Programowanie mikrokontrolerów 2.0

Interfejs użytkownika

Marcin Engel Marcin Peczarski

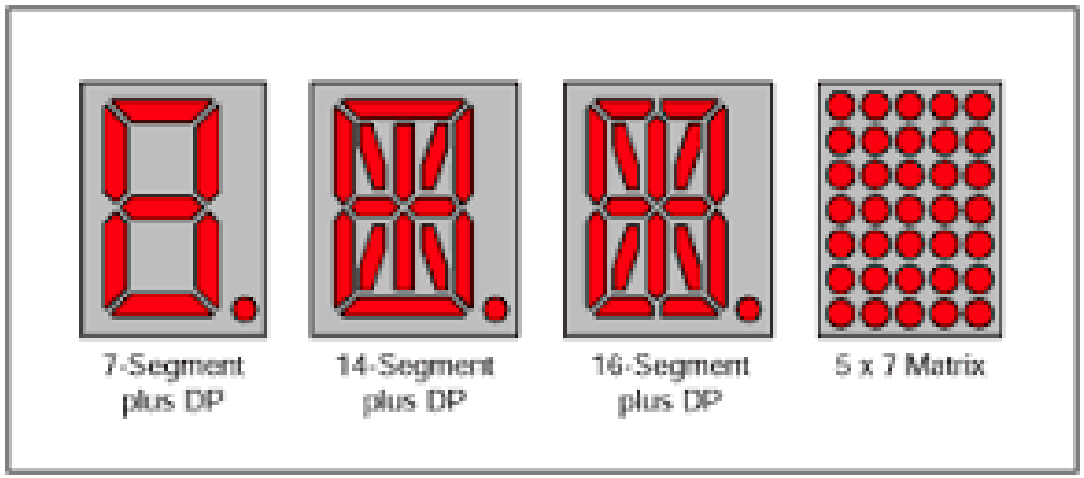
Instytut Informatyki Uniwersytetu Warszawskiego

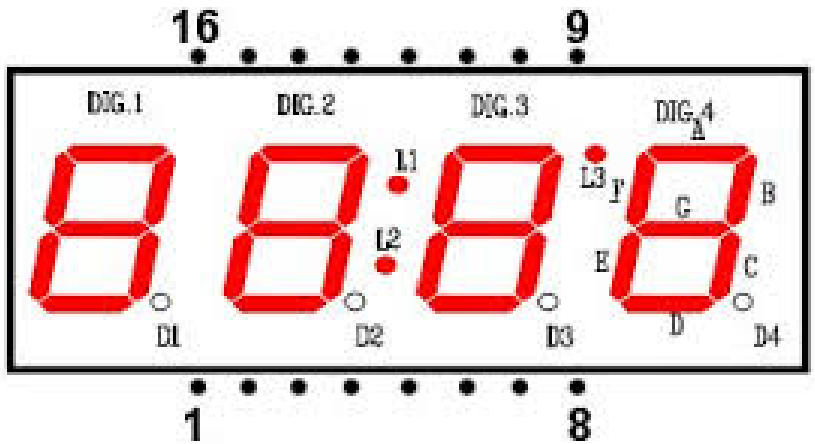
23 listopada 2021

Najczęściej spotykane urządzenia realizujące interfejs użytkownika w aplikacjach mikrokontrolerowych

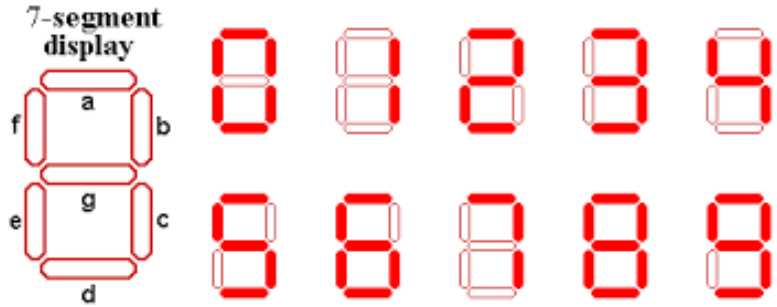
- Wyjście
  - Dioda świecąca, LED (ang. *light-emitting diode*)
  - Wyświetlacz LED: segmentowy albo matrycowy
  - Wyświetlacz ciekłokrystaliczny, LCD (ang. *liquid-crystal display*): matrycowy albo specjalizowany
  - Papier elektroniczny, e-papier (ang. *electronic paper*, *e-paper*, *electronic ink*, *e-ink*)
- Wejście
  - Przycisk
  - Dżojstik
  - Klawiatura matrycowa, np. 4×4
  - Enkoder obrotowy
  - Panel dotykowy, gładzik (ang. *touchpad*)
- Wyjście i wejście
  - Ekran dotykowy (ang. *touchscreen*), czyli połączenie LCD z przeźroczystym panelem dotykowym

Różne rodzaje wyświetlaczy diodowych





Wyświetlacz 7-segmentowy, idea działania



- Siedem diod świecących wyświetlających cyfry
- Zwykle też ósma dioda wyświetlająca punkt dziesiętny
- Dwie wersje
  - wspólna anoda – połączone anody wszystkich diod
  - wspólna katoda – połączone katody wszystkich diod
- Dalej opisujemy wyświetlacz ze wspólną anodą – drugi działa analogicznie
- Aby wyświetlić cyfrę
  - na anodę podajemy napięcie zasilania
  - odpowiednie katody łączymy do masy przez rezystory o wartości kilkaset omów

