KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

# „Pamięci i urządzenia peryferyjne”
# „Wprowadzenie do przedmiotu”

Prezentacja jest współfinansowana przez
Unię Europejską w ramach
Europejskiego Funduszu Społecznego w projekcie pt.

*„Innowacyjna dydaktyka bez ograniczeń - zintegrowany rozwój Politechniki Łódzkiej - zarządzanie Uczelnią, nowoczesna oferta edukacyjna i wzmacniania zdolności do zatrudniania osób niepełnosprawnych”*

**Dariusz Makowski**

**Katedra Mikroelektroniki i Technik Informatycznych**

**tel. 631 2720**

**dmakow@dmcs.pl**

**http://neo.dmcs.p.lodz.pl/PiUP**

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

# Sprawy formalne

- Informacje ogólne
- Zaliczenie
- Laboratorium
- Materiały do wykładu

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

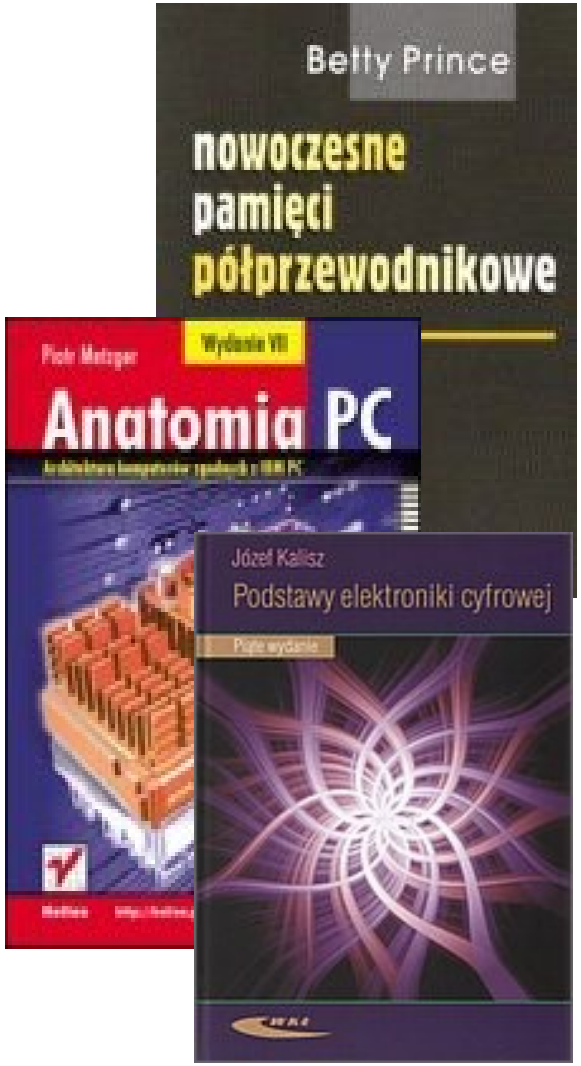
Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

# Literatura

## Literatura obowiązkowa:

- Materiały wykładowe i laboratoryjne
- P. Betty, „Nowoczesne pamięci półprzewodnikowe. Architektura i organizacja układów pamięci DRAM i SRAM”, WNT, 1999
- P. Metzger, „Anatomia PC”, Helion, 2006
- J. Kalisz, „Podstawy elektroniki cyfrowej”, WKŁ, 2008



# Zakres przedmiotu

- Wprowadzenie
- Układy peryferyjne
- Transmisja danych – interfejsy elektryczne
- Pamięci w systemach mikroprocesorowych
- Programowalne układy dekoderów adresowych

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

# Interfejsy w systemach wbudowanych

# Definicje podstawowe (1)

- **Pamięć komputerowa (ang. Computer Memory)**

Pamięć komputerowa to urządzenie elektroniczne lub mechaniczne służące do przechowywania danych i programów (systemu operacyjnego oraz aplikacji).

- **Urządzenia zewnętrzne, peryferyjne (ang. Peripheral Device)**

Urządzenia elektroniczne dołączone do procesora przez magistrale systemową lub interfejs. Urządzenia zewnętrzne wykorzystywane są do realizowania specjalizowanej funkcjonalności systemu.

- **Magistrala (ang. bus)**

Połączenie elektryczne umożliwiające przesyłanie danym pomiędzy procesorem, pamięcią i urządzeniami peryferyjnymi. Magistra systemowa zbudowane jest zwykłe z kilkudziesięciu połączeń elektrycznych (ang. Parallel Bus) lub szeregowego połączenia (ang. Serial Bus).

- **Interface (ang. Interface)**

Urządzenie elektroniczne lub optyczne pozwalające na komunikację między dwoma innymi urządzeniami, których bezpośrednio nie da się ze sobą połączyć.

# Współpraca procesora z urządzeniami peryferyjnymi (1)

## Interfejsy wykorzystywane w systemach mikroprocesorowych:

- Interfejs równoległy.
- Interfejsy szeregowe:
  - Interfejs DBGU - zgodny ze standardem EIA RS232,
  - Interfejs uniwersalny USART,
  - Interfejs Serial Peripheral Interface (SPI),
  - Interfejs Synchronous Serial Controller (SSC),
  - Interfejs I2C, Interfajs Two-wire Interface (TWI),
  - Interfejs Controlled Area Network (CAN),
  - Interfejs Universal Serial Bus (USB),
  - Interfejs Ethernet (10/100, 1 Gb, 10 Gb, 100 Gb),
  - Magistrala PCI,
  - Magistrala PCIe (PCI express),
  - RapidIO, Serial RapidIO,
  - ...

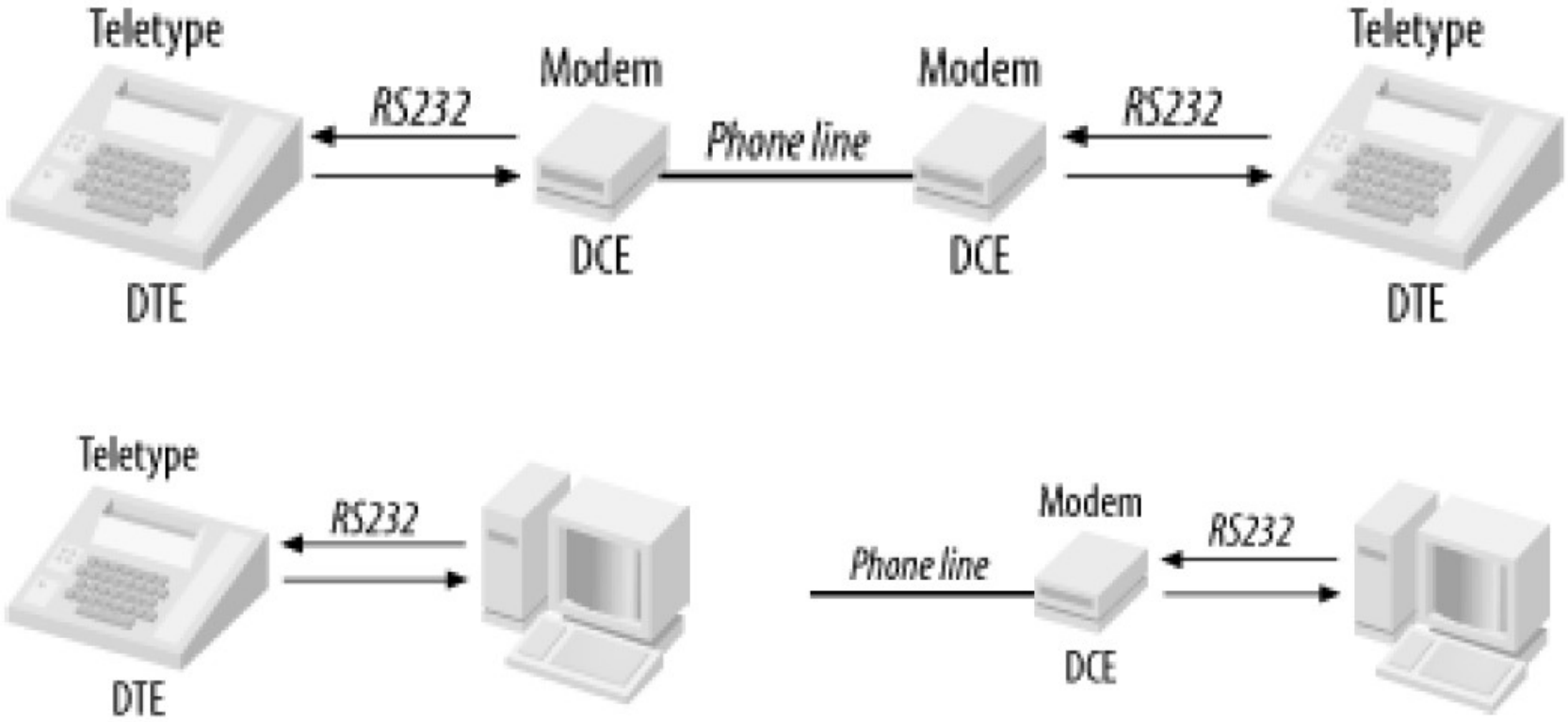
# Współpraca procesora z urządzeniami peryferyjnymi

## Interfejsy dostępne w procesorach rodziny ARM:

- Interfejs równoległy PIO (zwykle 32 bity),
- Interfejsy szeregowe:
  - Interfejs DBGU - zgodny ze standardem EIA RS232,
  - Interfejs uniwersalny USART,
  - Interfejs Serial Peripheral Interface (SPI),
  - Interfejs Synchronous Serial Controller (SSC),
  - Interfejs I2C, Interfajs Two-wire Interface (TWI),
  - Interfejs Controlled Area Network (CAN),
  - Interfejs Universal Serial Bus (USB),
  - Interfejs Ethernet 10/100.

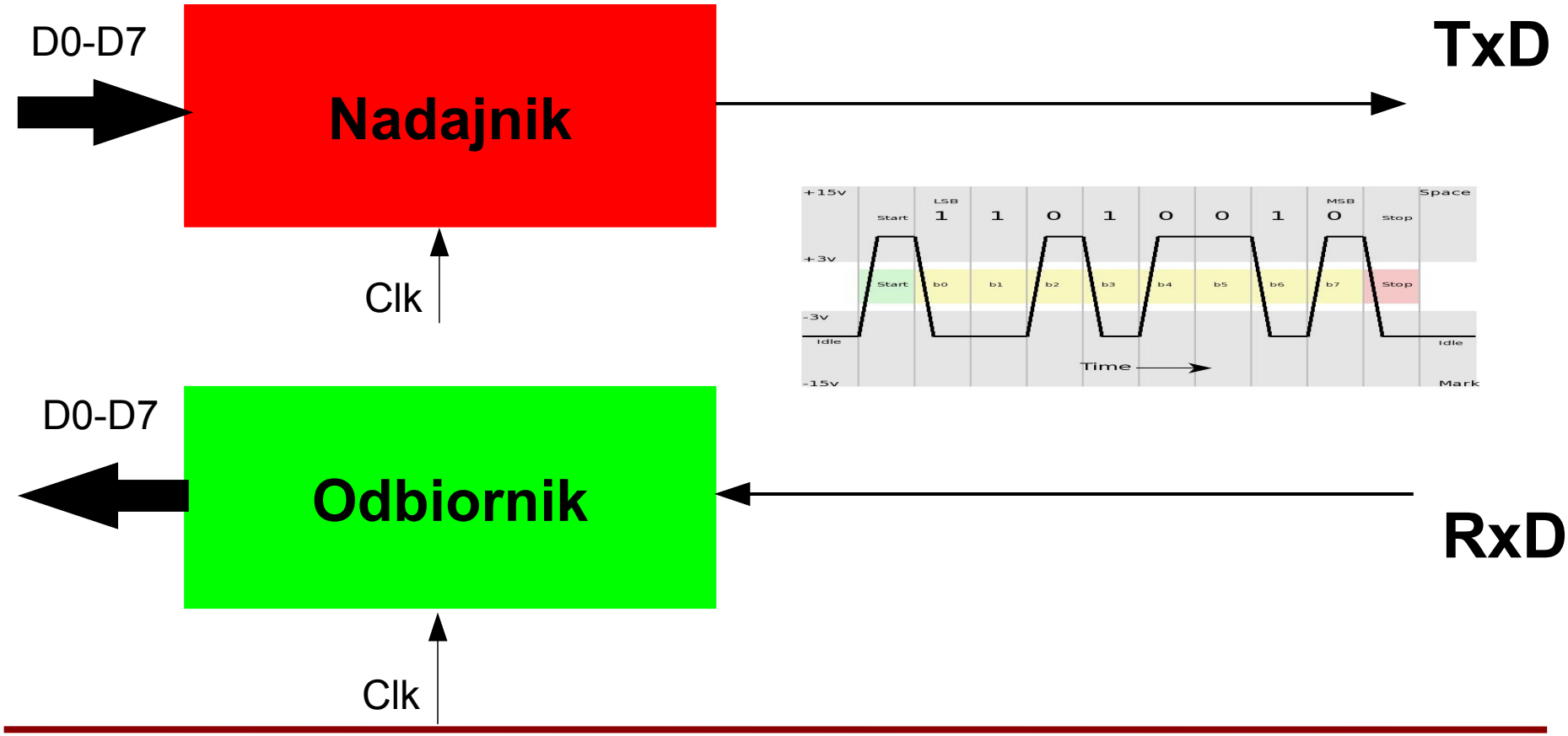
# Moduł transceivera szeregowego UART (Universal Asynchronous Receiver/Transmitter module)

# Interfejs szeregowy EIA RS232



# Transceiver UART

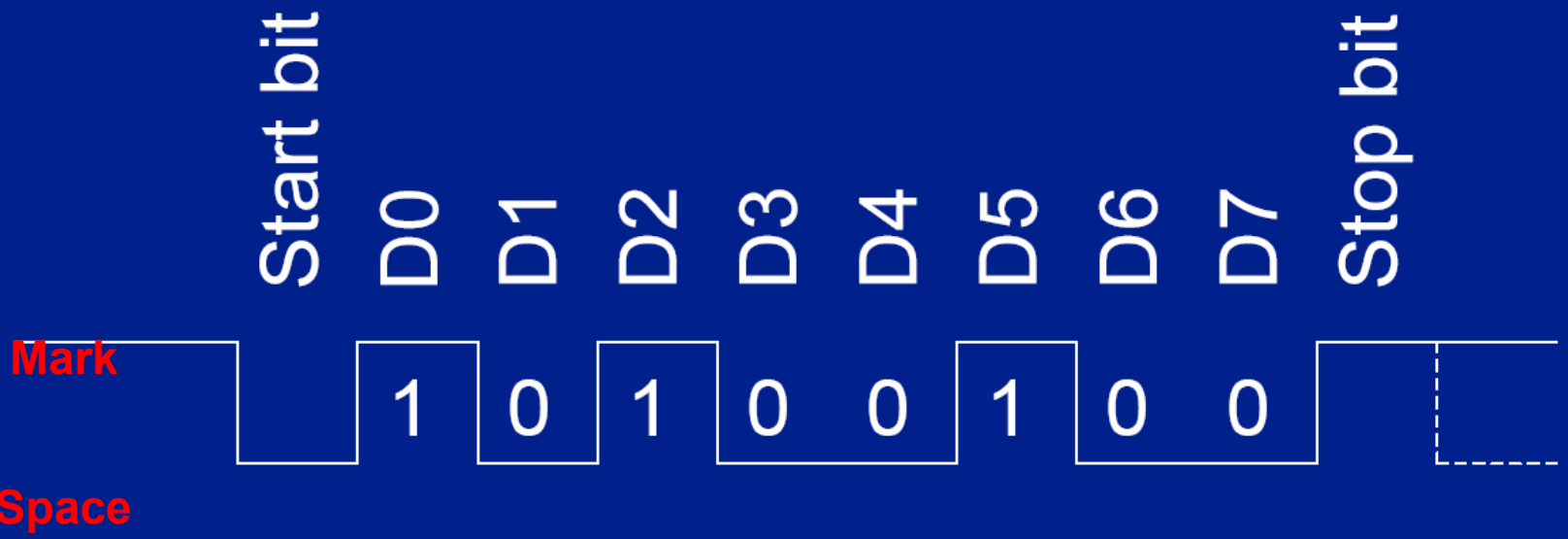
## Rejestr przesuwny



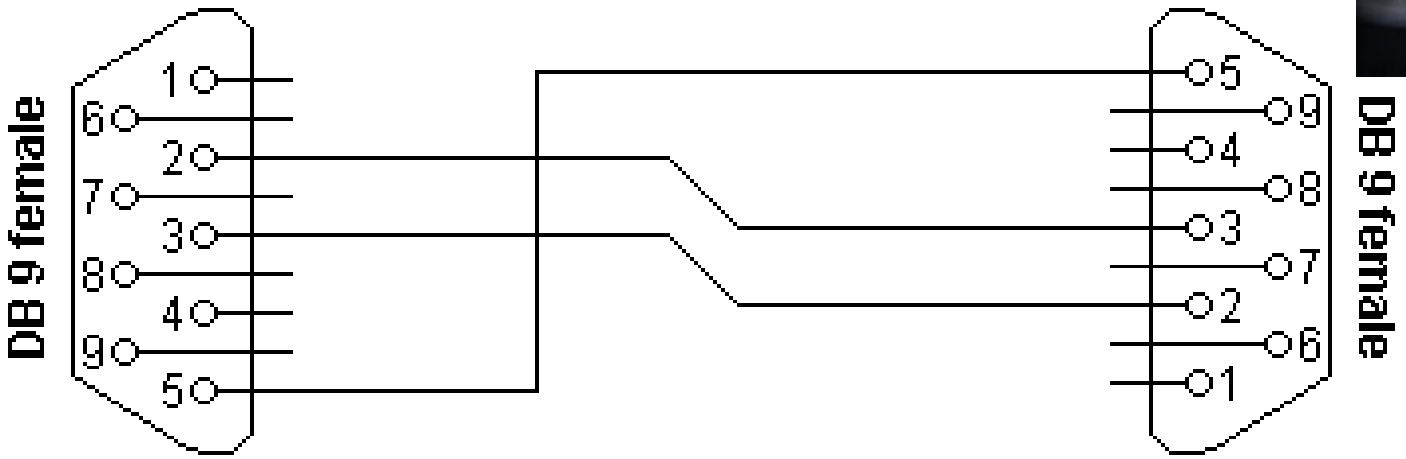
# Ramka danych transmitera UART (1)

