GPIO

```
RCC->AHB2ENR |= RCC_AHB2ENR_GPIOAEN | RCC_AHB2ENR_GPIOBEN
```

Porty GPIO

- Podstawowa funkcjonalność wejścia-wyjścia
  - Programowe sprawdzanie stanu wejścia
  - Programowe ustawianie stanu wyjścia
- Programowa konfiguracja portów
  - Określenie kierunku linii
  - Określenie charakterystyki elektrycznej
    - Rezystory ustalające stan wejść przy braku sterowania zewnętrznego
    - Wydajność prądowa i stromość zboczy wyjść
    - Jednostronne sterowanie wyjścia (tylko w stanie niskim) – OD (open drain – otwarty dren

GPIO w STM32

- Porty mają szerokość 16 bitów
- Liczba portów i dostępnych linii zależy od typu i obudowy uC (od 6 do ok. 200 linii)
- Porty są oznaczone literami – GPIOA, GPIOB, GPIOC, …
- Linie portów – symbol i numer – PA0, PA1, … PA15, PB0..PB15, ...

Model programowy GPIO STM32 - rejestry

- AFR – wybór funkcji (połączenia z blokiem peryferyjnym)
- MODER – określenie funkcji (kierunku) linii
- PUPDR – sterowanie rezystorami ustalającymi stan
- OTYPER, OSPEEDR – charakterystyka elektryczna wyjść
- ODR – odczyt (ew. ustawianie) stanu wyjść
- IDR – odczyt stanu wejść
- BSRR, BRR – ustawianie stanu wyjść

Programowanie GPIO

- Włączyć port!
- Jeśli linia jest używania w charakterze innym niż GPIO – wybrać funkcję w AFR
- Ustawić stan początkowy dla wyjścia - BSRR
- Ustawić charakterystykę elektryczną dla wyjścia – OTYPER, OSPEEDR
- Ustawić ustalanie stanu wejścia (jeśli potrzebne) – PUPDR
- Skonfigurować linię – rejestr MODER
- Jeśli żadna linia portu nie jest używana jako GPIO, dany port można po konfiguracji wyłączyć

Rejestr MODER

- 2-bitowe pola określają tryb pracy poszczególnych linii portu
- Tryby:
  - 00 – wejście GPIO
  - 01 – wyjście GPIO
  - 10 – linia połączona z peryferialem
  - 11 – wejście/wyjście analogowe
- Linie nieużywane (niepodłączone) powinny być ustawione jako analogowe)
  - W nowszych seriach STM32 – L, H, G i C domyślnym trybem jest tryb analogowy
  - W starszych seriach (F) – wejście cyfrowe

Zmiana stanu wyjść

- Do programowego zmieniania stanu wyjść używamy rejestrów BSRR i BRR
- BRR – 16-bitowy rejestr zerowania wyjść
  - Zapis 1 na pozycję n powoduje wyzerowanie wyjścia n, zapis 0 nie zmienia stanu
- BSRR – rejestr 32-bitowy ustawiania/zerowania wyjść
  - Bity 31..16 odpowiadają rejestrowi BRR – zapis 1 na pozycję (n+16) powoduje wyzerowanie linii n
  - Bity 15..0 – zapis 1 na pozycję n powoduje ustawienie stanu 1 wyjścia n, zapis 0 nie zmienia stanu
  - Równoczesny zapis 1 na pozycjach n i (n+16) powoduje ustawienie 1

Zmiana stanu wyjść – cd.