Wyświetlacz 7-segmentowy wielocyfrowy

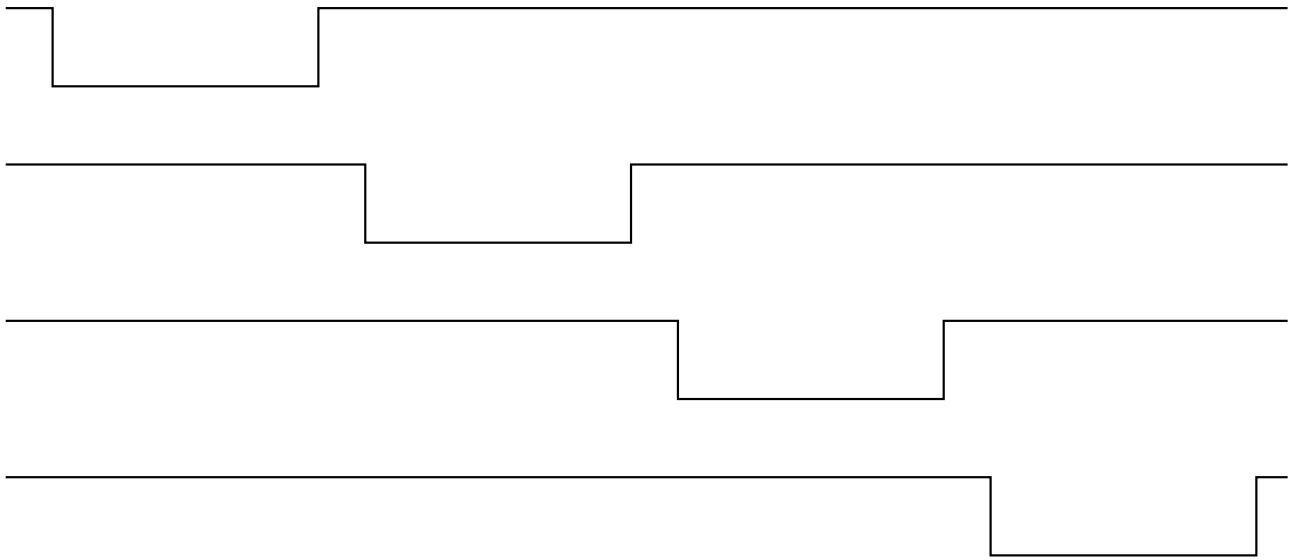
- Wyświetlacze łączy się w bloki kilkucyfrowe
- Osobne sterowanie każdej cyfry wymaga wielu wyprowadzeń mikrokontrolera: $8n$ dla n cyfr
- Często stosuje się sterowanie multipleksowane
  - katody tych samych segmentów we wszystkich cyfrach łączy się ze sobą
  - w danej chwili wyświetla się tylko jedną cyfrę, podając napięcie zasilania na jedną anodę i łącząc do masy (przez rezystory) odpowiednie katody
  - każdą cyfrę wyświetla się kilkadziesiąt razy na sekundę
  - wymaga $n + 8$ wyprowadzeń dla n cyfr

Wyświetlacz 7-segmentowy, sterowanie

- Katodami można sterować bezpośrednio z wyprowadzeń mikrokontrolera – aktywny poziom niski
- Maksymalny prąd wspólnej anody przekracza dopuszczalny prąd wyprowadzenia mikrokontrolera – sterowanie przez tranzystor – na wyjściu mikrokontrolera aktywny poziom niski
- Aby uniknąć cieni wyświetlanych cyfr na sąsiednich cyfrach, przełączenie z i-tej na $(i+1)$-szą cyfrę (numeracja modulo liczba cyfr) powinno odbywać się tak:
  - wyłącz katody cyfry i
  - wyłącz anodę cyfry i
  - włącz anodę cyfry $i+1$
  - włącz katody cyfry $i+1$

Wyświetlacz 7-segmentowy, sterowanie za pomocą licznika

- Sygnały sterujące czterema anodami powinny wyglądać tak:



- Do wytworzenia takich przebiegów należy wykorzystać licznik
- Anody i katody przełącza się w przerwaniach licznika, np.:
  - włączanie przy zdarzeniu uaktualnienia
  - wyłączanie przy zdarzeniu zgodności
- Dodatkową zaletą takiego sterowania jest możliwość regulowania jasności wyświetlacza przez dobór stosunku wartości, przy której następuje zdarzenie zgodności, do wartości maksymalnej, do której zlicza licznik

Wyświetlacz 7-segmentowy w zestawie laboratoryjnym

- Wyjścia sterujące anodami i katodami typu otwarty dren
- Anody
  - cyfra 1 – PB0
  - cyfra 2 – PA4
  - cyfra 3 – PA1
  - cyfra 4 – PA0
- Katody
  - a – PB6
  - b – PC7
  - c – PA9
  - d – PA8
  - e – PB10
  - f – PB4
  - g – PB5
  - dp – PB3
- Dlaczego tak dziwnie?