## Asynchronous 8 bit waveform example

- Data is H'25' = B'00100101'



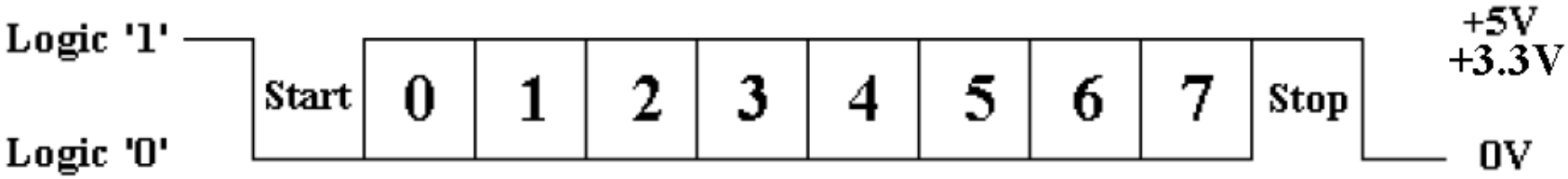
# Kabel null-modem EIA 232



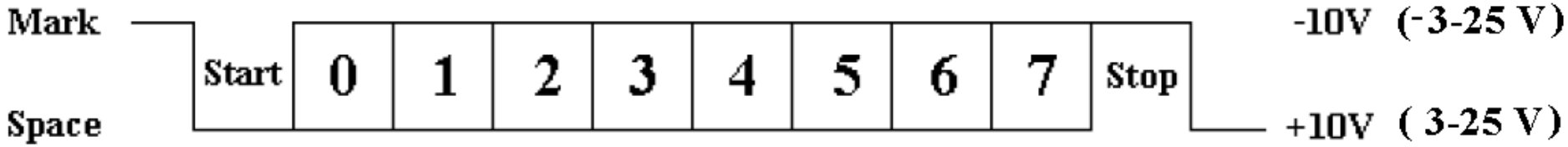

| Connector 1 | Connector 2 | Function |
|---|---|---|
| 2 | 3 | Rx ← Tx |
| 3 | 2 | Tx → Rx |
| 5 | 5 | Signal ground |

# Poziomy napięć określone przez standard EIA 232

## Wyjście procesora




## Standard EIA RS 232

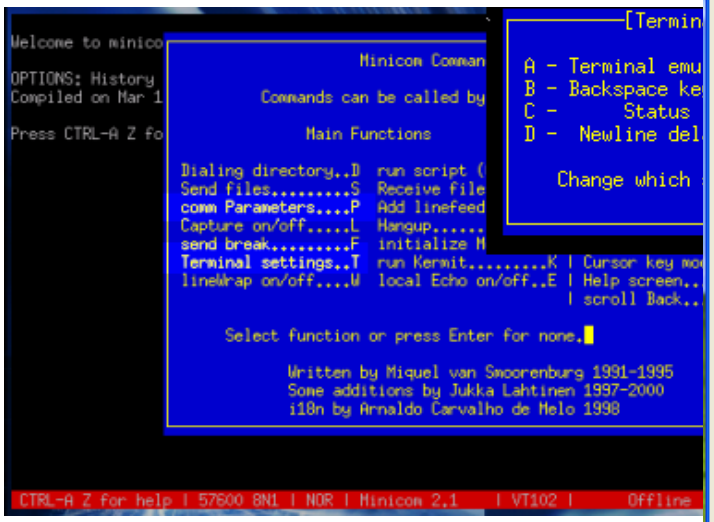



RS-232 Logic Waveform

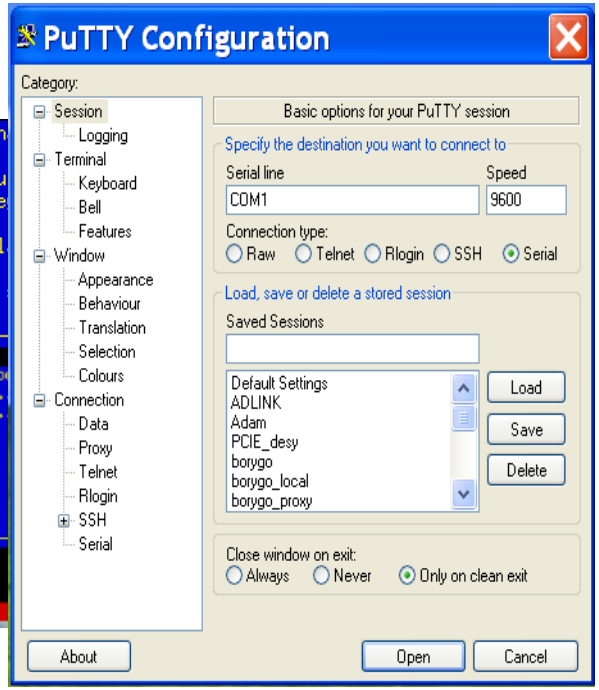
# Programy do komunikacji z wykorzystaniem standardu EIA RS232

## Program Hyper terminal

## Program minicom




## Program ssh




## Program Terminal

(http://www.elester-pkp.com.pl/index.php?id=92&lang=pl&zoom=0)

# Program Minicom

**Skróty klawiszy:**

- Ctrl + A → O – konfiguracja
- Ctrl + A → X – wyjście z programu
- Ctrl + A → C – wyczyszczenie ekranu

Okno konfiguracji

```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+--------------------------+
```

Okno konfiguracji parametrów transmisji

```
[Comm Parameters]

Current:  9600 8N1

  Speed          Parity          Data

A: 300           L: None         S: 5
B: 1200          M: Even         T: 6
C: 2400          N: Odd          U: 7
D: 4800          O: Mark         V: 8
E: 9600          P: Space
F: 19200                      Stopbits
G: 38400                         W: 1
H: 57600                         X: 2
I: 115200        Q: 8-N-1
J: 230400        R: 7-E-1


Choice, or <Enter> to exit?
```

# AT91SAM9263 – moduł diagnostyczny DBGU

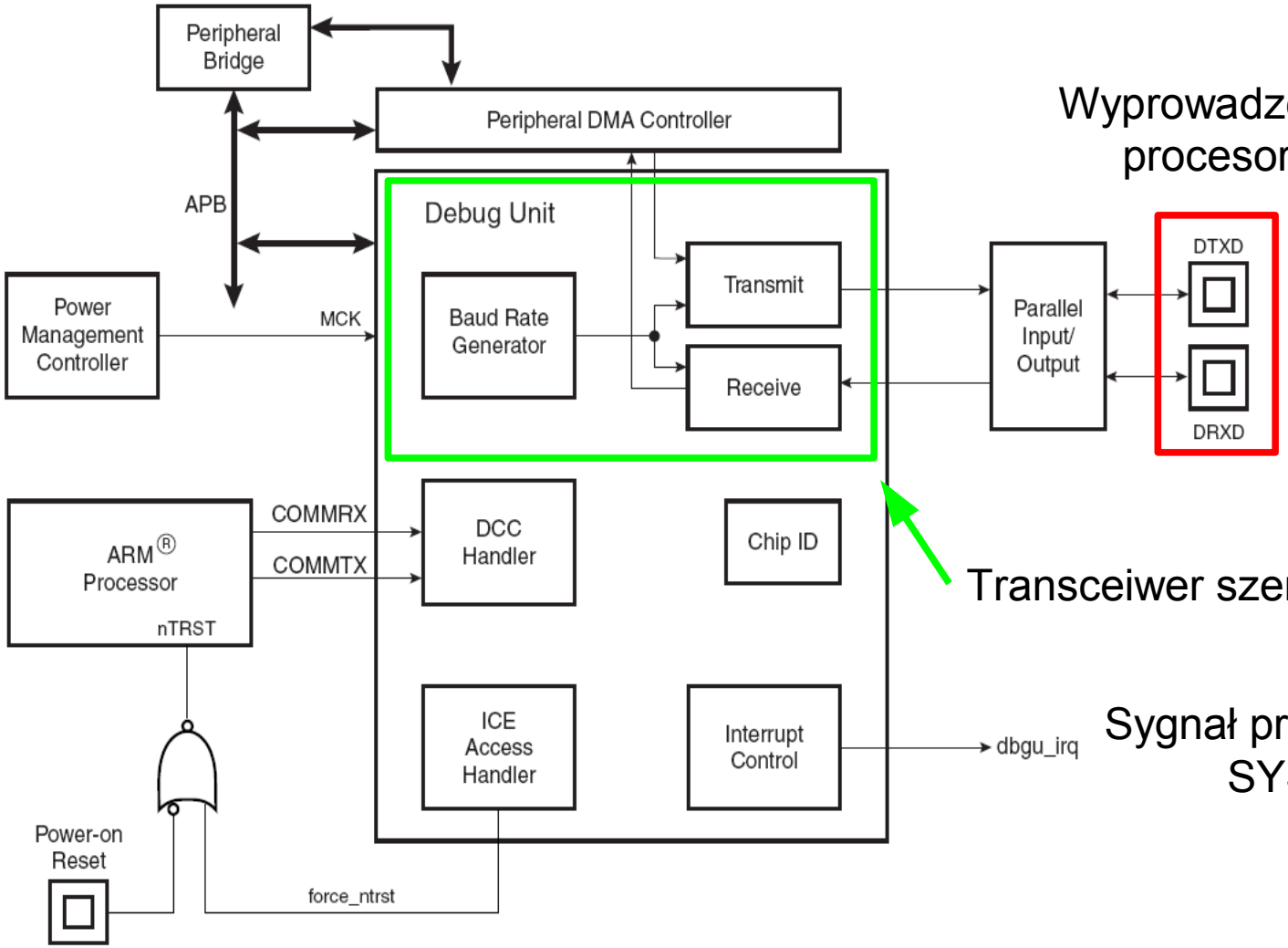
**(rozdział 30)**

# Port szeregowy jako interfejs diagnostyczny

Cechy portu diagnostycznego DBGU (DeBuG Unit):

- Asynchroniczna transmisja danych zgodna ze standardem RS232 (8 bitów danych, jeden bit parzystości z możliwością wyłączenia),
- Możliwość zgłaszania przerwań systemowych współdzielonych (PIT, RTT, WDT,DMA, PMC, RSTC, MC),
- Azaliza poprawności odebranych ramek,
- Sygnalizacja przepełnionego bufora TxD lub RxD,
- Trzy tryby diagnostyczne: zdalny loopback, lokalny loopback oraz echo,
- Maksymalna szybkość transmisji rzędu 1 Mbit/s,
- Możliwość komunikacji z rdzeniem procesora COMMRx/COMMTx.

# Schemat blokowy portu DBGU procesora ARM 9

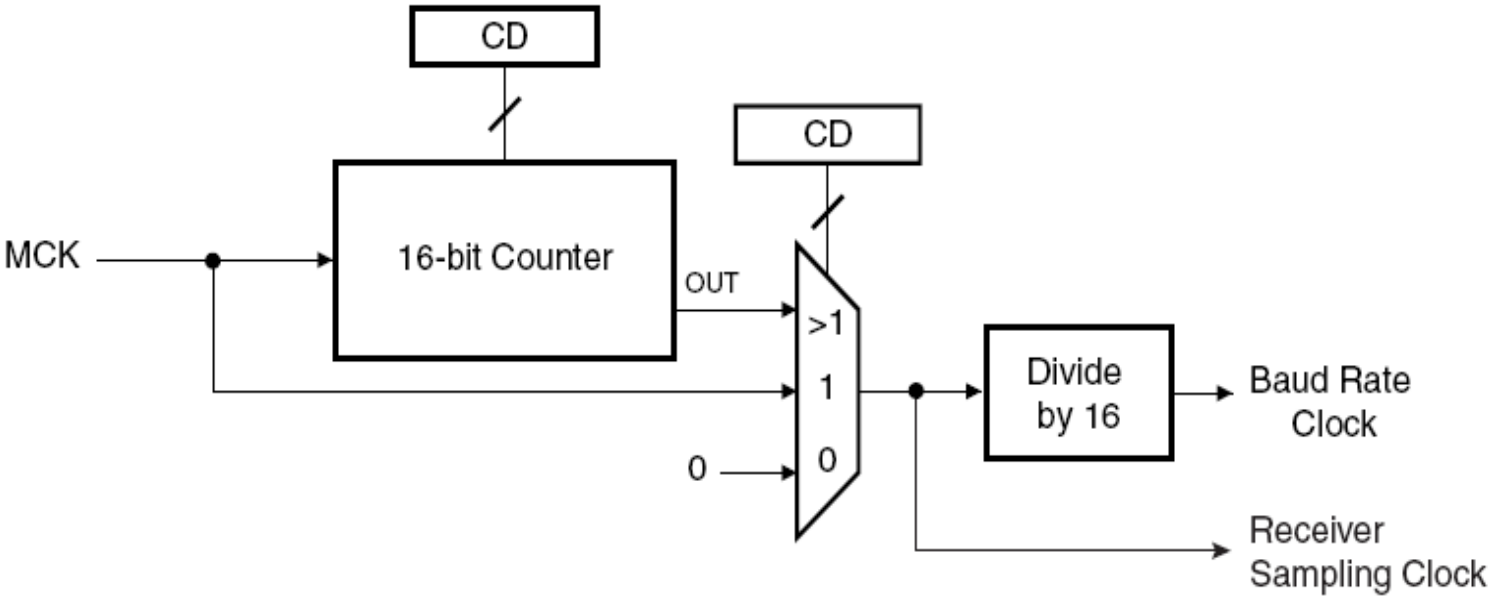



Wyprowadzenia procesora

Transceiwer szeregowy

Sygnał przerwnia SYS

# Szybkość transmisji



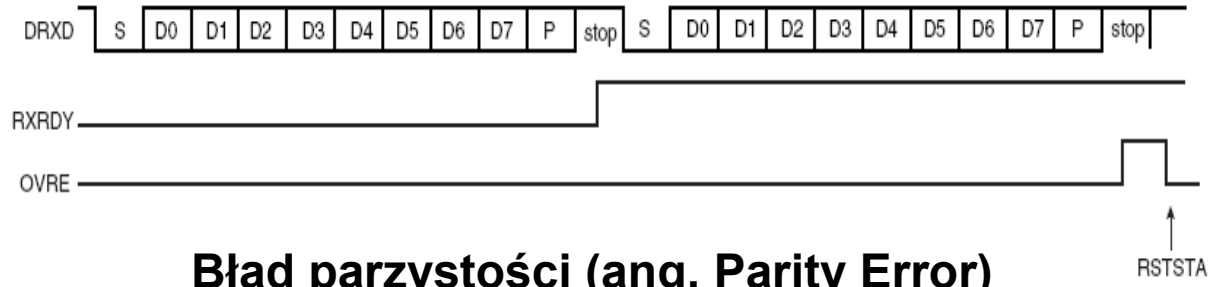

Generator sygnału zegarowego odpowiedzialnego za szybkość transmisji (ang. Baud Rate).