- Równoczesny zapis jedynek odpowiadających tej samej linii portu do górnej i dolnej połowy BSRR powoduje ustawienie wyjścia w stan 1
  - „SET” ma wyższy priorytet niż „RESET”
- Odwrócenie stanu linii wyjściowej:

```
GPIOx->BSRR = MASK << 16 | (MASK & ~GPIOx->ODR);
```

SysTick

- Timer rdzenia ARM, służy do zgłaszania przerwań ze stałą częstotliwością
- Jeśli nie ma innych uwarunkowań, zwykle programuje się go na częstotliwość 1 kHz
- Programowanie – zapis 3 rejestrów lub użycie pseudofunkcji SysTick_Config()
  - Funkcja ustawia najniższy możliwy priorytet przerwania! - nie zawsze jest to właściwe
- Obsługa przerwania – procedura SysTick_Handler()

Przykład – zaświecanie i gaszenie LED

Sprawdzanie stanu przycisku

- W celu zignorowania wpływu drgań styków, stan wejść połączonych z przyciskami i czujnikami zestykowymi należy testować w przerwaniu timera, z okresem dłuższym od maksymalnego okresu drgań

Timery wielofunkcyjne

- Mikrokontrolery STM32 są wyposażone w kilka..kilkanaście timerów/liczników wielofunkcyjnych TIMx
  - Timer o określonym numerze x jest identyczny we wszystkich modelach STM32
- Poszczególne timery mają jednakowy blok licznika/bazy czasu; różnią się dostępnością i liczbą kanałów chwytania/porównania i dodatkowymi możliwościami związanymi z generowaniem sygnałów
  - TIM6, 7 – tylko blok bazy czasu
  - TIM1/8 – do 6 kanałów, 3 pary wyjść komplementarnych i jedno pojedyncze
  - TIM2/3/4/5 – 4 kanały
  - Pozostałe – 1 lub 2 kanały
- TIM2 jest 32-bitowy, pozostałe - 16-bitowe

TIM – blok licznika/bazy czasu

- Timer może zliczać cykle przebiegu zegarowego albo imuplsy zewnętrzne
- Preskaler umożliwia podział częstotliwości przebiegu przez dowolną liczbę
- Wartość okresu licznika programowana w rejestrze ARR
  - Przy osiągnięciu tej wartości licznik zeruje się lub zmienia kierunek zliczania
- Timer może zgłaszać przerwanie na końcu okresu
- Programowanie licznika:
  - Włączyć timer w RCC!
  - PSC – wartość preskalera
  - ARR – okres
  - DIER – włączanie przerwań
  - CR1 – uruchomienie timera

TIM – kanały porówniania/chwytania

- Timery TIMx (oprócz TIM6, TIM7) są wyposażone w 1..6 kanałów porównania/chwytania i związanych z nimi rejestrów CCRy i CCMRz
- W trybie porównania kanał porównuje wartość rejestru CCRy z wartością licznika CNT; przy zrównaniu wartości timer ustawia znacznik CCyIF w rejestrze SR oraz – opcjonalnie – generuje przerwanie lub zmienia stan wyjścia TIMxCHy
- W trybie chwytania wartość CNT jest kopiowana do CCRx przy wystąpieniu zdarzenia – zmiany stanu wejścia TIMxCHy; równocześnie jest ustawiany znacznik CCyIF i może być zgłoszone przerwanie

Sterowanie PWM

- Technika PWM (pulse width modulation) umożliwia cyfrowe i niemal bezstratne regulowanie mocy dostarczanej do odbiornika – urządzenia wykonawczego (np. źródła światła, silnika lub grzejnika) i tym samym regulację wybranego parametru urządzenia (jasności światła, prędkości ruchu, dostarczanego ciepła)
- Regulacja polega na załączaniu odbiornika na określony ułamek okresu regulacji i wyłączaniu go na pozostałą część okresu
- Do 4 kanałów porównania każdego timera TIMx może być użytych do generowania przebiegów PWM

Programowanie wyjść PWM

- Zaprogramować bazę czasu timera
- Ustawić odpowiedni tryb PWM dla wybranych wyjść w rejestrach CCMR1/2
- Ustawić wypełnienie w rejestrach CCRy
- Włączyć wyjścia PWM i określić ich polaryzację – rejestr CCER
- W timerach wyposażonych w blokadę PWM ustawić bit MOE w rejestrze BDTR