Magistrala SPI

- Akronim Serial Peripheral Interface
- Interfejs szeregowy, synchroniczny, dwukierunkowy (ang. *full-duplex*)
- Bardzo często wykorzystywana do komunikacji z różnorodnymi wyświetlaczami, a także z pamięciami EEPROM, Flash oraz kartami SD

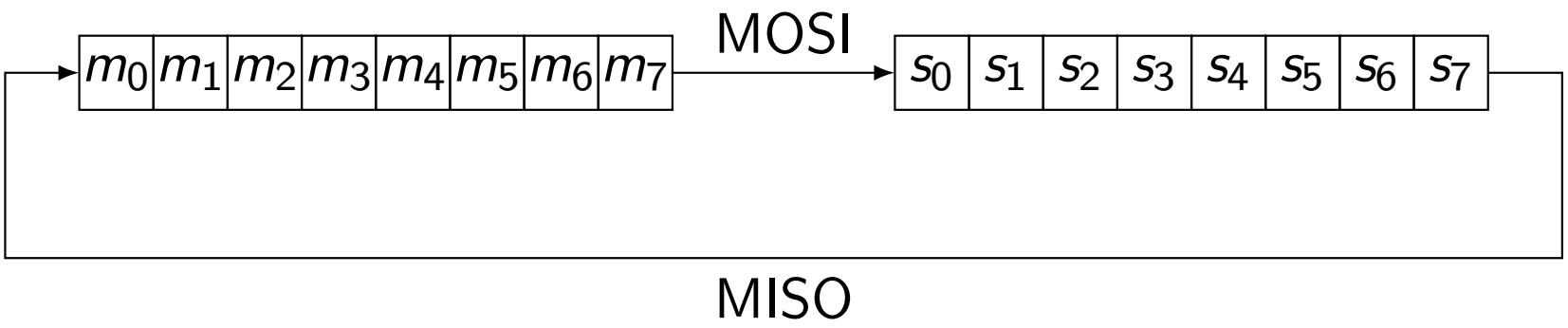
Warstwa fizyczna SPI

- Trzy wspólne linie
  - SCK lub CLK – jedno urządzenie (master) generuje na niej sygnał zegarowy
  - MOSI (Master Out Slave In) – master przesyła po niej dane do urządzenia slave
  - MISO (Master In Slave Out) – slave przesyła po niej dane do urządzenia master
- Do magistrali może być przyłączonych wiele urządzeń slave
  - dane są przesyłane zawsze między urządzeniem master a jednym z urządzeń slave
  - urządzenie master połączone jest z każdym urządzeniem slave za pomocą osobnej linii SS (Slave Select) lub CS (Chip Select)

Protokół komunikacyjny SPI

- Master i slave mają wewnętrzny rejestr przesuwny
- Master inicjuje komunikację, podając na wejście SS wybranego urządzenia slave stan niski
- Master i slave umieszczają dane do przesłania w swoich rejestrach przesuwnych
- Master generuje sygnał zegarowy na linii SCK
- W każdym cyklu zegara:
  - po linii MOSI jest przesyłany jeden bit z rejestru urządzenia master do rejestru urządzenia slave
  - po linii MISO jest przesyłany jeden bit z rejestru urządzenia slave do rejestru urządzenia master
  - zawartość rejestrów jest przesuwana o jeden bit

Protokół komunikacyjny SPI, cd.



- Po n cyklach zegara, gdzie n jest rozmiarem rejestru przesuwnego, wartość z rejestru urządzenia slave trafia do rejestru urządzenia master i na odwrót
- Po przesłaniu wszystkich bitów master podaje na wejście SS urządzenia slave stan wysoki
- Gdy żadne urządzenie slave nie jest wybrane, sygnał zegarowy nie jest generowany

Warianty pracy SPI

- Kolejność przesyłania bitów (najpierw LSB lub najpierw MSB) jest konfigurowalna
- Gdy sygnał zegarowy nie jest generowany, linia SCK jest w stanie spoczynkowym i może to być stan niski lub wysoki
- Bity są przesyłane przy nieparzystym lub parzystym zboczu sygnału zegarowego
- Zarówno nieparzyste, jak i parzyste zbocze sygnału zegarowego może być zboczem narastającym lub opadającym, zależnie od stanu spoczynkowego sygnału zegarowego
- Mamy zatem osiem kombinacji ustawień

SPI w STM32

- Mikrokontrolery STM32 mają po kilka układów peryferyjnych obsługujących magistralę SPI
- Jak zwykle trzeba poustawiać odpowiednie bity we właściwych rejestrach
- Można, a nawet należy, korzystać z przerwań i DMA
- Szczegóły w dokumentacji

LCD

- Wyświetlacz ciekłokrystaliczny składa się z matrycy ciekłokrystalicznej i sterownika
- Sterownik może być zintegrowany z matrycą w postaci modułu LCD
  - komunikacja między mikrokontrolerem a sterownikiem odbywa się za pomocą interfejsu szeregowego (np. SPI) lub równoległego
- Sterownik może być zintegrowany w mikrokontrolerze
  - niektóre modele STM32 mają taki sterownik
  - sterownik LCD widoczny jako układ peryferyjny, dostępny poprzez odpowiednie rejestry

Moduł LCD

- Do zestawu laboratoryjnego możemy podłączyć moduł wyświetlacza graficznego ze sterownikiem ST7735S
- Rozdzielczość ekranu: 128×160 pikseli
- Głębia koloru: 8, 9, 16 lub 18 bitów na piksel
- Interfejs to nieco zmodyfikowany, jednokierunkowy SPI, w dokumentacji sterownika nazywany „4-line Serial Interface”
  - SCK (w dokumentacji sterownika SCL) – sygnał zegarowy
  - SDA – linia danych, odpowiednik MOSI, 8 bitów transmitowanych szeregowo
  - A0 (w dokumentacji sterownika D/CX) – dodatkowy 9. bit wybierający między danymi a poleceniami
  - CS (w dokumentacji sterownika CSX) – linia wyboru modułu
- Dokumentacja sterownika w katalogu `/opt/arm/stm32/doc` w pliku `ST7735S_V1.1_20111121.pdf`