Szybkość transmisji danych wyrażona jest wzorem:

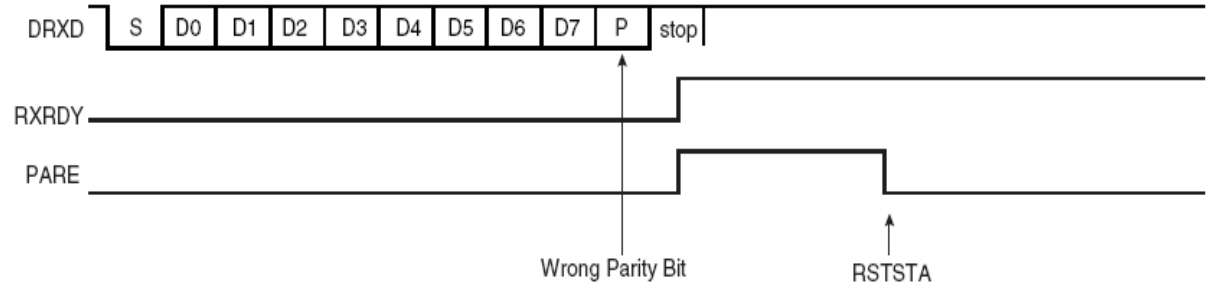
**Baud Rate = MCK / (16 x CD)**, gdzie CD (Clock Divisor) jest polem rejestru DBGU_BRGR

# Błędy podczas transmisji danych

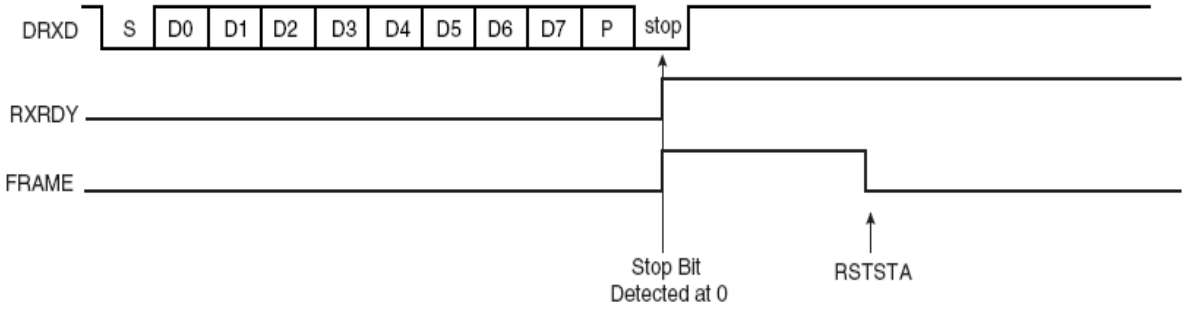
## Przepełnienie bufora odbiorczego BGU_RHR (ang. Buffer Overflow)




## Błąd parzystości (ang. Parity Error)




## Błąd ramki (ang. Frame Error)



# Konfiguracja portu DBGU

**static void Open_DBGU (void)**{

1. Wyłącz przerwania od portu DBGU, rejestr AT91C_BASE_DBGU->DBGU_IDR
2. Resetuj i wyłącz odbiornik AT91C_BASE_DBGU->DBGU_CR
3. Resetuj i wyłącz nadajnik AT91C_BASE_DBGU->DBGU_CR
4. Konfiguracja portów wejścia-wyjścia jako porty RxD i TxD DBGU,
   rejestry AT91C_BASE_PIOC->PIO_ASR oraz AT91C_BASE_PIOC->PIO_PDR
5. Konfiguracja szybkości transmisji portu szeregowego AT91C_BASE_DBGU->DBGU_BRGR
6. Konfiguracja trybu pracy, tryb normalny bez przystości (8N1),

   rejestr AT91C_BASE_DBGU->DBGU_MR, flagi AT91C_US_CHMODE_NORMAL,
   AT91C_US_PAR_NONE;
7. Skonfiguruj przerwania jeżeli są wykorzystywane: Open_DBGU_INT()
8. Włącz odbiornik, rejestr AT91C_BASE_DBGU->DBGU_CR
9. Włącz nadajnik, rejestr AT91C_BASE_DBGU->DBGU_CR

}

# Odczyt i zapis danych do portu DBGU

```
void dbgu_print_ascii (const char *Buffer)
{
   while ( data_are_in_buffer ) {
        while ( …TXRDY... ){};          /* wait intil Tx buffer busy – check TXRDY flag */
        DBGU_THR = ...                  /* write a single char to Transmitter Holding Register  */
         }
}

void dbgu_read_ascii (const char *Buffer, unsigned int Size){
        do {
        While ( ...RXRDY... ){};        /* wait until data available */
        Buffer[...] = DBGU_RHR;         /* read data from Receiver Holding Register */
        } while ( …read_enough_data... )
}
```

# Rejestr statusowy

**30.5.6 Debug Unit Status Register**
**Name:** DBGU_SR

**Address:** 0xFFFFEE14

**Access Type:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| COMMRX | COMMTX | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | RXBUFF | TXBUFE | – | TXEMPTY | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PARE | FRAME | OVRE | ENDTX | ENDRX | – | TXRDY | RXRDY |

- **RXRDY: Receiver Ready**

0 = No character has been received since the last read of the DBGU_RHR or the receiver is disabled.

1 = At least one complete character has been received, transferred to DBGU_RHR and not yet read.

- **TXRDY: Transmitter Ready**

0 = A character has been written to DBGU_THR and not yet transferred to the Shift Register, or the transmitter is disabled.

1 = There is no character written to DBGU_THR not yet transferred to the Shift Register.

- **ENDRX: End of Receiver Transfer**

0 = The End of Transfer signal from the receiver Peripheral Data Controller channel is inactive.

1 = The End of Transfer signal from the receiver Peripheral Data Controller channel is active.

- **ENDTX: End of Transmitter Transfer**

0 = The End of Transfer signal from the transmitter Peripheral Data Controller channel is inactive.

1 = The End of Transfer signal from the transmitter Peripheral Data Controller channel is active.

- **OVRE: Overrun Error**

0 = No overrun error has occurred since the last RSTSTA.

1 = At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0 = No framing error has occurred since the last RSTSTA.

1 = At least one framing error has occurred since the last RSTSTA.

- **PARE: Parity Error**

0 = No parity error has occurred since the last RSTSTA.

1 = At least one parity error has occurred since the last RSTSTA.

- **TXEMPTY: Transmitter Empty**

0 = There are characters in DBGU_THR, or characters being processed by the transmitter, or the transmitter is disabled.

1 = There are no characters in DBGU_THR and there are no characters being processed by the transmitter.

# AT91SAM9263 – USART

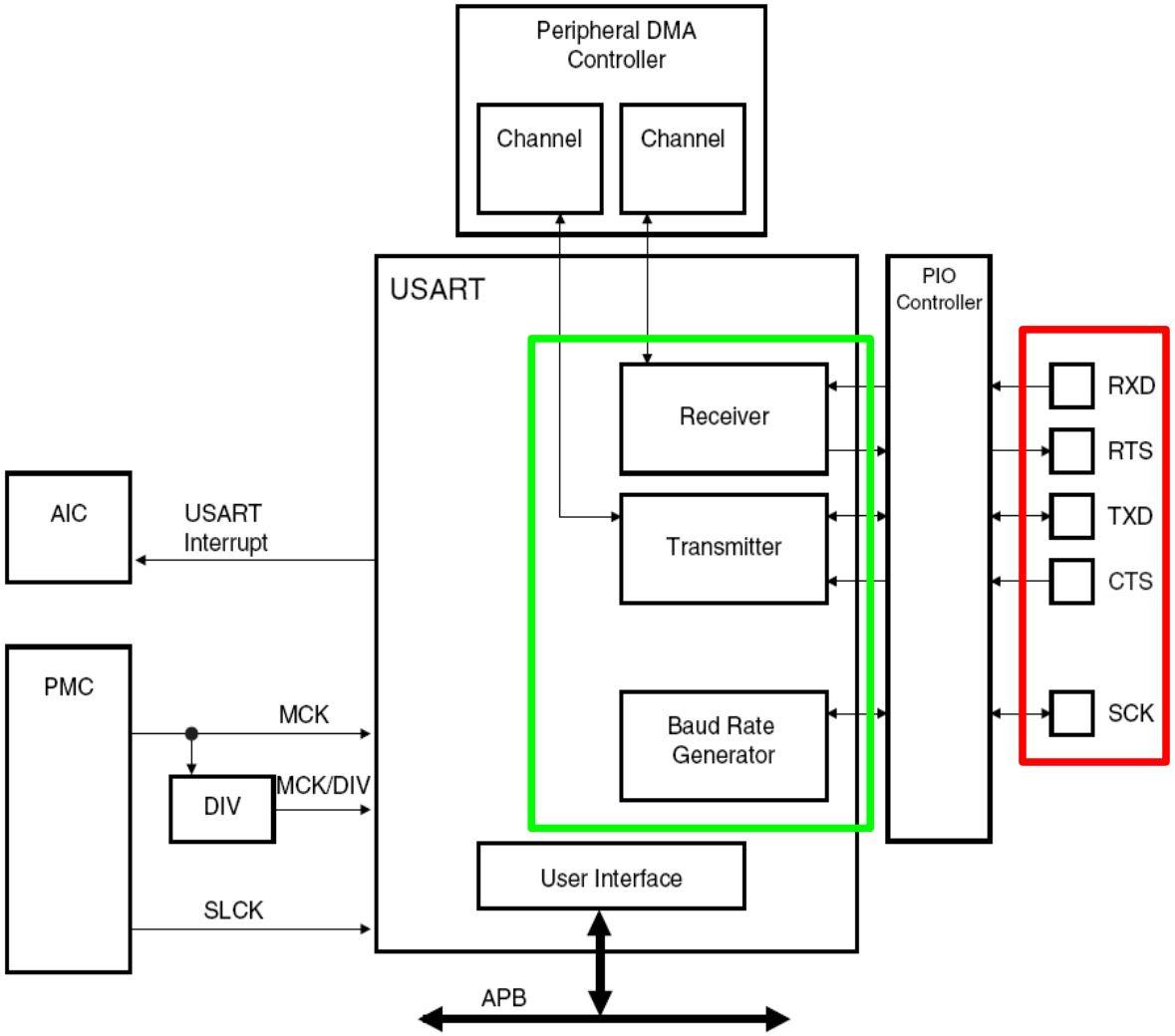
(rozdział 34)

# Port szeregowy USART

## Cechy portu USART (Universal Synch. Asynch. Receiver-Transmitter):

- Asynchroniczna lub transmisja danych,
- Programowalna długość ramki, kontrola parzystości, liczba bitów stopu,
- Możliwość zgłaszania przerwań systemowych współdzielonych (PIT, RTT, WDT,DMA, PMC, RSTC, MC),
- Analiza poprawności odebranych ramek,
- Sygnalizacja przepełnionego bufora TxD lub RxD,
- Możliwość odbierania ramek o zmiennej długości – wykorzystanie dodatkowego licznika do odmierzania czasu,
- Trzy tryby diagnostyczne: zdalny loopback, lokalny loopback oraz echo,
- Maksymalna szybkość transmisji rzędu 1 Mbit/s,
- Wsparcie sprzętowej kontroli przepływu danych,
- Możliwość transmisji w systemie Multidrop, transmisja danej i adresu,
- Możliwość transmisji danych z wykorzystaniem kanału DMA (Direct Memory Access),
- Wsparcie dla standardu transmisji różnicowej RS485 oraz systemów pracujących w zakresie podczerwieni (wbudowany modulator-demodulator IrDA).

# Schemat blokowy transceivera USART



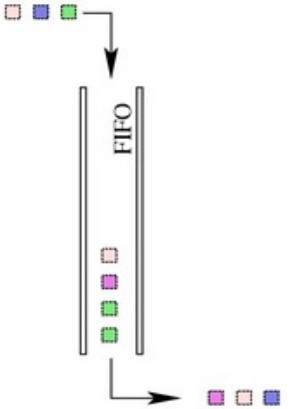
# Struktury danych

# Stack and FIFO (1)

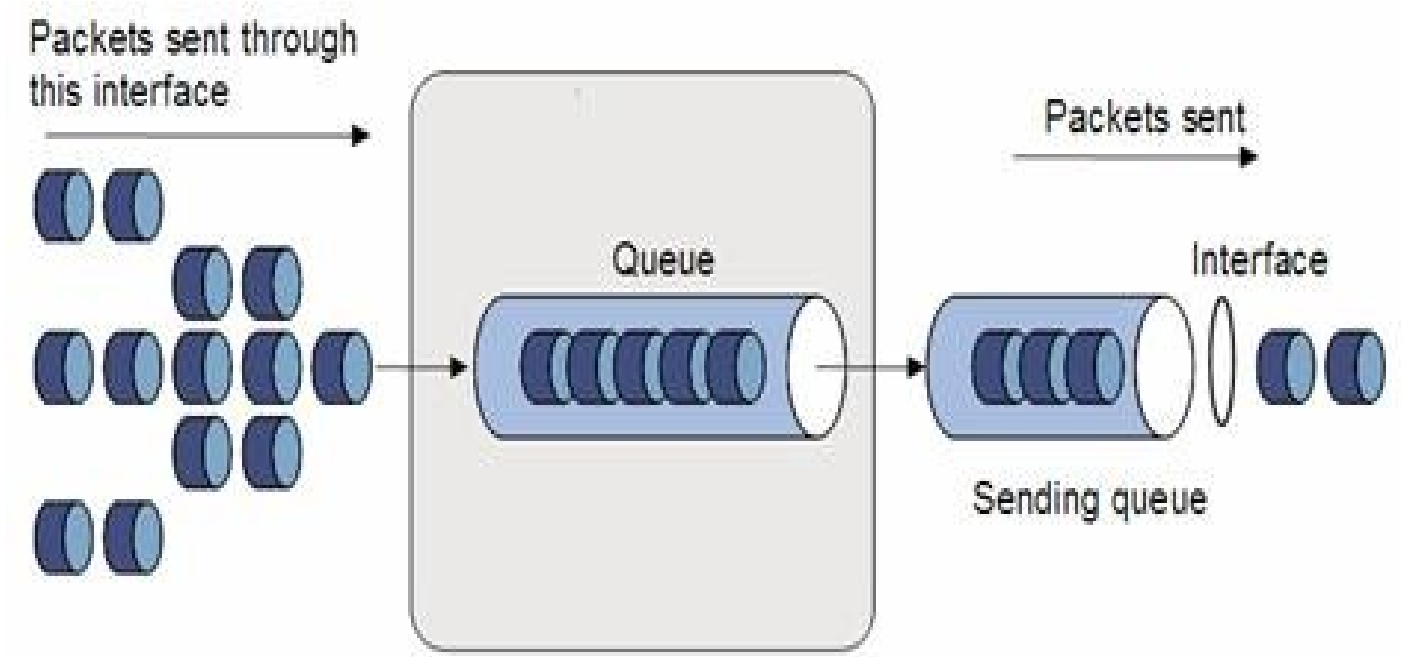
**Stos (ang. stack lub LIFO Last-In, First-Out) -** liniowa struktura danych, w której dane odkładane są na wierzch stosu i z wierzchołka stosu są zdejmowane. Ideę stosu danych można zilustrować jako stos położonych jedna na drugiej książek – nowy egzemplarz kładzie się na wierzch stosu i z wierzchu stosu zdejmuje się kolejne egzemplarze. Elementy stosu poniżej wierzchołka stosu można wyłącznie obejrzeć, aby je ściągnąć, trzeba najpierw po kolei ściągnąć to, co jest nad nimi

**FIFO (ang. First In, First Out) -** przeciwieństwem stosu LIFO jest kolejka, bufor typu FIFO (pierwszy na wejściu, pierwszy na wyjściu), w którym dane obsługiwane są w takiej kolejności, w jakiej zostały dostarczone (jak w kolejce do kasy)

First–in First–out (FIFO)



# Kolejka FIFO (1)

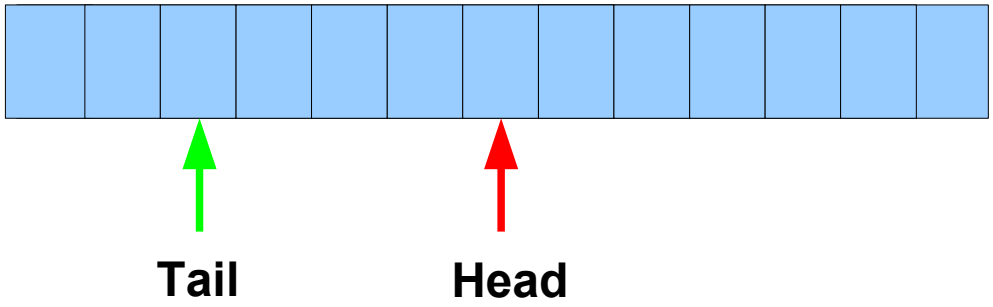



- Dane do kolejki FIFO mogą być wpisywane przez kilka niezależnych aplikacji, wątków lub urządzeń. W takiej sytuacji dostęp do kolejki kontrolowany jest przez Semafor (zmienna globalna).
- Dane zgomadzone w kolejce wysyłane są w kolejności w jakiej zostały wpisane.

# Kolejka FIFO (2)

**Dane w kolejce FIFO**

**Adres w pamięci:** 0xffD50 0xffD50 + size -1




## Zapis danej do kolejki FIFO:

- Zwiększ wskaźnik Head o jeden, zapisz daną.

## Odczyt danej z kolejki FIFO:

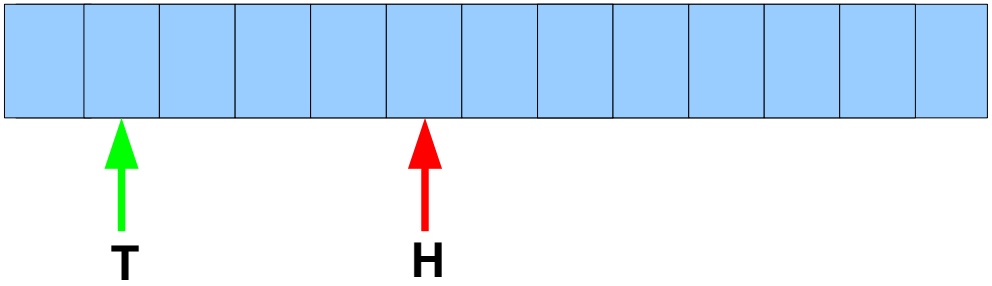
- Odczytaj daną, zwiększ wskaźnik Tail o 1.

W przypadku, gdy Tail lub Head wskazuje na ostatni dostępny elemet kolejki zamiast inkrementacji wskaźnik jest zerowany. Pozwala to na płynne przesuwanie wskaźników – bufor kołowy (ang. circular buffer).

# Kolejka FIFO (3)

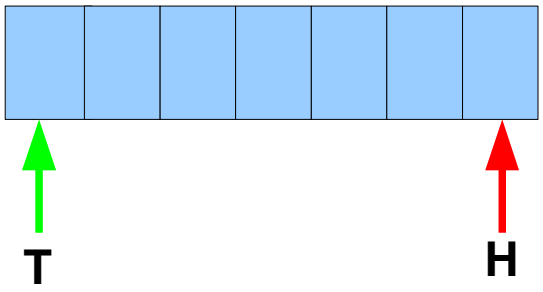
**Kolejka pusta T = H**

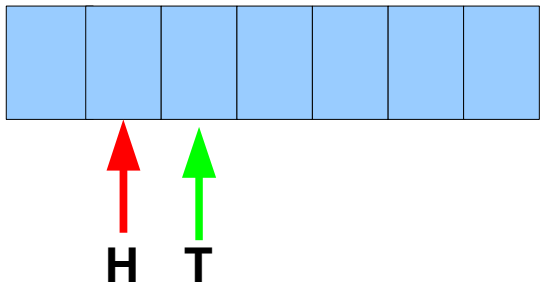

**Dane w kolejce, ilość danych = H – T**




**Brak miejsca w kolejce**

(T = 0) & (H = Size) lub T – H = 1



# Kolejka FIFO – implementacja w C (1)

```
#define BUFFERSIZE   0xFF                    /* FIFO buffer size and mask */
typedef struct FIFO {
    char buffer [BUFFERSIZE+1];
    unsigned int head;
    unsigned int tail;
};
void FIFO_Init (struct FIFO *Fifo);
void FIFO_Empty (struct FIFO *Fifo);
int FIFO_Put (struct FIFO *Fifo, char Data);
int FIFO_Get (struct FIFO *Fifo, char *Data)


void FIFO_Init (struct FIFO *Fifo){
        Fifo->head=0;
        Fifo->tail=0;

        /* optional: initialize data in buffer with 0 */
}
```

# Kolejka FIFO – implementacja w C (2)

```
void FIFO_Empty (struct FIFO *Fifo){
        Fifo->head = Fifo->tail;                                    /* now FIFO is empty*/
}
int FIFO_Put (struct FIFO *Fifo, char Data){
        if ((Fifo->tail-Fifo->head)==1 || (Fifo->tail-Fifo->head)==BUFFERSIZE)){
        return -1; };                                               /* FIFO overflow */
        Fifo->buffer[Fifo->head] = Data;
        Fifo->head = (Fifo->head + 1) & BUFFERSIZE;
        return 1;                                                   /* Put 1 byte successfully */
}
int FIFO_Get (struct FIFO *Fifo, char *Data){
        If ((TxFifo.head!=TxFifo.tail)){
        *Data = Fifo->buffer[Fifo->tail];
        Fifo->tail = (Fifo->tail + 1) &BUFFERSIZE;
        return 1;                                                   /* Get 1 byte successfully */
        } else return -1;                                           /* No data in FIFO */
}
```

# Kolejka FIFO – pułapka

```
void FIFO_Empty (struct FIFO *Fifo){
        Fifo->head = Fifo->tail;                                    /* now FIFO is empty*/
}
int FIFO_Put (struct FIFO *Fifo, char Data){
        if ((Fifo->tail-Fifo->head)==1 || (Fifo->tail-Fifo->head)==BUFFERSIZE)){
        return -1; };                                               /* FIFO overflow */
        Fifo->buffer[Fifo->head++] = Data;
        Fifo->head = Fifo->head & BUFFERSIZE;                       /* be carefull with interrupts */
        return 1;                                                   /* Put 1 byte successfully */
}
int FIFO_Get (struct FIFO *Fifo, char *Data){
        If ((TxFifo.head!=TxFifo.tail)){
        *Data = Fifo->buffer[Fifo->tail++];
        Fifo->tail &= BUFFERSIZE;                                   /* be carefull with interrupts */
        return 1;                                                   /* Get 1 byte successfully */
        } else return -1;                                           /* No data in FIFO */
}
```

# Schematy elektryczne

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

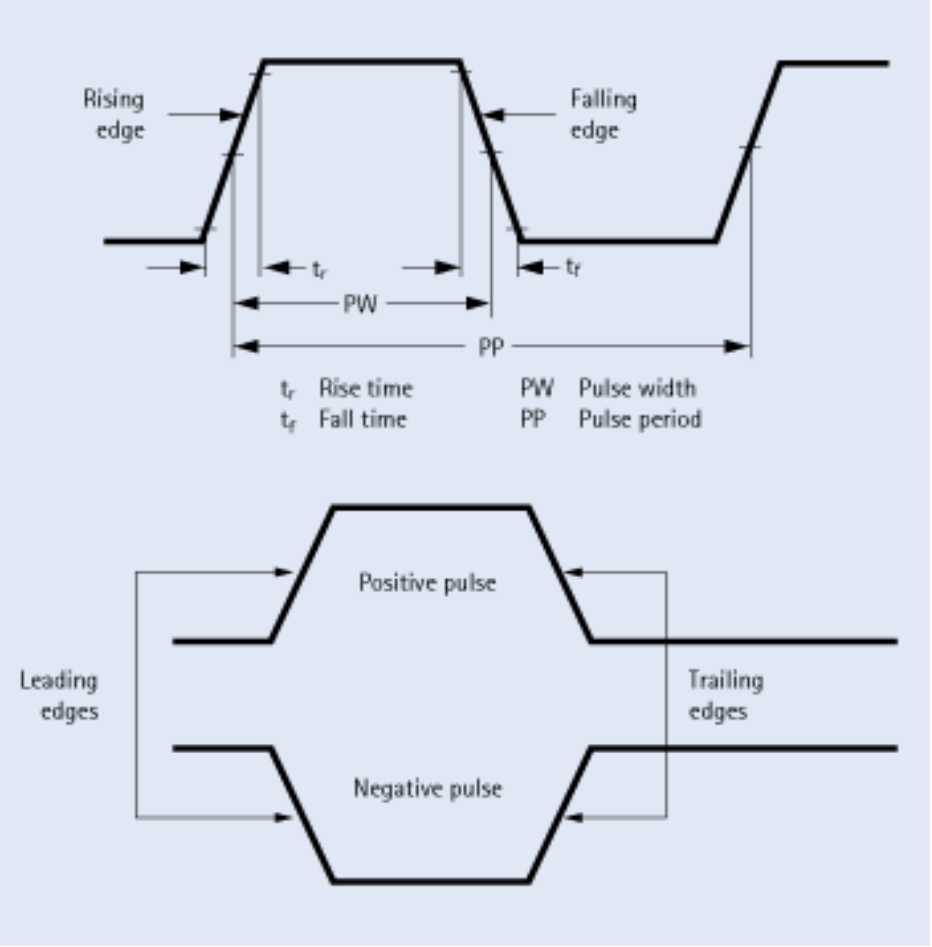
# Sygnały cyfrowe

**Sygnały cyfrowe określają dwa parametry:**

- f – częstotliwość (okres),
- A – amplituda.

**Układy cyfrowe mogą być wyzwalane:**

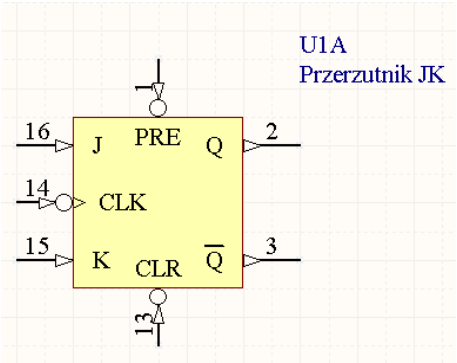
- Zmianą poziomu (większy lub mniejszy poziom od poziomu odniesienia),
- Zboczem sygnału (zmiana poziomu sygnału z '0' na '1' lub z '1' na '0').