Złącze modułu LCD

- Modułu LCD do zestawu laboratoryjnego podłącza się ośmioma przewodami
- Przykład

Pin	Złącze CN7		Opis
LED	12	IOREF	zasilanie diod podświetlających
SCK	1	PC10	sygnał zegarowy
SDA	3	PC12	linia danych
A0	2	PC11	bit wybierający dane lub polecenie
RESET	14	RESET	zerowanie
CS	4	PD2	wybór modułu
GND	8	GND	masa
VCC	16	+3V3	zasilanie modułu

Sterownik LCD

- Nie będziemy omawiać, jak obsługuje się moduł LCD
- W katalogach `/opt/arm/stm32/inc`, `/opt/arm/stm32/src` umieściliśmy pliki z implementacją bardzo prostego sterownika
- Programowa obsługa SPI
- Tylko tryb tekstowy
  - deklaracja interfejsu programowego w pliku `lcd.h`

    ```
    void LCDconfigure(void);
    void LCDclear(void);
    void LCDgoto(int textLine, int charPos);
    void LCDputchar(char c);
    void LCDputcharWrap(char c);
    ```
  - implementacja w pliku `lcd.c`
  - czcionki o rozmiarach 8×16, 10×18, 12×24 i 14×28 w plikach `fonts.h`, `fonts.c`
- Pliki `lcd_board_def.h`, `xcat.h` pozwalają uniezależnić implementację od wyboru konkretnych wyprowadzeń, do których podłącza się moduł, choć nie zalecamy zmieniania proponowanej konfiguracji

Sterownik LCD ze sprzętową obsługą SPI i korzystający z DMA, wskazówki

- Korzystamy z układu `SPI3` oraz kanału 0 i strumienia 5 `DMA1`
- Wyprowadzenia CS i A0 obsługujemy programowo i konfigurujemy jak w wyżej omówionym sterowniku
- Na wyprowadzeniach SDA i SCK konfigurujemy funkcję alternatywną `GPIO_AF_SPI3`, wyjścia przeciwsobne o maksymalnej szybkości, bez rezystorów podciągających (parametry `GPIO_OType_PP`, `GPIO_High_Speed`, `GPIO_PuPd_NOPULL`)
- Nie zapominamy o włączeniu taktowania układów `GPIO`*x*, `SPI3` i `DMA1` w odpowiednich rejestrach układu `RCC`

Konfiguracja SPI i DMA

- SPI ustawiamy w trybie *master* (bit `MSTR`), z programowym sterowaniem linią CS (bity `SSM`, `SSI`) oraz opcjonalnie jako interfejs jednoprzewodowy (bity `BIDIMODE`, `BIDIOE`)

```
SPI3->CR1 = SPI_CR1_MSTR |
            SPI_CR1_SSM | SPI_CR1_SSI |
            SPI_CR1_BIDIMODE | SPI_CR1_BIDIOE;
```

- Bity `BR[2:0]` w rejestrze `CR1` pozostawiamy wyzerowane, co daje maksymalną szybkość transmisji: $f_{PCLK1}/2$
- Dane będziemy wysyłać za pomocą DMA

```
SPI3->CR2 = SPI_CR2_TXDMAEN;
```

- Będziemy korzystać z przerwania DMA

```
NVIC_EnableIRQ(DMA1_Stream5_IRQn);
```

- Na koniec włączamy układ SPI

```
SPI3->CR1 |= SPI_CR1_SPE;
```

Inicjowanie wysyłania danych po SPI

- Dane z bufora w pamięci przesyłamy do układu `SPI3` za pomocą `DMA1`
- Inicjowanie transferu DMA musi być nieblokujące
- Transfery DMA muszą być kolejkowane
  - transfer można zainicjować, jeśli DMA nie jest zajęte
  - jeśli DMA jest zajęte, wstawiamy żądanie do kolejki
- Jeśli DMA jest wolne, aktywujemy linię CS i inicjujemy transfer DMA

Inicjowanie transferu DMA

- Ustawiamy odpowiednio linię A0
- Konfigurujemy
  - dokąd mają być przesyłane dane – rejestr `PAR`
  - skąd mają być kopiowane – rejestr `MOAR`
  - liczbę bajtów do przesłania – rejestr `NDTR`

```
DMA1_Stream5->PAR = (uint32_t)&SPI3->DR;
DMA1_Stream5->MOAR = ...;
DMA1_Stream5->NDTR = ...;
```

Inicjowanie transferu DMA, cd.