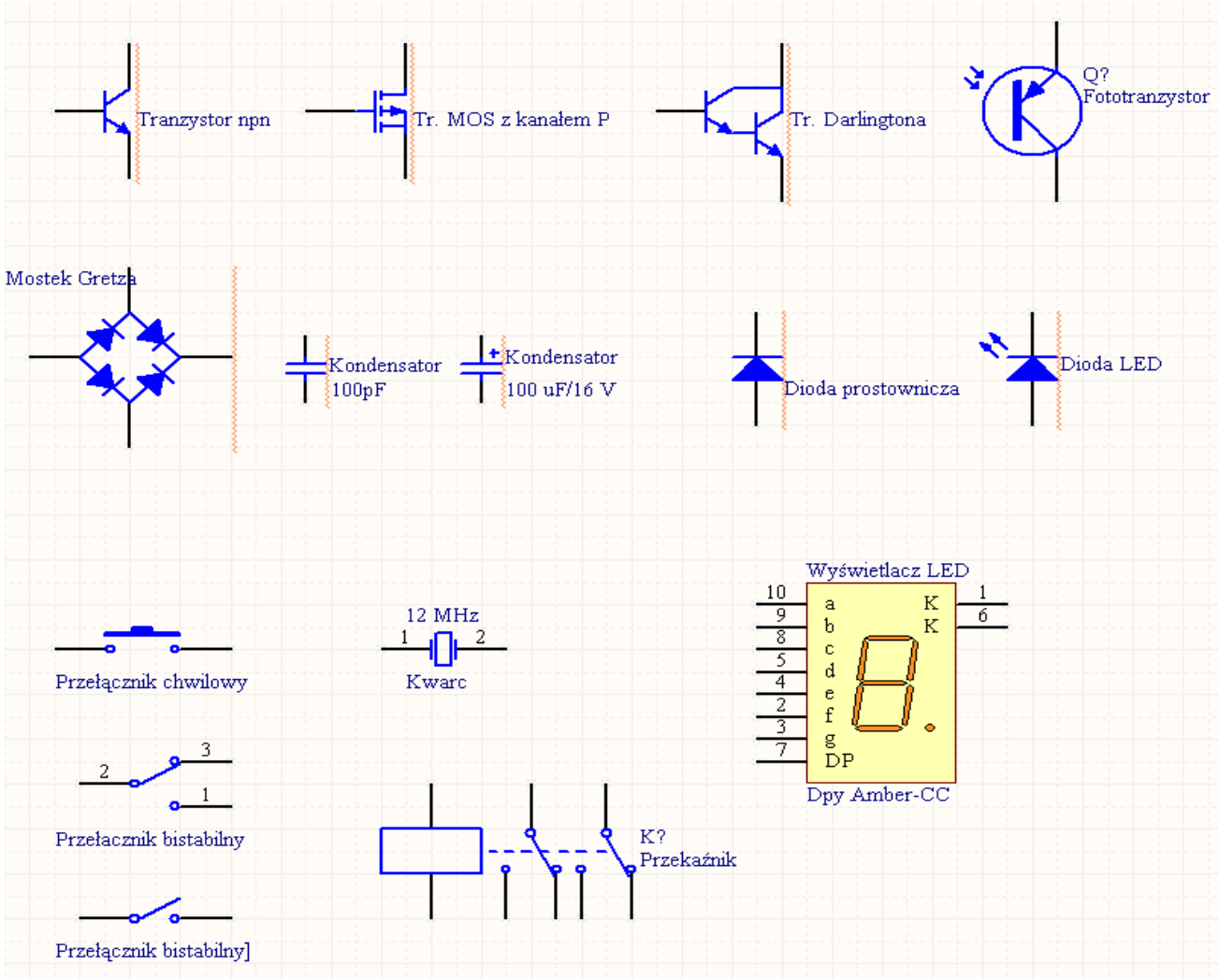


# Schematy elektryczne (1)

## Symbole zasilania

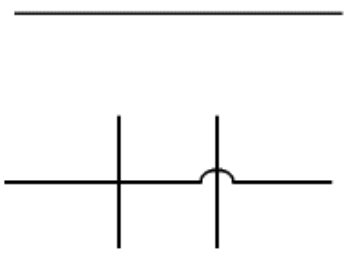

## Symbole masy

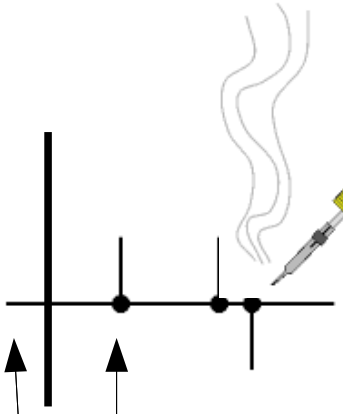



# Schematy elektryczne (2)

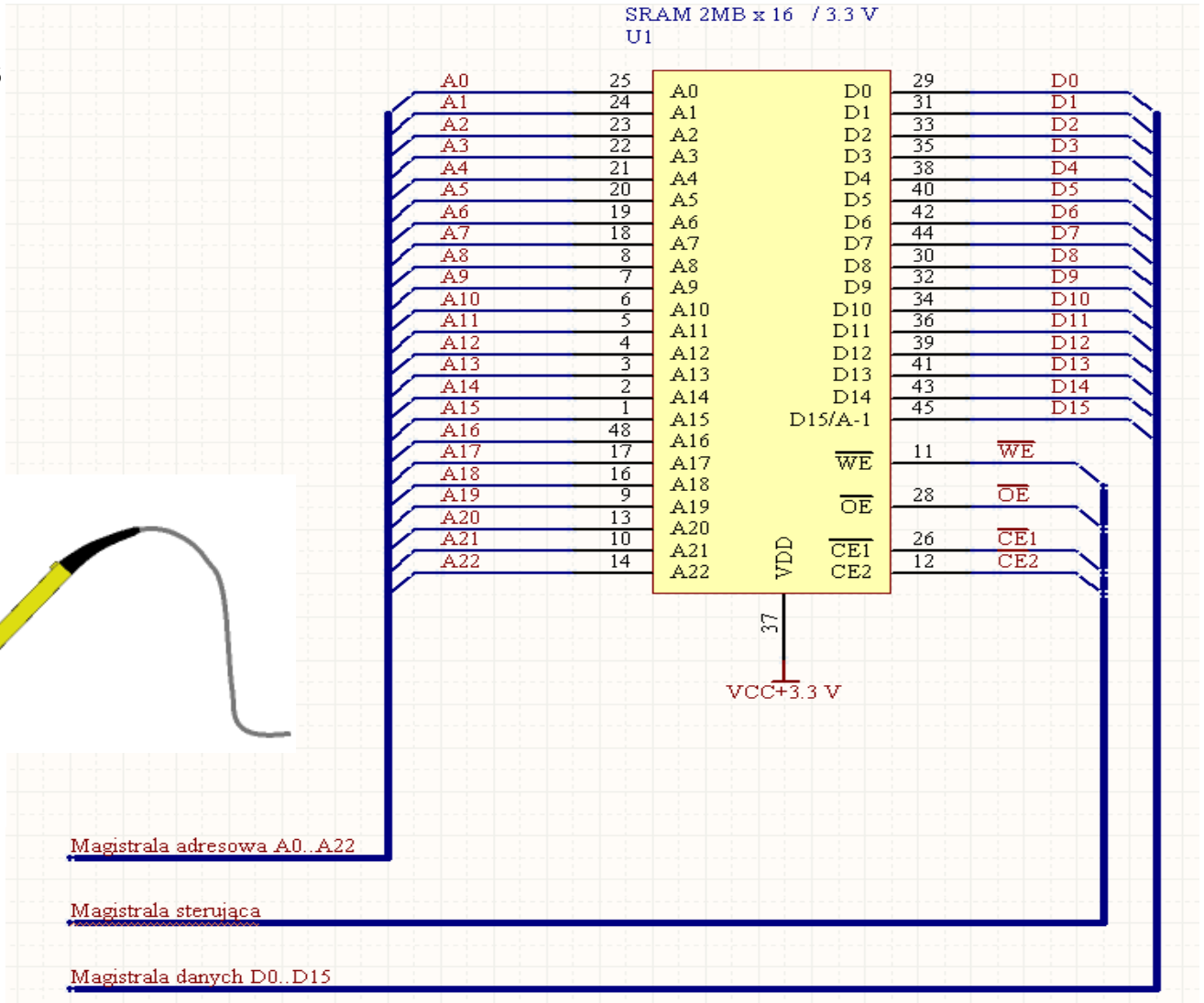
Electrical connections



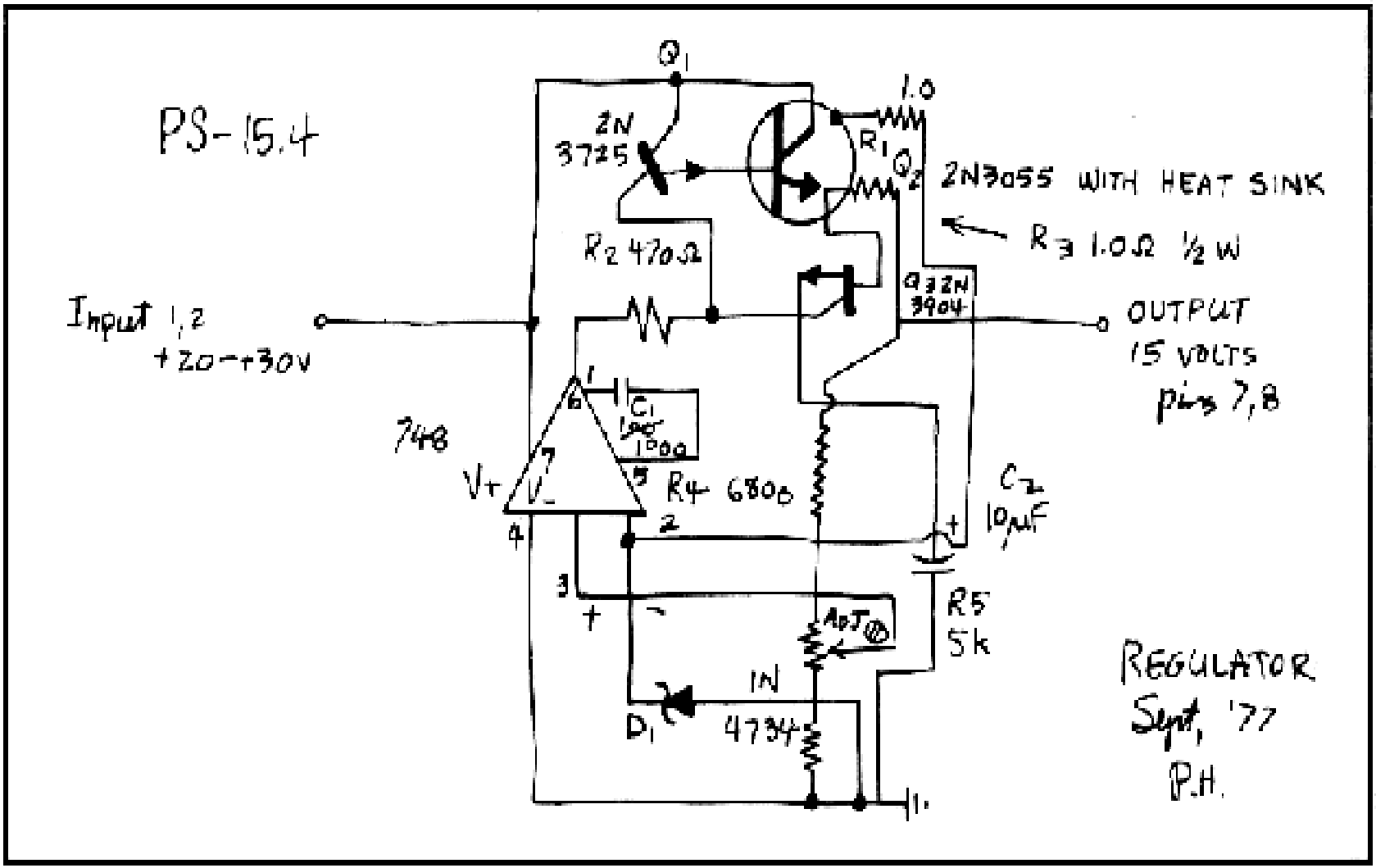


Connection

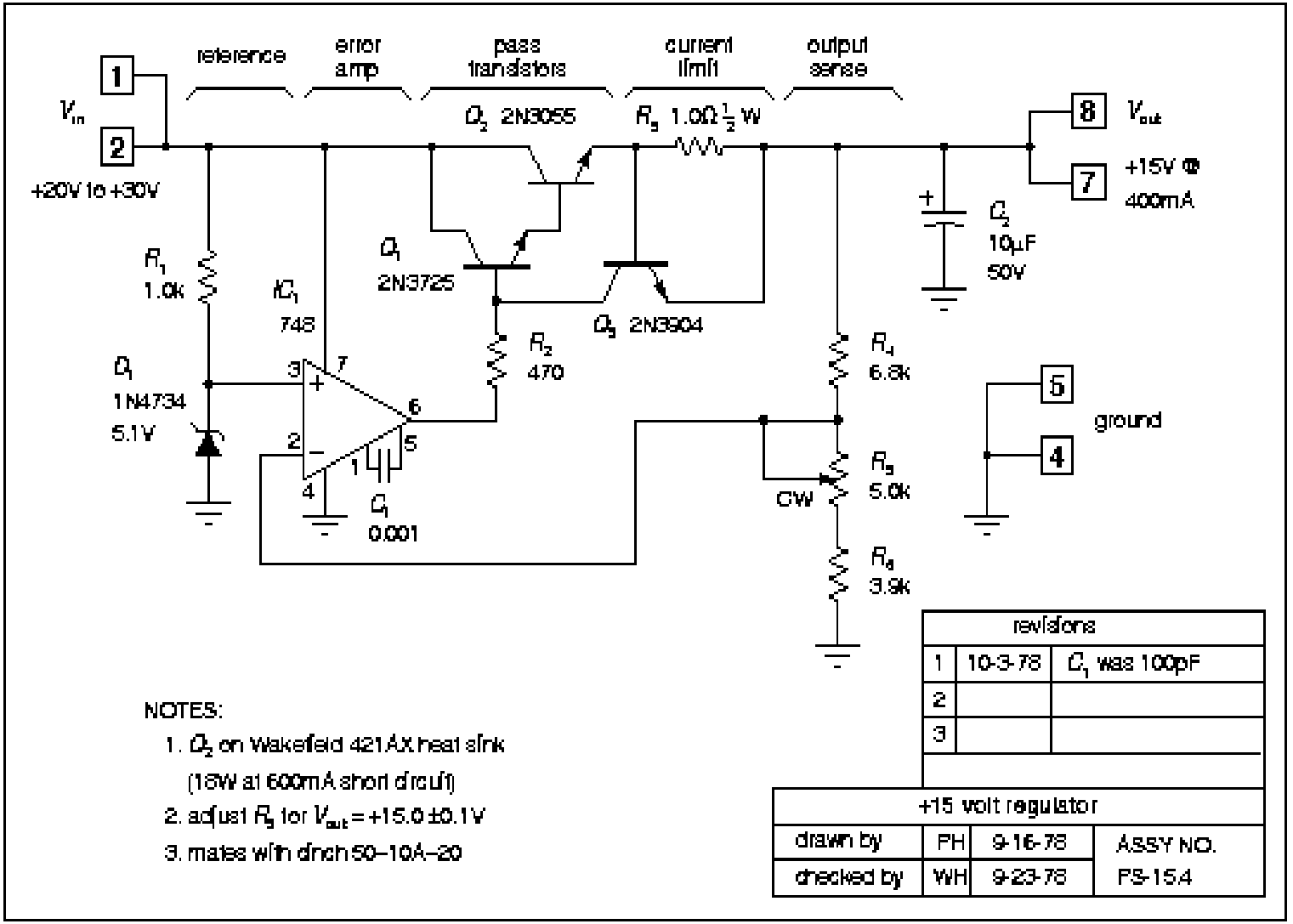
No connection



# Schematy elektryczne – jak tego nie robić ?



# Schematy elektryczne – lepszy sposób



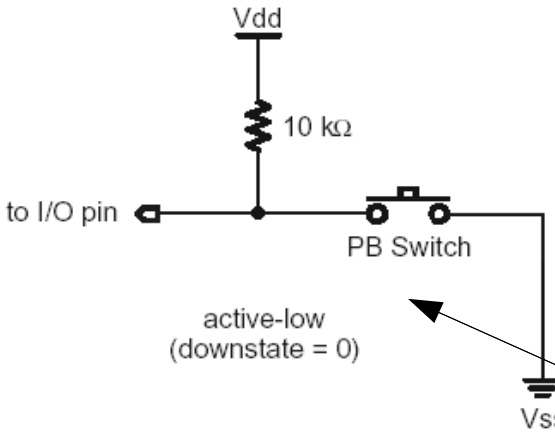

NOTES:
1. $Q_2$ on Wakefield 421AX heat sink (16W at 600mA short circuit)
2. adjust $R_5$ for $V_{out}$ = +15.0 ±0.1V
3. mates with dinch 50–10A–20

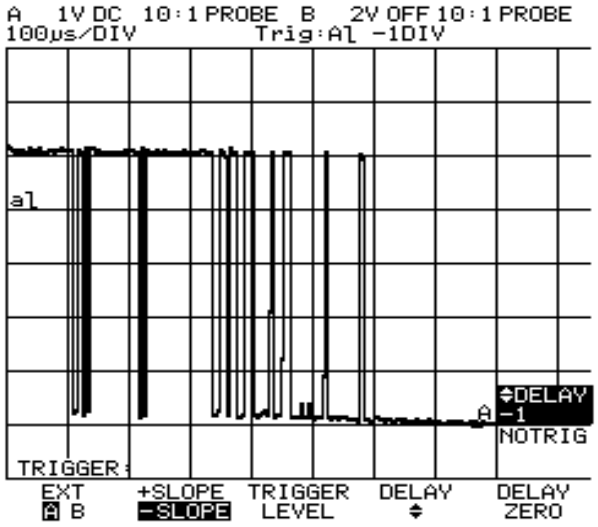
| revisions | | |
|---|---|---|
| 1 | 10-3-78 | $C_1$ was 100pF |
| 2 | | |
| 3 | | |

| +15 volt regulator | | | |
|---|---|---|---|
| drawn by | PH | 9-16-78 | ASSY NO. |
| checked by | WH | 9-23-78 | PS-154 |

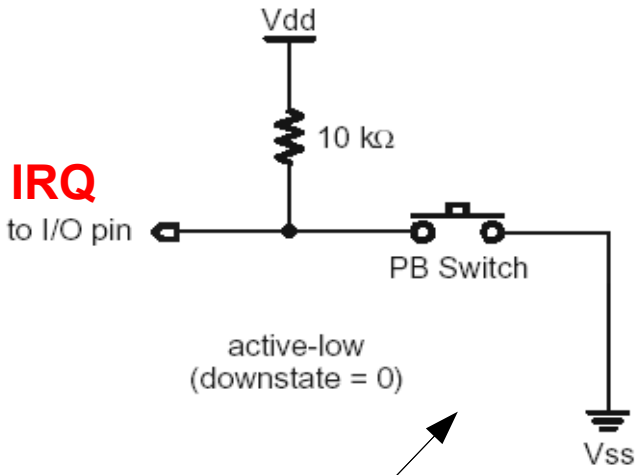
# Odczyt stanu przełącznika

## Polling loop



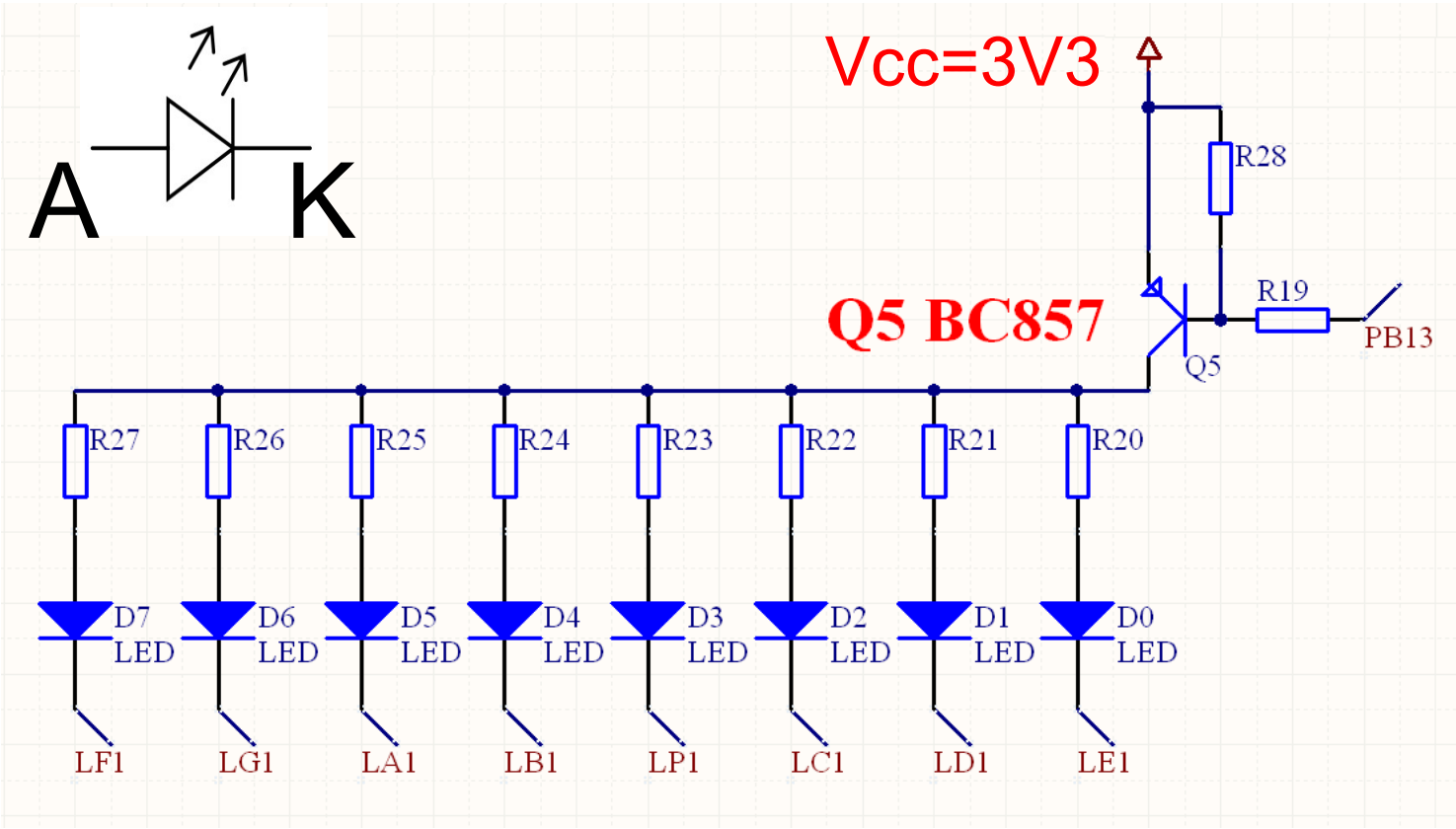

## Interrupt




**Sygnał asynchroniczny**

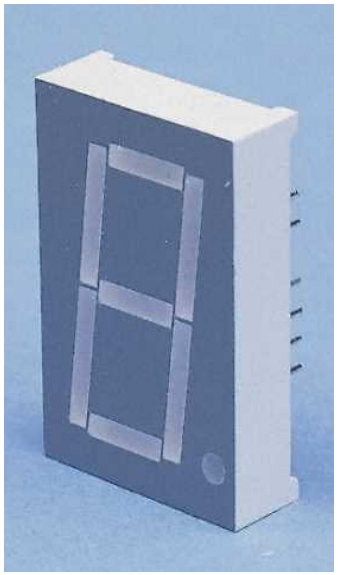
# Dioda elektroluminescencyjna

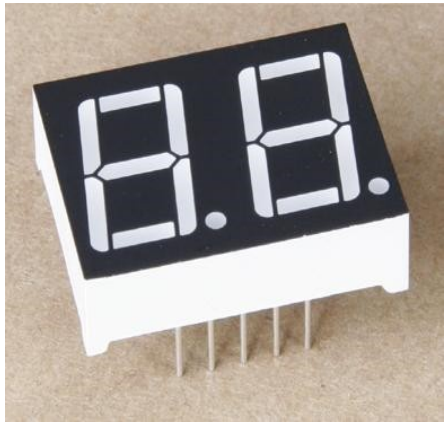
- Dioda elektroluminescencyjna, dioda świecąca LED (ang. light-emitting diode) – dioda półprzewodnikowa, emitujących promieniowanie w zakresie światła widzialnego, podczerwieni lub ultrafioletu.



# Wyświetlacz siedmiosegmentowy LED

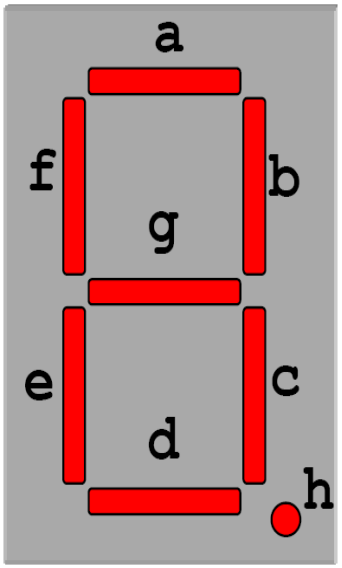
- Wyświetlacz siedmiosegmentowy jest wyświetlaczem zbudowanym z siedmiu segmentów pozwalających na wyświetlanie znaków dziesiętnych 0-9 oraz pozostałych znaku szesnastkowego (A-F). Wyświetlacze siedmiosegmentowe LED posiadają również dodatkowy, ósmy segment w postaci kropki dziesiętnej (H, dp).

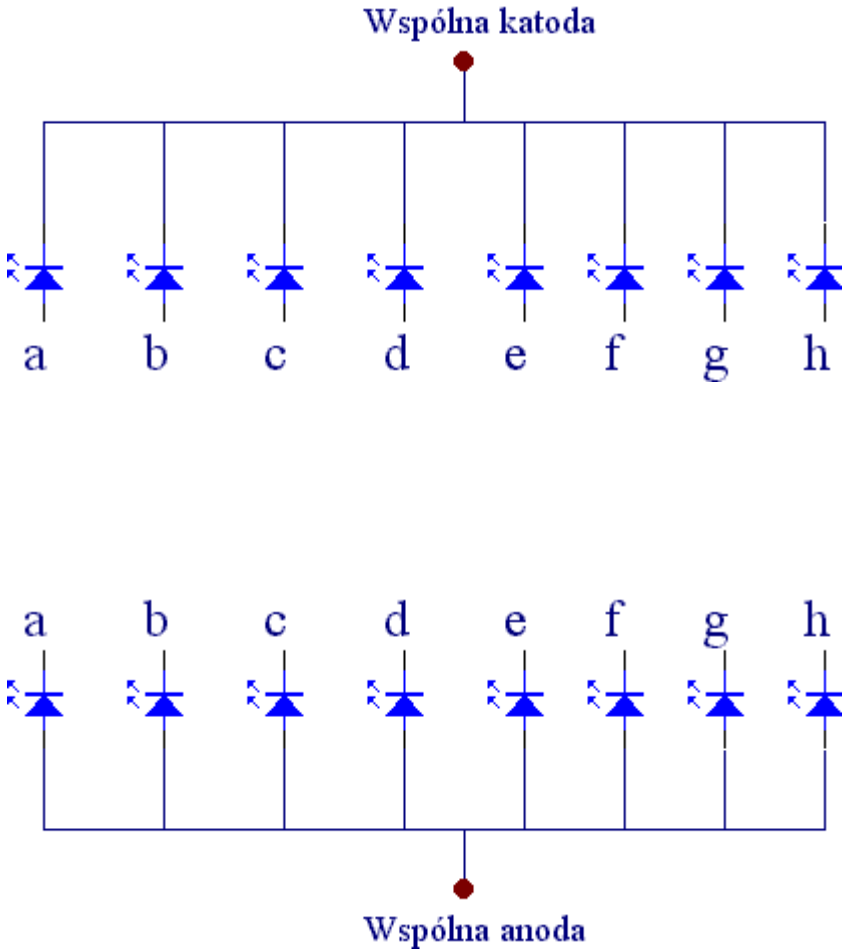





# Wyświetlacz siedmiosegmentowy LED

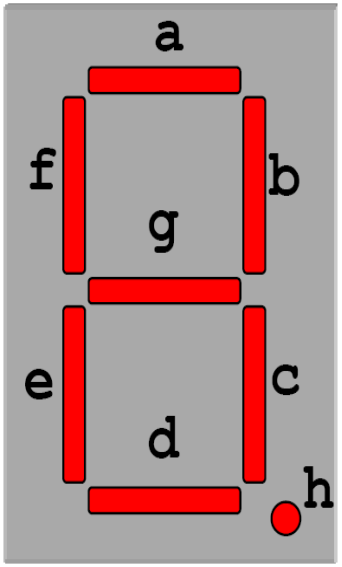
- Wyświetlacze dzielimy na układy:
  - Ze wspólną anodą,
  - Ze wspólną katodą.

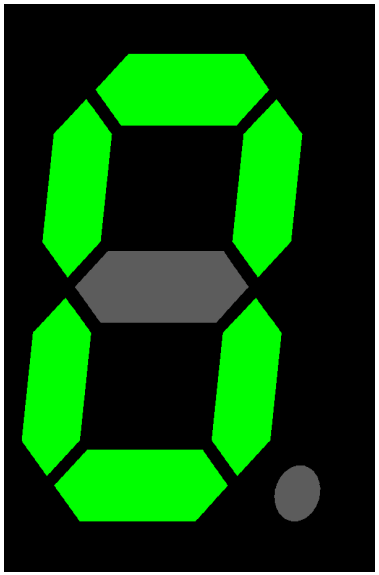


# Wyświetlacz siedmiosegmentowy – dekoder

- Tablica realizujące dekoder kodu BCD (binarny kod dziesiętny) na kod wyświetlacza 7-segmentowego (wspólna katoda).

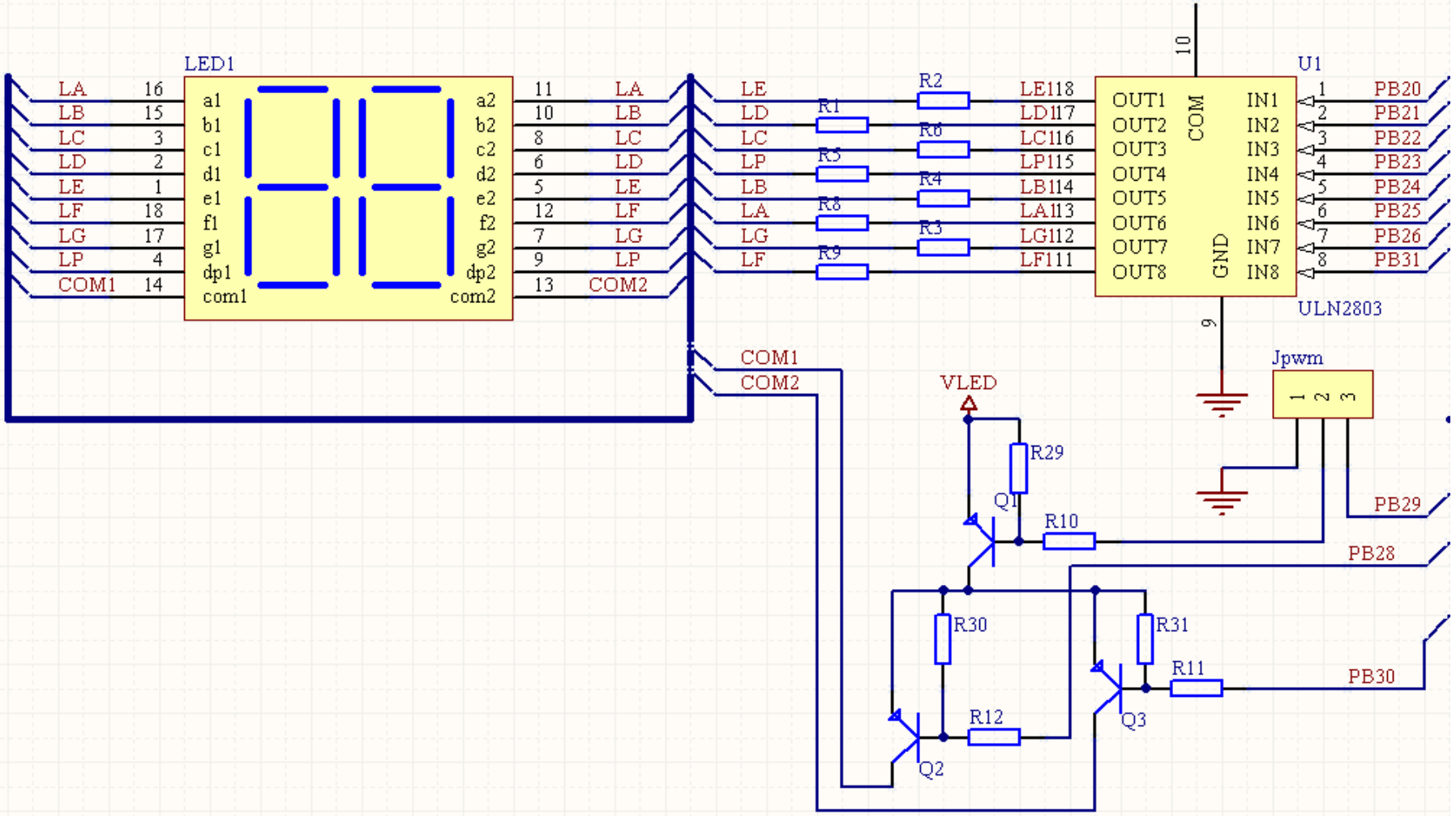


| Cyfra | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 4 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 5 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

Jak wygląda tablica realizująca dekoder BCD na kod szesnastkowy?

# Płytka z wyświetlaczem LED



# Enkoder obrotowy

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

# Enkoder obrotowy

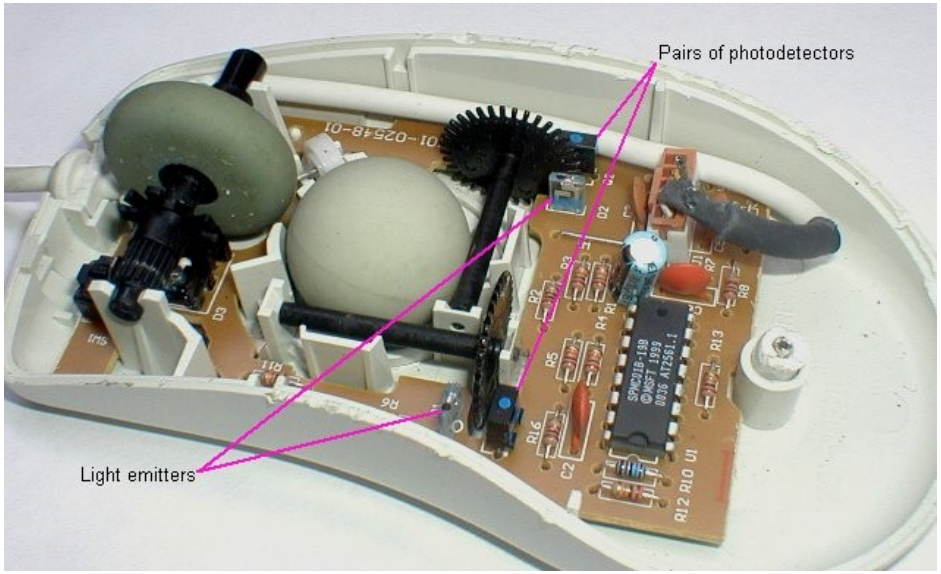
- Enkoder obrotowy (ang. rotary lub shaft encoder) jest elektromechanicznym urządzeniem wykorzystywanym do konwersji położenia kątowego do sygnału analogowego lub cyfrowego. Enkodery obrotowe wykorzystywane są w różnych aplikacjach wymagających precyzyjnego określenia położenia kątowego lub kierunku obrotu,np. systemy sterowania, robotyka, specjalizowane obiektywy fotograficzne, urządzenia wejściowe do urządzeń komputerowych (myszki, tackball-e) oraz urządzenia radarowe.
- Wyróżniamy dwa rodzaje enkoderów podające położenie:
  - Bezwzględne,
  - Względne (przyrostowe).
- Enkodery bezwzględne generują unikalny kod cyfrowy dla różnych kątów obrotu.
- Enkodery przyrostowe, znane jako kwadraturowe, posiadają zwykle dwa wyprowadzenia. Do określenia zmiany kąta położenia wykorzystuje się różnicę faz wygenerowanych sygnałów.
- Enkodery mogą być mechaniczne lub optyczne. Optyczne enkodery wykorzystują cyfrowe mozaiki oparte na kodzie Graya.

# Przykłady enkoderów

Precyzyjny pomiar przesunięcia, kąta obrotu

Potencjometry cyfrowe

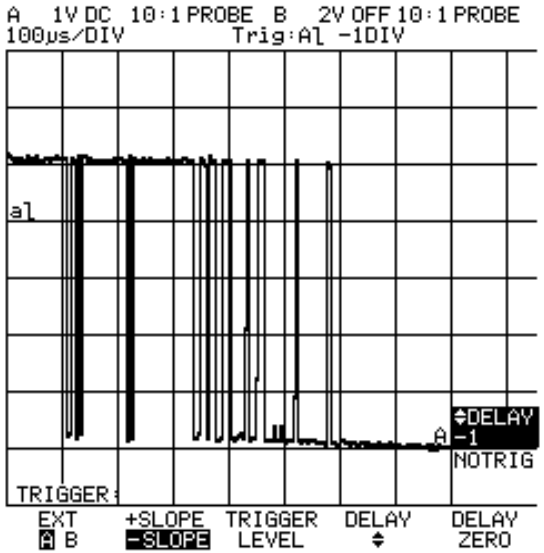
Myszka komputerowa i enkodery optyczne

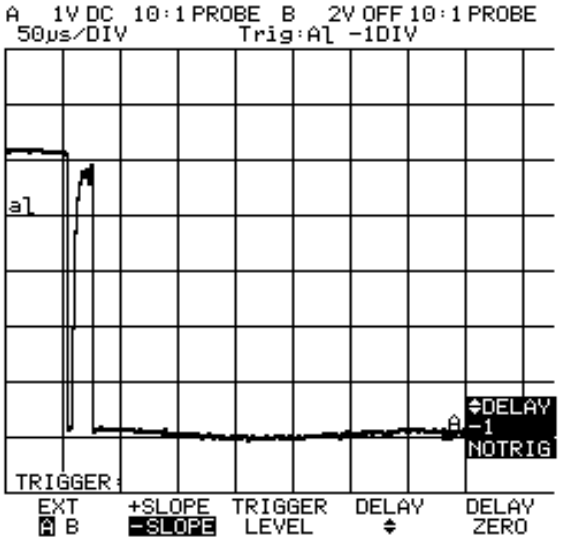



# Enkodery kwadraturowe (1)

- Enkodery mechaniczne wymagają filtracji drgań styków mechanicznych. Ze względu na niską cenę wykorzystywane są jako potencjometry cyfrowe w sprzęcie audio (potencjometr regulujący głośność, sterowanie menu). Ze względu na drgania styków enkodery mechaniczne mają stosunkowo niską szybkość obrotową.

Drgania styków enkodera mechanicznego



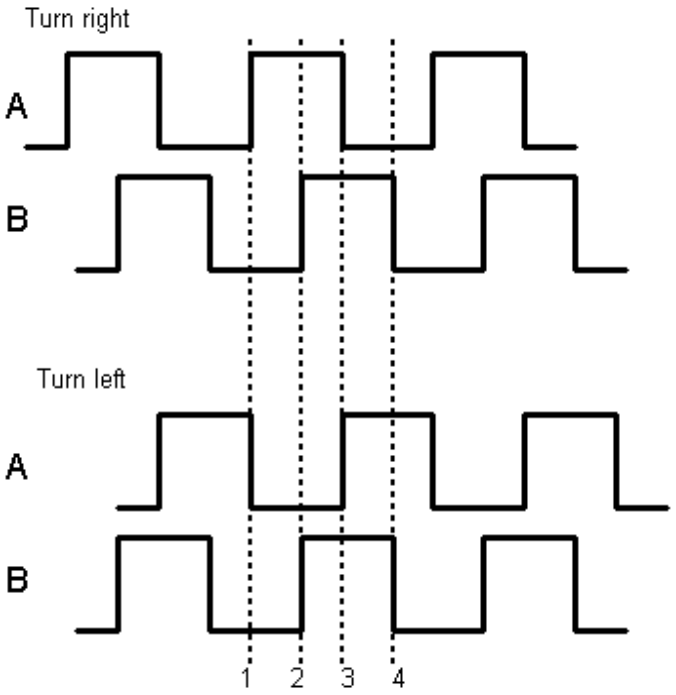



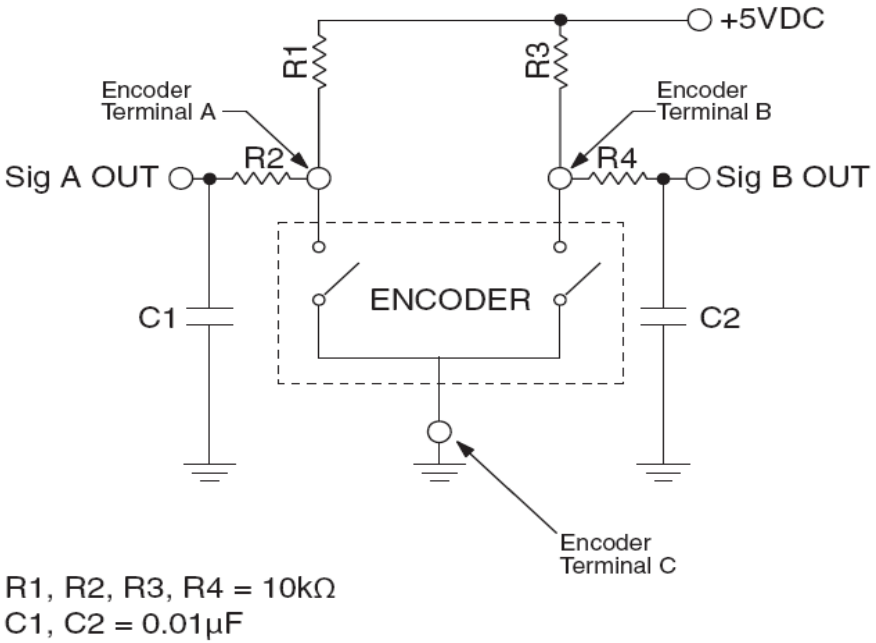

**Electrical**
MAXIMUM VOLTAGE: 5 VDC.
MAXIMUM CURRENT DC: 0,5 mA
DIELECTRIC STRENGTH: 50 VAC.
INSULATION RESISTANCE: 10 MΩ.
BOUNCE TIME: ≤ 2 ms

**Electrical**
MAXIMUM VOLTAGE: 5 VDC.
CURRENT DC: 1 mA
DIELECTRIC STRENGTH: 50 VAC.
INSULATION RESISTANCE: 50 MΩ min / 50 VDC.
BOUNCE TIME: ≤ 5 ms

# Enkodery kwadraturowe (2)

- Różnica faz sygnałów generowanych przez enkodery cyfrowe (sygnały A i B) pozwala na określenie kierunku obrotu. Częstotliwość generowanego przebiegu prostokątnego pozwala na określenie prędkości kątowej.

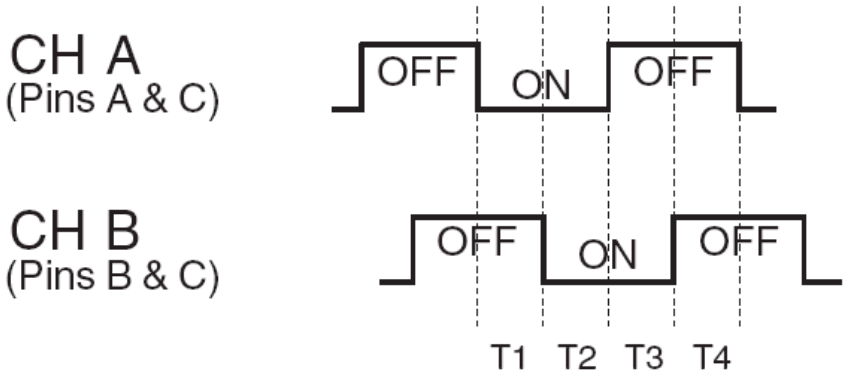



Analogowy układ filtrujący drgania styków

# Enkodery kwadraturowe (3)

- Wyjścia enkodera generują sygnał kwadraturowy – różnica faz obu sygnałów wynosi 90 stopni. Odfiltrowane sygnały wykorzystywane są do generowania impulsów inkrementujących lub dekrementujących licznik. W systemach mikroprocesorowych wykorzystuje się wyzwalanie dowolnym zboczem lub poziomem sygnału generowanego przez enkoder. W celu wyznaczenia kierunku obrotu należy zaimplementować maszynę stanową, jeżeli poprzednim stanem było “00”, a obecny stan wynosi “01” to enkoder obracany jest zgodnie ze wskazówkami zegara. Drgania styków mogą spowodować niewłaściwe działanie automatu stanowego, np. przejście 00->11 uniemożliwia określenie kierunku obrotu, mamy dwie możliwości: 00->01->11 oraz 00->10->11.