- Konfigurujemy parametry transferu
  - w rejestrze `CR` musimy ustawić
    - nr kanału – bity `CHSEL[3:0]`
    - kierunek przesyłania danych – bity `DIR[1:0]`
    - przerwanie po zakończeniu transferu – bit `TCIE`
  - w rejestrze `CR` możemy opcjonalnie ustawić
    - priorytet – bity `PL[1:0]`
    - zwiększanie adresu pamięci po każdym odczycie – bit `MINC`
    - rozmiar dostępu do pamięci – bity `MSIZE[1:0]`
    - rozmiar paczki – bity `MBURST[1:0]`
  - w rejestrze `FCR` możemy
    - włączyć FIFO – bit `DMDIS`
    - ustawić próg FIFO – bity `FTH[1:0]`

```
DMA1_Stream5->CR = DMA_SxCR_PL_1 |
                   DMA_SxCR_DIR_0 |
                   DMA_SxCR_TCIE | ...;
DMA1_Stream5->FCR = ...;
```

- Startujemy transfer

```
DMA1_Stream5->CR |= DMA_SxCR_EN;
```

Obsługa zakończenia transferu DMA i transmisji SPI

- Zakończenie transferu DMA jest sygnalizowane za pomocą przerwania: ustawiany jest bit `TCIF5` w rejestrze `DMA1->HISR`

```
void DMA1_Stream5_IRQHandler() {
  uint32_t isr = DMA1->HISR;
  if (isr & DMA_HISR_TCIF5) {
    ...
  }
}
```

- W procedurze obsługi przerwania zerujemy znacznik zdarzenia

```
DMA1->HIFCR = DMA_HIFCR_CTCIF5;
```

- Zakończenie transferu DMA nie oznacza jednak zakończenia transmisji SPI
  - czekamy na ustawienie bitu `TXE` w rejestrze `SPI3->SR`
  - czekamy na wyzerowanie bitu `BSY` w rejestrze `SPI3->SR`
- Po zakończeniu transmisji SPI
  - jeśli kolejka nie jest pusta, inicjujemy następny transfer DMA
  - jeśli kolejka jest pusta, dezaktywujemy linię CS

Klawiatura matrycowa

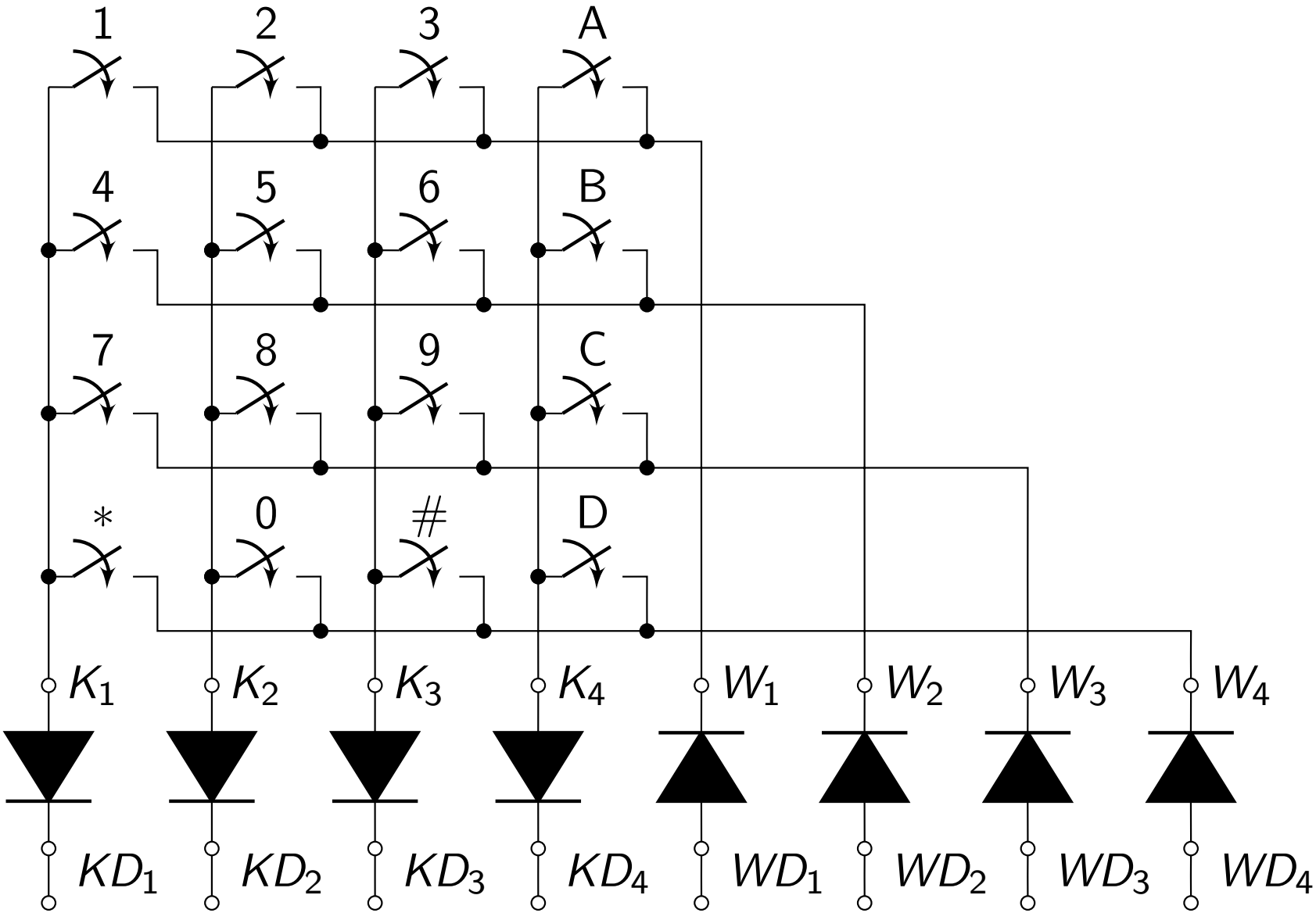
- Gdy mamy do obsłużenia wiele klawiszy, to możemy oczywiście każdy podłączyć do innego wejścia mikrokontrolera
- Przy dużej liczbie klawiszy zabraknie portów w mikrokontrolerze
- Dużo efektywniejszym sposobem jest połączenie klawiszy w matrycę
- Tak skonstruowana jest np. klawiatura w PC