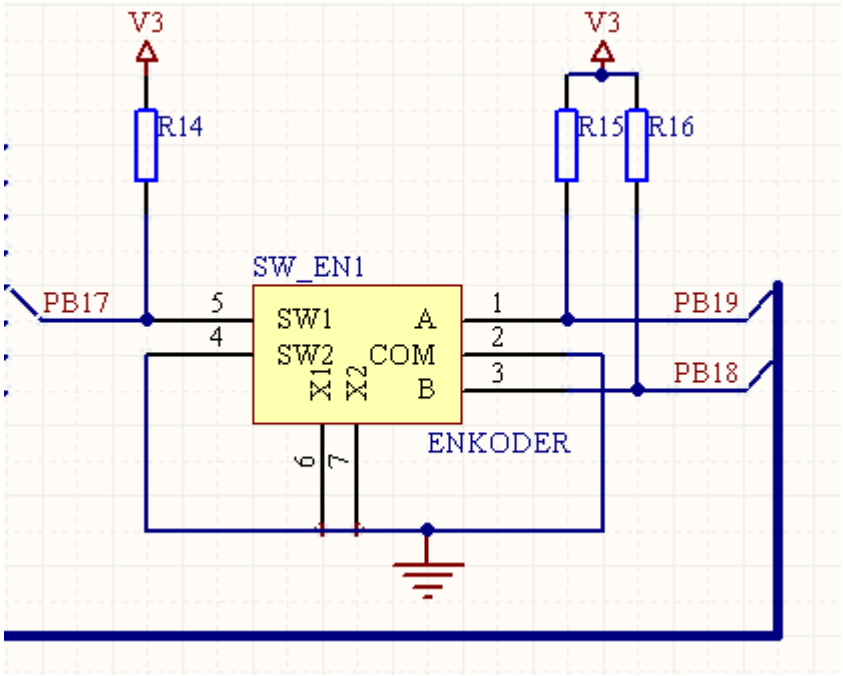



| Phase | A | B |
|---|---|---|
| 1 | 0 | 0 |
| 2 | 0 | 1 |
| 3 | 1 | 1 |
| 4 | 1 | 0 |

| Phase | A | B |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 1 | 1 |
| 3 | 0 | 1 |
| 4 | 0 | 0 |

# Enkodery kwadraturowe – laboratorium

- Schemat elektryczny układu enkodera podłączonego do procesora AMR wykorzystywanego na zajęciach.
- Enkoder wymaga dobrego algorytmu filtrującego drgania styków.

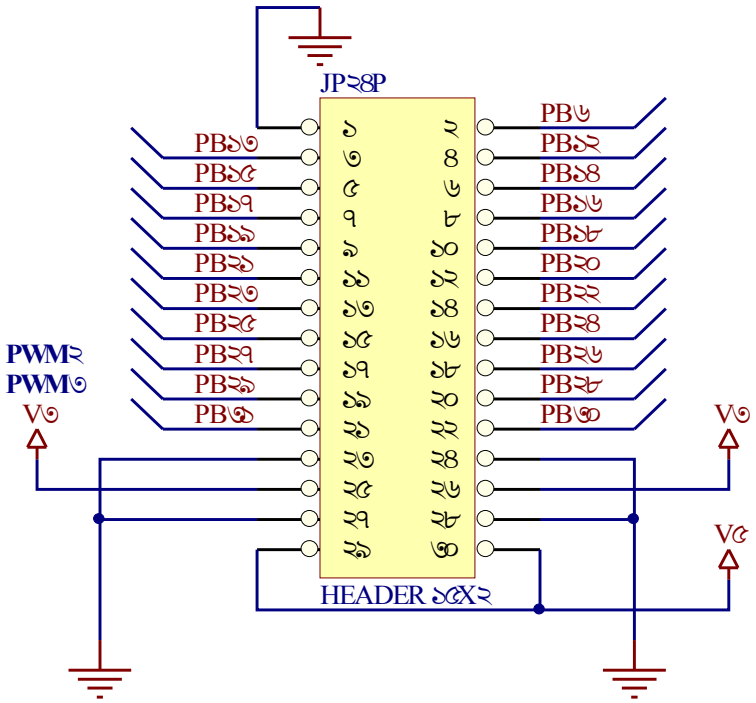




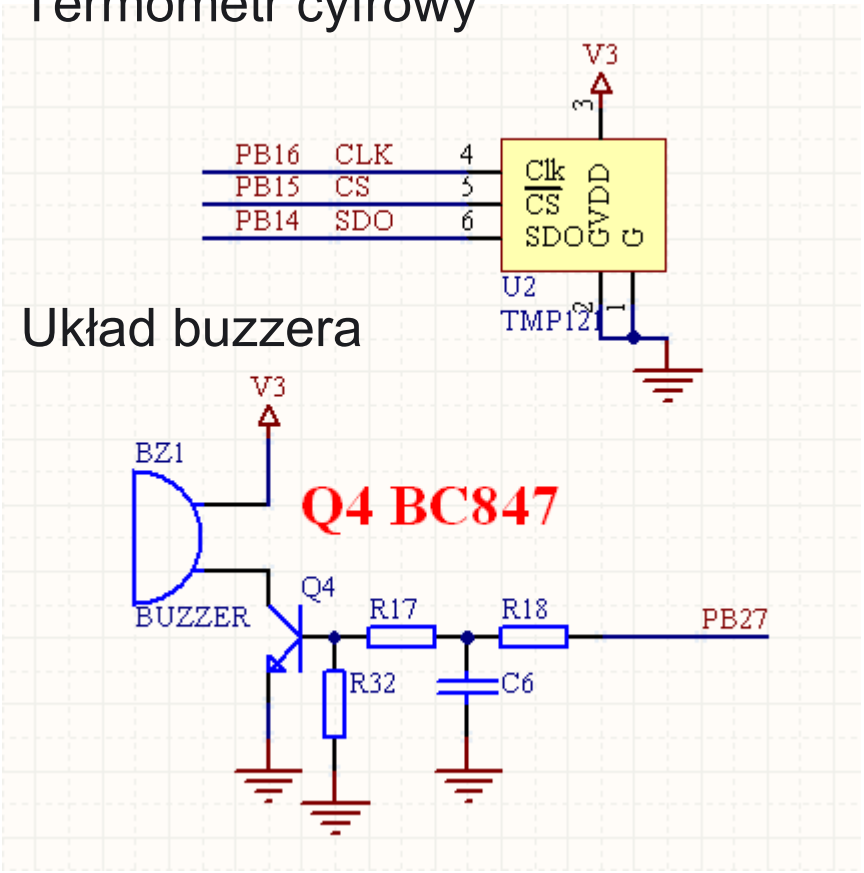

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

# Płyta nakładkowa – termometr, buzzer

## Złącze do procesora ARM

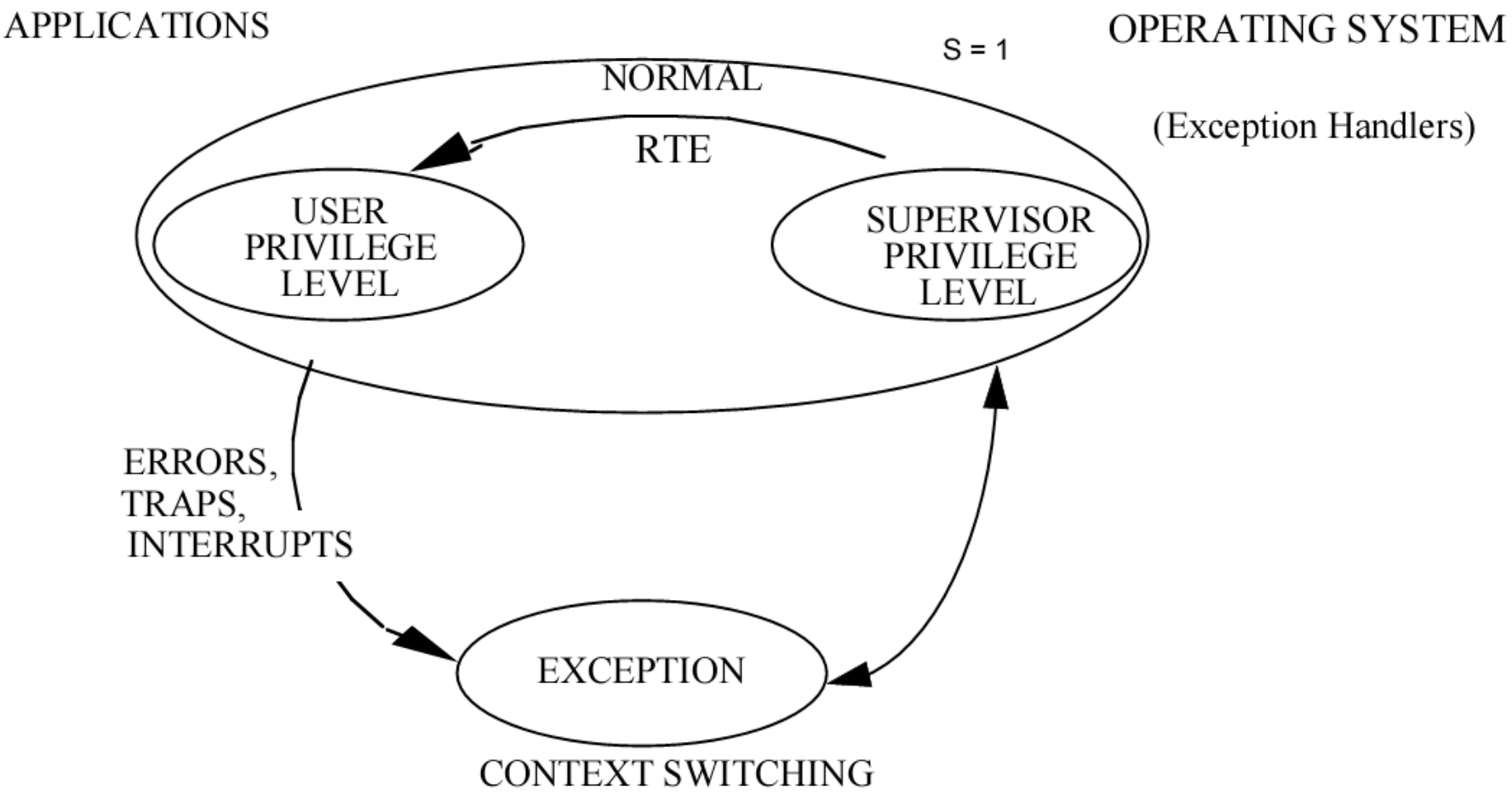



## Termometr cyfrowy



# Obsługa sytuacji wyjątkowych, kontroler przerwań AIC (Advanced Interrupt Controller)

# Obsługa sytuacji wyjątkowych



# Wyjątki

**Wyjątek (ang. exception)** – mechanizm kontroli przepływu danych występujący w mikroprocesorach oraz we współczesnych językach programowania służący do obsługi zdarzeń wyjątkowych, a w szczególności sytuacji błędnych. Szczególnym przypadkiem wyjątku jest przerwanie.

Wyjątki dzielimy na:

- przerwania (ang. interrupts),
- niepowodzenia (ang. fault),
- błędy nienaprawialne (ang. abort),
- pułapki (ang. trap).

Procesory ARM obsługują dwa rodzaje przerwań:

- **FIQ** (Fast interrupt) – przerwania z szybką obsługą,
- **IRQ** (Interrupt) – przerwania normalne.

# Przerwania

**Przerwanie (ang. interrupt)** lub żądanie przerwania (IRQ – Interrupt ReQuest) – sygnał powodujący zmianę przepływu sterowania, niezależnie od aktualnie wykonywanego programu. Pojawienie się przerwania powoduje wstrzymanie aktualnie wykonywanego programu i wykonanie przez procesor kodu procedury obsługi przerwania, uchwytu przerwania (ang. interrupt handler).

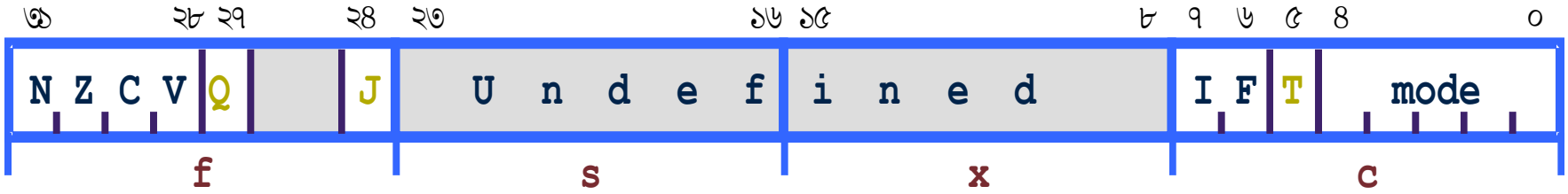
W procesorach ARM wystąpienie przerwania powoduje wygenerowanie sygnału IRQ lub FIQ oraz rozpoczęcie procedury obsługi przerwania. Jeżeli system przerwań jest aktywny (rdzeń procesora) oraz dane przerwanie nie jest zamaskowane (sterownik przerwań) następuje przyjęcie przerwania przez procesor - skok do programu obsługującego przerwanie.

Przykłady przerwań:

- odebranie lub zakończenie transmisji danej przez port szeregowy,
- zmiana stanu wyprowadzenia portu procesora.

Stan urządzenia można sprawdzać programowo, jednak wymaga to ciągłego sprawdzania stanu rejestru statusowego. Taka operacja nazywana jest odpytywaniem (ang. poolling). Powoduje to znaczne obciążenie procesora, np. transmisja jednego znaku zajmuje ok. 100 us (procesor może w tym czasie wykonać kilka tysiące operacji).

# Rejestr statusowy SPSR procesora ARM




## Wskaźniki stanu

- V – przepełnienie podczas operacji ALU (oVerflow)
- C – przeniesienie/pożyczka podczas operacji ALU
- Z – ujemny wynik podczas operacji ALU
- N – ujemny wynik operacji ALU lub „mniejszy niż”

**Wskaźniki dostępne jedynie dla architektury 5TE/J**

- J – Procesor w trybie Jazelle
- Q – Sticky Overflow – wskaźnik nasycenia podczas operacji ALU (QADD, QDADD, QSUB or QDSUB, lub rezultat operacji SMLAxy or SMLAWx przekracza 32-bity)

## Przerwania

- I=1 Przerwania IRQ wyłączone
- F=1 Przerwania FIQ wyłączone

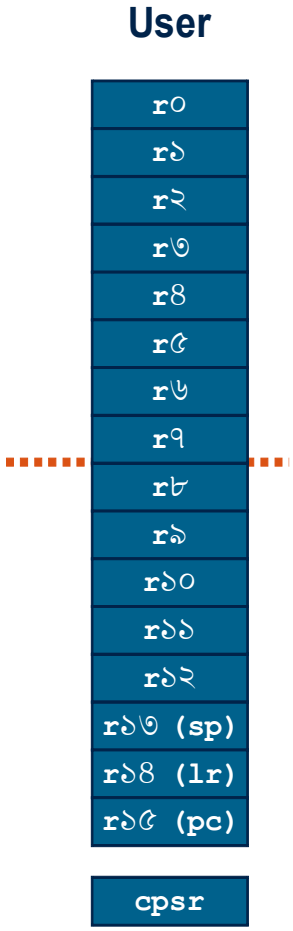
## Wskaźniki dostępne dla arch. xT

- T=0 Tryb pracy ARM
- T=1 Tryb pracy Thumb

## Tryb pracy procesora

- Definiują jeden z 7 trybów operacyjnych rdzenia procesora

# Model programowy procesora ARM




**Note: System mode uses the User mode register set**

# Obsługa sytuacji wyjątkowych

Wykonanie niedozwolonej operacji przez procesor w danym stanie uprzywilejowania powoduje wygenerowanie wyjątku.

Obsługa wyjątku obejmuje wszystkie operacje od momentu wykrycia błędu do pobrania pierwszej instrukcji obsługującej sytuację wyjątkową.