Klawiatura matrycowa 16-klawiszowa

- Bardzo popularnym typem klawiatury matrycowej jest klawiatura 4×4

- 16 klawiszy ułożonych w 4 wiersze i 4 kolumny
- Można spotkać np. w bankomatach, czytnikach kodów otwierających drzwi (domofony) itp.

Schemat modułu klawiatury 4×4



Moduł klawiatury 4×4

- Naciśnięcie klawisza zwiera linię wiersza z linią kolumny
- W mikrokontrolerze potrzebujemy po jednym wyprowadzeniu na każdy wiersz i na każdą kolumnę
- Zauważmy, że przy wciskaniu więcej niż jednego klawisza pewne kombinacje wciśniętych klawiszy są nierozróżnialne – proszę podać przykład
- Można wymyślić wiele algorytmów obsługi takiej klawiatury
- Na kolejnych slajdach przedstawiamy naszą propozycję

Moduł klawiatury 4×4, cd.

- Do kolumn można się podpiąć
  - bezpośrednio, odpowiednio do pinów K_1 do K_4
  - przez diody, odpowiednio do pinów KD_1 do KD_4
- Do wierszy można się podpiąć
  - bezpośrednio, odpowiednio do pinów W_1 do W_4
  - przez diody, odpowiednio do pinów WD_1 do WD_4
- Piny KD_1 do KD_4 i WD_1 do WD_4 są zdublowane
- Dla ustalenia uwagi, będziemy korzystać
  - z pinów kolumn KD_1, KD_2, KD_3, KD_4, które podłączymy odpowiednio do wyprowadzeń `PC0`, `PC1`, `PC2`, `PC3`
  - z pinów wierszy W_1, W_2, W_3, W_4, które podłączymy odpowiednio do wyprowadzeń `PC6`, `PC7`, `PC8`, `PC9`
  - Dlaczego tak?

Algorytm inicjowania klawiatury 4×4

- Skonfiguruj `TIMx`, aby zgłaszał przerwanie w regularnych odstępach czasu, np. co 10 ms, ale nie włączaj licznika
- Ustaw stan niski na wyprowadzeniach kolumn
- Skonfiguruj wyprowadzenia kolumn jako wyjścia przeciwsobne bez rezystorów podciągających
- Skonfiguruj wyprowadzenia wierszy jako wejścia z rezystorami podciągającymi, zgłaszające przerwanie przy zboczu opadającym
- Wyzeruj znaczniki przerwań wierszy – przez wpisanie jedynek do odpowiedniego rejestru
- Uaktywnij przerwanie wierszy w układzie `NVIC`
- Kolejność inicjowania jest istotna

Algorytm obsługi klawiatury 4×4

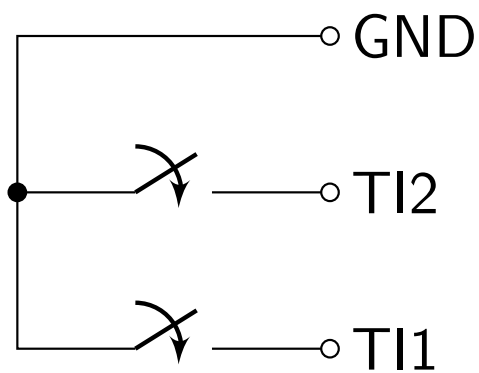
- W procedurze obsługi przerwania `EXTI`
  - dezaktywuj przerwania wierszy w układzie `EXTI`
  - wyzeruj znaczniki przerwań wierszy w układzie `EXTI` – przez wpisanie jedynek do odpowiedniego rejestru
  - ustaw stan wysoki na liniach kolumn
  - wyzeruj rejestr `CNT` licznika
  - uruchom licznik `TIMx`
- W procedurze obsługi przerwania `TIMx`
  - wyzeruj znacznik przerwania `TIMx` – przez wpisanie zera do odpowiedniego rejestru
  - skanuj stan klawiatury
  - jeśli nic nie jest wciśnięte
    - zatrzymaj licznik
    - ustaw stan niski na liniach kolumn
    - wyzeruj znaczniki przerwań wierszy w układzie `EXTI` – przez wpisanie jedynek do odpowiedniego rejestru
    - aktywuj przerwania wierszy w układzie `EXTI`

Algorytm skanowania klawiatury 4×4

- Dla każdej kolumny
  - ustaw stan niski na wyprowadzeniu tej kolumny
  - wczytaj stan wierszy – tu jest haczyk
  - ustaw stan wysoki na wyprowadzeniu tej kolumny
  - decyduj, który klawisz w tej kolumnie należy uznać za wciśnięty i czy jest to przytrzymanie
  - umieść informację o wciśniętym klawiszu w kolejce zdarzeń do obsłużenia

Enkoder obrotowy

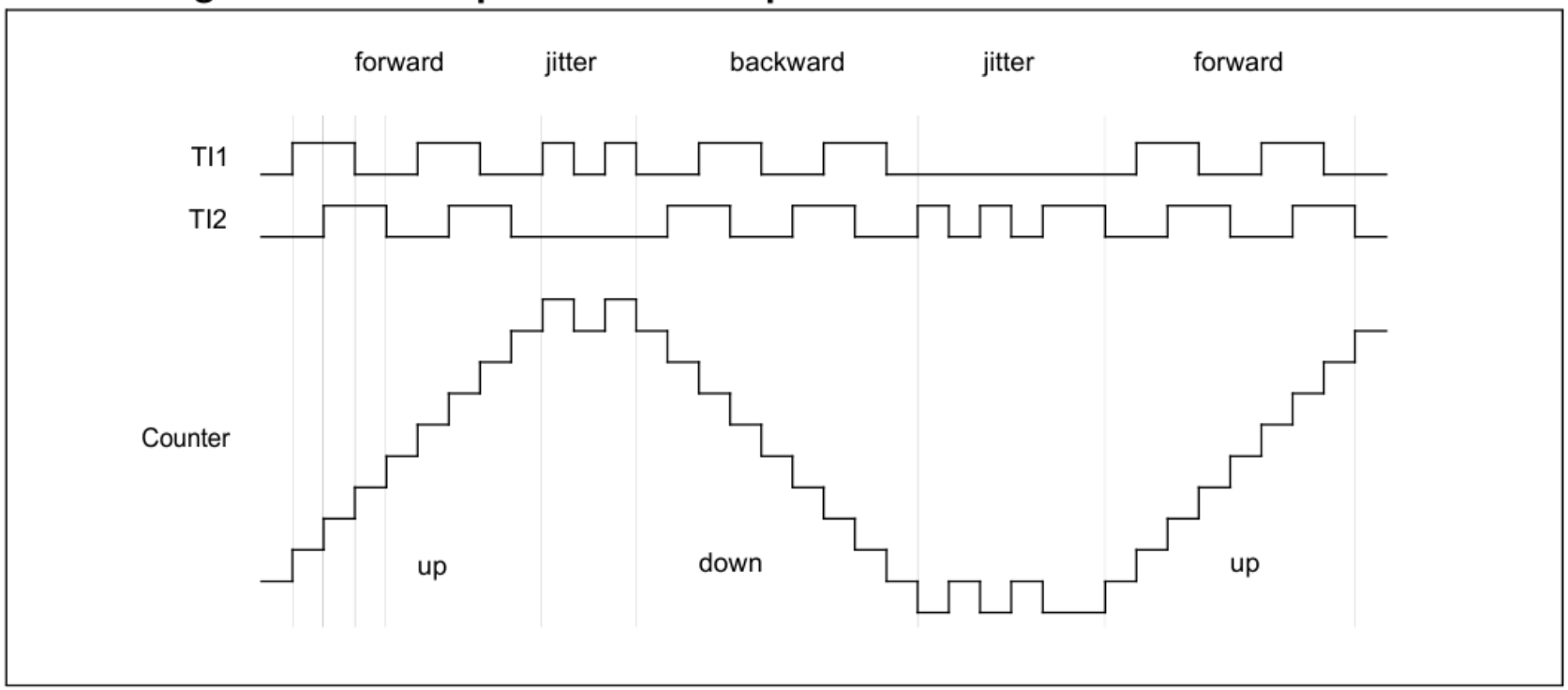
- Bardzo popularnym elementem sterującym jest enkoder



- Może być mechaniczny lub optyczny
- Znajduje zastosowanie do regulacji siły głosu (zamiast klasycznego potencjometru), do wyboru pozycji z menu, ustawiania czasu itp.

Zasada pracy enkodera obrotowego

Figure 121. Example of counter operation in encoder interface mode



Źródło: RM0383 – Reference manual, STM32F411xC/E advanced ARM-based 32-bit MCUs

Moduł enkodera

- Widok złącza, patrząc od góry

  - GND – masa
  - NC – niepodłączony
  - TI2 – styk 2
  - NC – niepodłączony
  - TI1 – styk 1

Obsługa enkodera w STM32

- Liczniki w STM32 mają trzy tryby pracy przeznaczone do obsługi enkoderów
- Wykorzystamy licznik `TIM3`
- Styk TI*y* podłączymy do wejścia `TIM3_CH`*y* licznika
- Jako `TIM3_CH1` użyjemy wejścia `PB4` mikrokontrolera, a jako `TIM3_CH2` – `PB5`
- Do `PB4` i `PB5` podłączone są odpowiednio przyciski „w prawo” i „do góry” dżojstika – pozwala to na zaobserwowanie, jak zbocza na tych wejściach wpływają na wartość licznika

Przykład konfiguracji enkodera (1)

- Kolejność włączania sygnałów zegarowych może być istotna – dlaczego?

```
RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;
RCC->AHB1ENR |= RCC_AHB1ENR_GPIOBEN;
```

- Zacznij od skonfigurowania licznika

```
TIM3->CR1 = TIM_CR1_URS;
```

- Ustaw tryb 3 obsługi enkodera: licznik zlicza w górę i w dół przy zboczach na obu wejściach, zależnie od poziomu na drugim wejściu

```
TIM3->SMCR = TIM_SMCR_SMS_1 | TIM_SMCR_SMS_0;
```

Przykład konfiguracji enkodera (2)

- W rejestrze `TIMx->CCMR1` konfiguruje się wejścia i wyjścia kanałów 1 i 2 licznika x
  - bit `CC1S_0` ustawia `TIMx_CH1` jako wejście sygnału TI1
  - bit `CC2S_0` ustawia `TIMx_CH2` jako wejście sygnału TI2
  - bity `ICyF` włączają filtr cyfrowy dla sygnałów TIy, likwidujący przypadkowe zliczenia

```
TIM3->CCMR1 = TIM_CCMR1_CC1S_0 | TIM_CCMR1_CC2S_0 |
              TIM_CCMR1_IC1F_0 | TIM_CCMR1_IC1F_1 |
              TIM_CCMR1_IC2F_0 | TIM_CCMR1_IC2F_1;
```

Przykład konfiguracji enkodera (3)

- W rejestrze `TIMx->CCER` za pomocą bitu `CCyP` ustawia się kierunek zliczania kanału y licznika x
- Przy podłączeniach jak w tym przykładzie obracanie pokrętła w prawo zwiększa, a w lewo – zmniejsza wartość licznika
  - oba bity wyzerowane

    ```
    TIM3->CCER = 0;
    ```
  - ten sam efekt uzyskuje się, ustawiając oba bity

    ```
    TIM3->CCER = TIM_CCER_CC1P | TIM_CCER_CC2P;
    ```
- Odwrócenie kierunku zliczania (obracanie w prawo zmniejsza, a w lewo – zwiększa wartość licznika) można uzyskać
  - odwracając kierunek zliczania wejścia TI1

    ```
    TIM3->CCER = TIM_CCER_CC1P;
    ```
  - albo odwracając kierunek zliczania wejścia TI2

    ```
    TIM3->CCER = TIM_CCER_CC2P;
    ```

Przykład konfiguracji enkodera (4)

- Ustawienie preskalera wydaje się być istotne

```
TIM3->PSC = 0;
```

- Ustaw maksymalną wartość, po doliczeniu do której licznik się przekręci, np. liczbę zliczeń przy pełnym obrocie pomniejszoną o jeden

```
TIM3->ARR = 95;
TIM3->EGR = TIM_EGR_UG;
```

- Przypominamy, że ustawianie bitu `UG` w rejestrze `TIMx->EGR` wpływa na wartość rejestru `TIMx->CNT`
- W trybie obsługi enkodera bit `DIR` w rejestrze `TIMx->CR1` jest ustawiany sprzętowo, zależnie od aktualnego kierunku zliczania

Przykład konfiguracji enkodera (5)

- Skonfiguruj wejścia licznika, funkcja alternatywna 2

```
GPIOafConfigure(GPIOB, 4, GPIO_OType_PP,
                GPIO_Low_Speed, GPIO_PuPd_UP,
                GPIO_AF_TIM3);
GPIOafConfigure(GPIOB, 5, GPIO_OType_PP,
                GPIO_Low_Speed, GPIO_PuPd_UP,
                GPIO_AF_TIM3);
```

  - ponieważ są to wejścia, parametry `GPIO_OType_PP` i `GPIO_Low_Speed` nie mają znaczenia
  - w zestawie laboratoryjnym są zewnętrzne rezystory podciągające `PB4` i `PB5`, ale gdyby ich nie było, trzeba użyć wewnętrznych – parametr `GPIO_PuPd_UP`
- Konfigurowanie wejść powoduje wygenerowanie zboczy
- Włącz licznik dopiero po skonfigurowaniu wejść

```
TIM3->CR1 |= TIM_CR1_CEN;
```

Korzystanie z enkodera

- Można czytać wartość licznika – rejestr `TIMx->CNT`
- Działają przerwania związane ze zdarzeniami zgodności i uaktualnienia wartości licznika
- Można nakłonić DMA do regularnego kopiowania wartości licznika do bufora w pamięci
- Można skonfigurować przerwania `EXTI` na liniach enkodera

Przerwania na liniach enkodera

- Włącz taktowanie układu `SYSCFG` w rejestrze `RCC->APB2ENR`
- Skonfiguruj przerwania za pomocą funkcji `GPIOinConfigure` z parametrami `GPIO_PuPd_NOPULL`, `EXTI_Mode_Interrupt`, `EXTI_Trigger_Rising_Falling`, ale przed wywołaniem funkcji `GPIOafConfigure`
- Skasuj znaczniki przerwań w rejestrze `EXTI->PR`
- Jeśli potrzebujesz, skonfiguruj priorytet przerwań
- Włącz przerwania w układzie `NVIC` za pomocą funkcji `NVIC_EnableIRQ`
- W procedurze obsługi przerwania, jeśli jest to przerwanie enkodera (sprawdź wartość `EXTI->IMR & EXTI->PR`)
  - skasuj znacznik przerwania w rejestrze `EXTI->PR`
  - odczytaj wartość licznika `TIMx->CNT`

Uwagi końcowe

- Obsługa interfejsu użytkownika
  - jest czasochłonna (zwłaszcza LCD)
  - nie jest krytyczna czasowo
- W procedurach obsługi przerwań nie należy wykonywać czasochłonnych czynności
- Czynności czasochłonne i mało krytyczne czasowo należy umieszczać w procedurach przerwań o niskim priorytecie wywłaszczania albo wręcz poza procedurami przerwań w głównej pętli programu
- Należy zadbać o synchronizację dostępu do struktur danych modyfikowanych w procedurach działających z różnymi priorytetami wywłaszczania