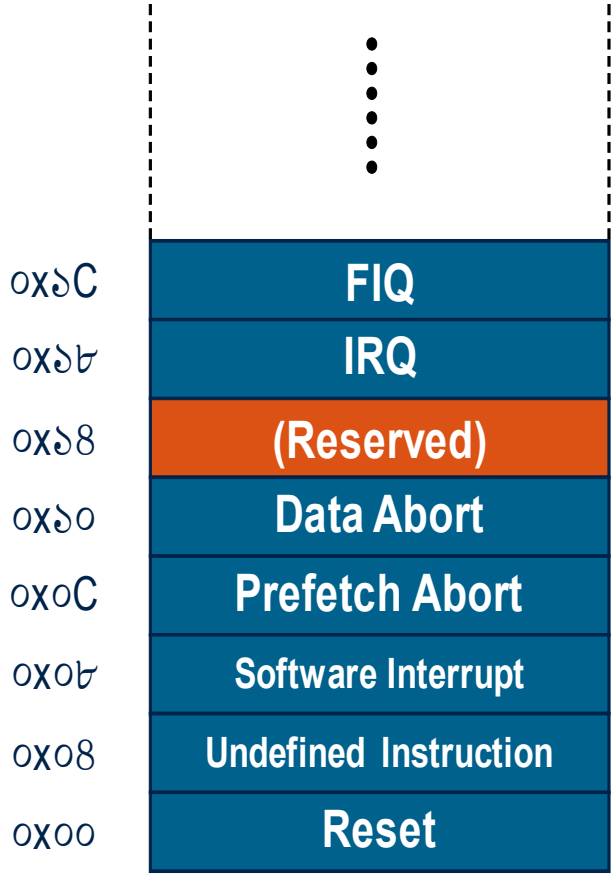
1. a) Wykonanie kopii CPSR → SPSR oraz PC (r15) → Link Register (r14),
   b) Przejście do trybu ARM (z trybu Thumb lub Jazelle),
   c) Przejście do trybu obsługi przerwań (FIQ/IRQ) lub wyjątków,
   d) Ustawienie maski IRQ na poziomie zgłaszanego przerwania (lub wyłączenie przerwań).
   e) Przełączenie banku rejestrów,
   f) Uaktywnienie rejestru SPSR.
2. Określenie wektora obsługiwanego wyjątku (przerwania).
3. Obliczenie adresu pierwszej instrukcji procedury obsługującej dany wyjątek (przerwanie).

# Powrót z obsługi sytuacji wyjątkowych

1. a) Odtworzenie rejestru CPSR (r15),
   b) PC (Link Register r14),
   c) Powrót do wykonywanego rozkazu.

Tablica wektorów przerwań umieszczona po resecie pod adresem 0x0.

Tablice można przesunąć pod adres 0xFFFF.0000 (ARM 7/9/10).



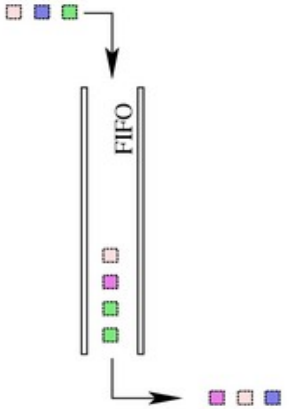

Fragment pamięci

# Struktura stosu (1)

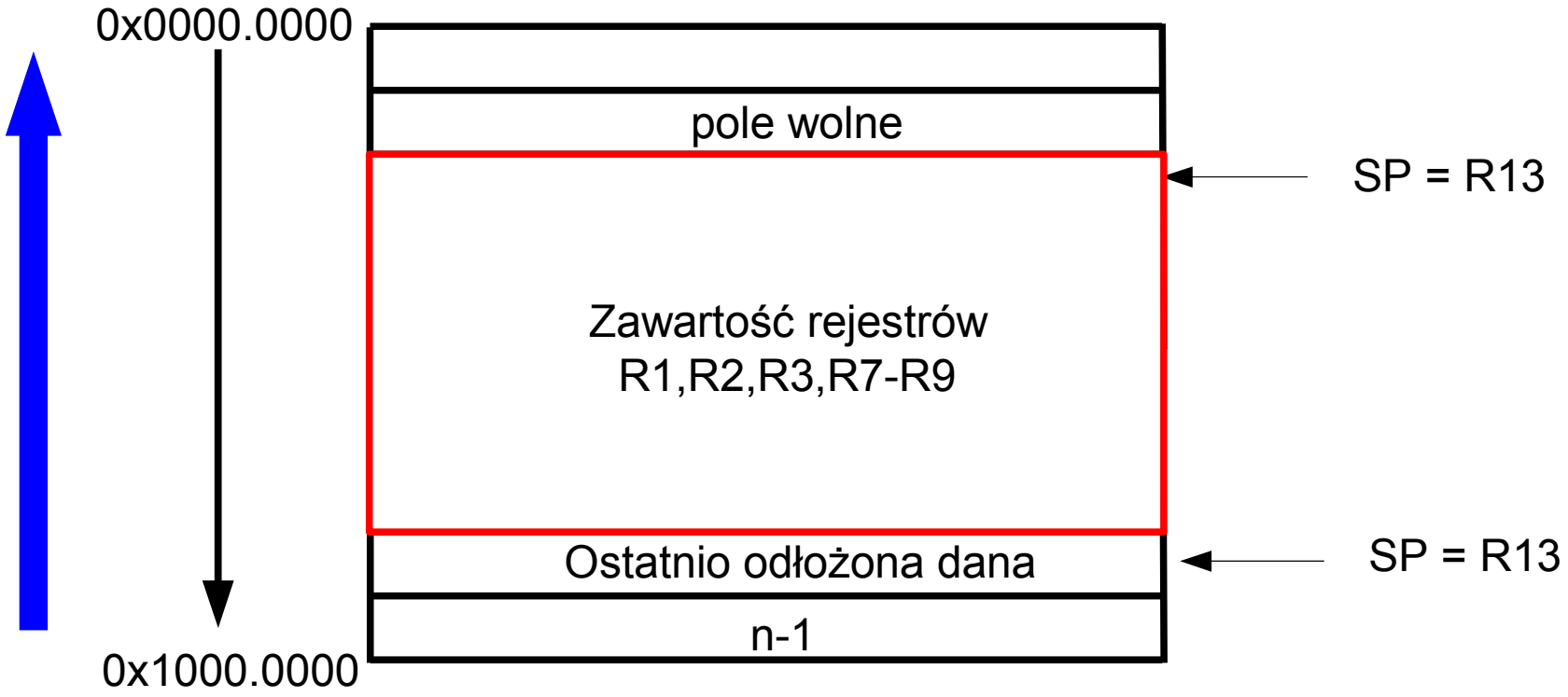
**Stos (ang. stack lub LIFO Last-In, First-Out) -** liniowa struktura danych, w której dane odkładane są na wierzch stosu i z wierzchołka stosu są zdejmowane. Ideę stosu danych można zilustrować jako stos położonych jedna na drugiej książek – nowy egzemplarz kładzie się na wierzch stosu i z wierzchu stosu zdejmuje się kolejne egzemplarze. Elementy stosu poniżej wierzchołka stosu można wyłącznie obejrzeć, aby je ściągnąć, trzeba najpierw po kolei ściągnąć to, co jest nad nimi

**FIFO (ang. First In, First Out) -** przeciwieństwem stosu LIFO jest kolejka, bufor typu FIFO (pierwszy na wejściu, pierwszy na wyjściu), w którym dane obsługiwane są w takiej kolejności, w jakiej zostały dostarczone (jak w kolejce do kasy)

First–in First–out (FIFO)

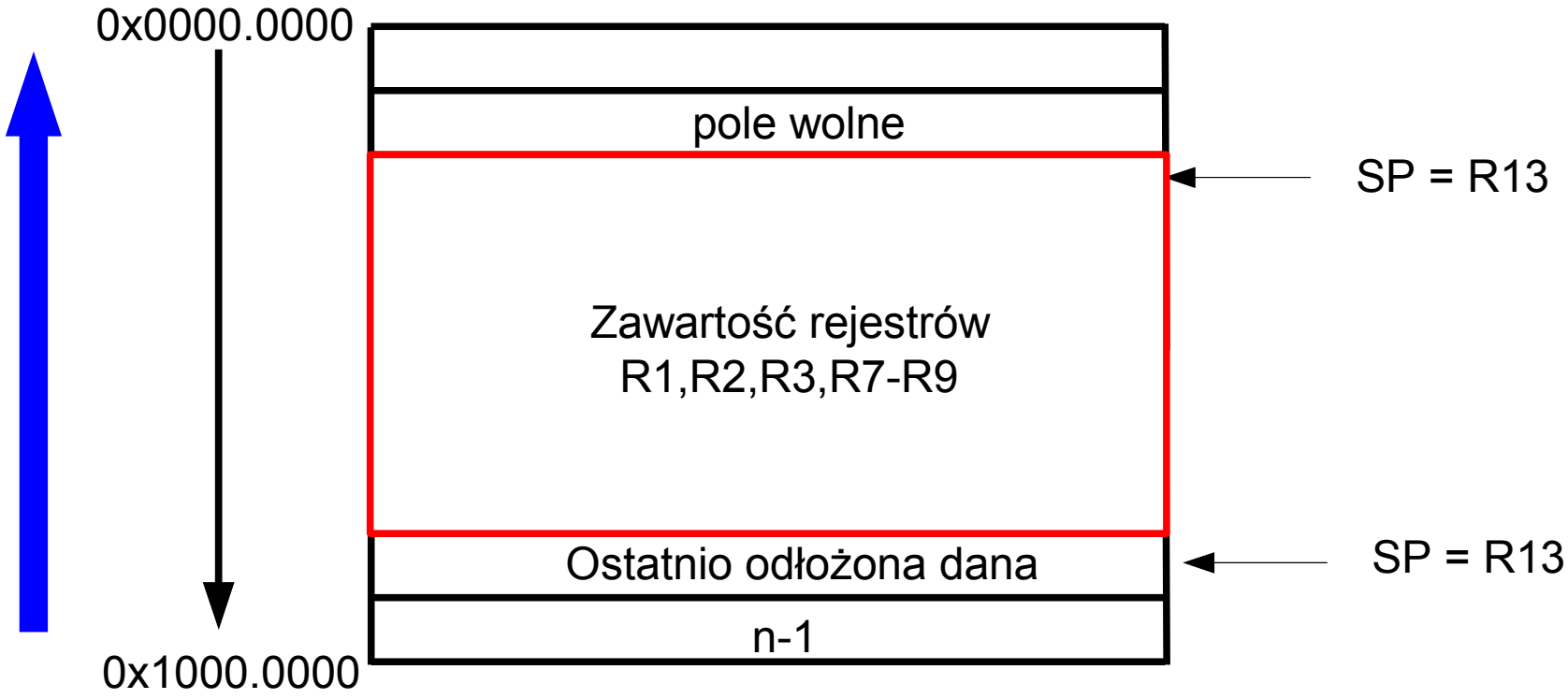


# Struktura stosu – odłożenie danej na stos



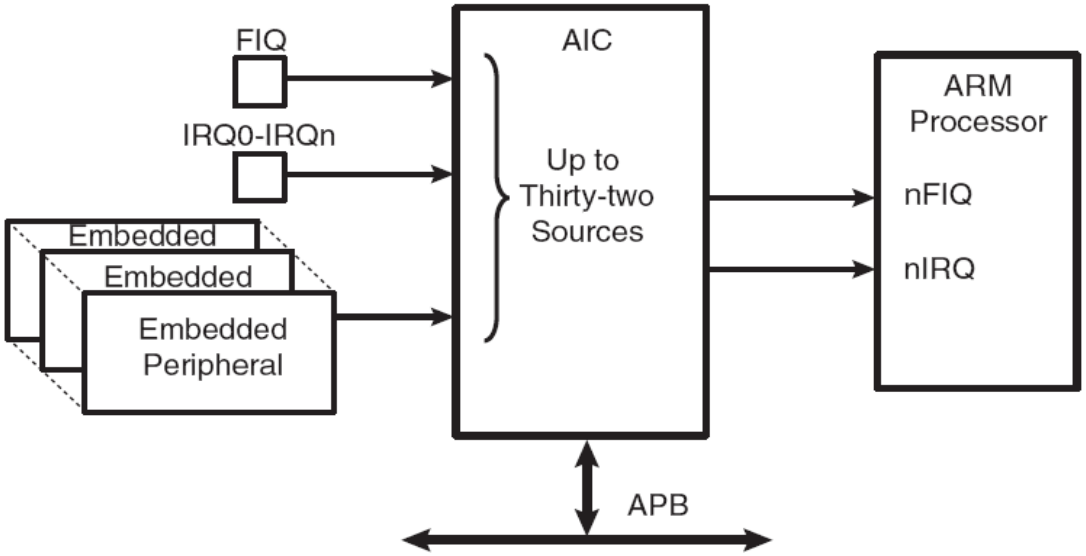

```
STMDB   SP!, {lista rejestrów}      | zmniejszenie SP o 24, odłożenie zawartości rejestrów  na
STMDB   SP!, {R1,R2,R3,R7-R9}         stos,
```

# Struktura stosu – zdjęcie danej ze stosu




```
LDMIA   SP!, {lista rejestrów}      | zwiększenie SP o 24, odłożenie zawartości rejestrów  na
LDMIA   SP!, {R1,R2,R3,R7-R9}         stos,
```

# Schemat blokowy sterownika przerwań procesora ARM




- Obsługa przerwań wektorowych,
- Obsługa do 32 przerwań zewnętrznych i wewnętrznych,
- Możliwość maskowania dowolnego przerwania,
- Obsługa przerwań nIRQ i szybkich nFIR (ang. Fast Interrupt Request),
- 8 poziomów priorytetów (0- najniższy, 7- najwyższy),
- Obsługa przerwań wyzwalanych poziomem lub zboczem.

# Schemat blokowy sterownika przerwań procesora ARM

- Sterownik przerwań wykorzystuje zegar systemowy. Zegar doprowadzany jest przez cały czas pracy procesora (nie ma możliwości odcięcia zegara).
- Przerwania mogą zostać wykorzystane do wyprowadzenia procesora ze stanu uśpienia (Idle mode).
- Przerwanie o numerze 0 (FIQ) jest zawsze przerwaniem typu FIQ.
- Przerwanie o numerze 1 (SYS) sumą logiczną przerwań od urządzeń peryferyjnych procesora. W procedurze obsługi przerwania należy określić urządzenie/a zgłaszające przerwanie/a.
- Przerwania o numerach 2-31 (PID2-PID31) mogą zostać dołączone do urządzeń peryferyjnych (użytkownika) lub portów I/O.
- Sterownik obsługuje przerwania wyzwalane poziomem lub zboczem sygnału.

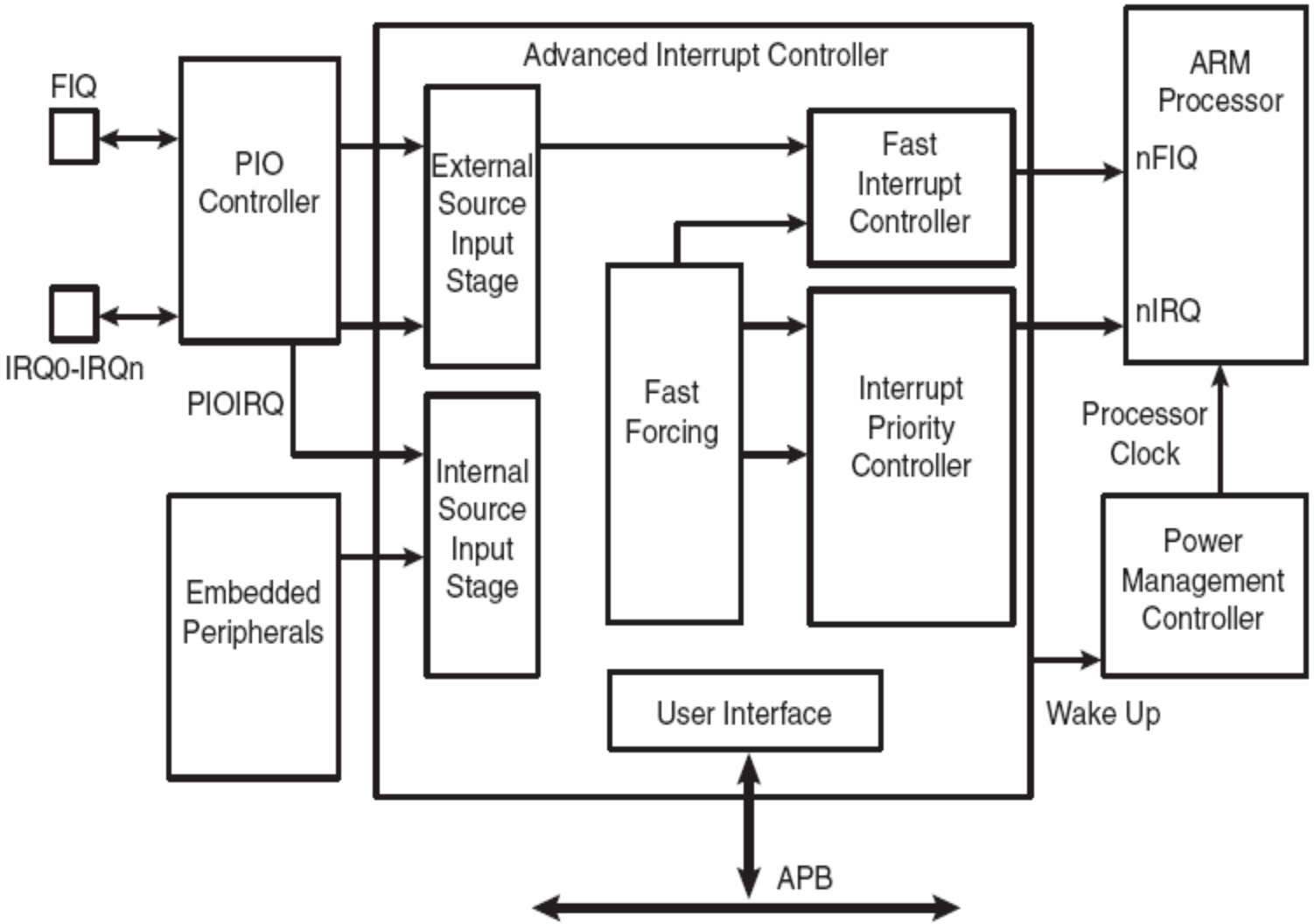
# Przerwania współdzielone

Blok urządzeń systemowy (AT91C_ID_SYS) dysponuje jednym, wspólnym przerwaniem SYS (ang. shared interrupt) o numerze ID=1, które obejmuje następujące urządzenia:

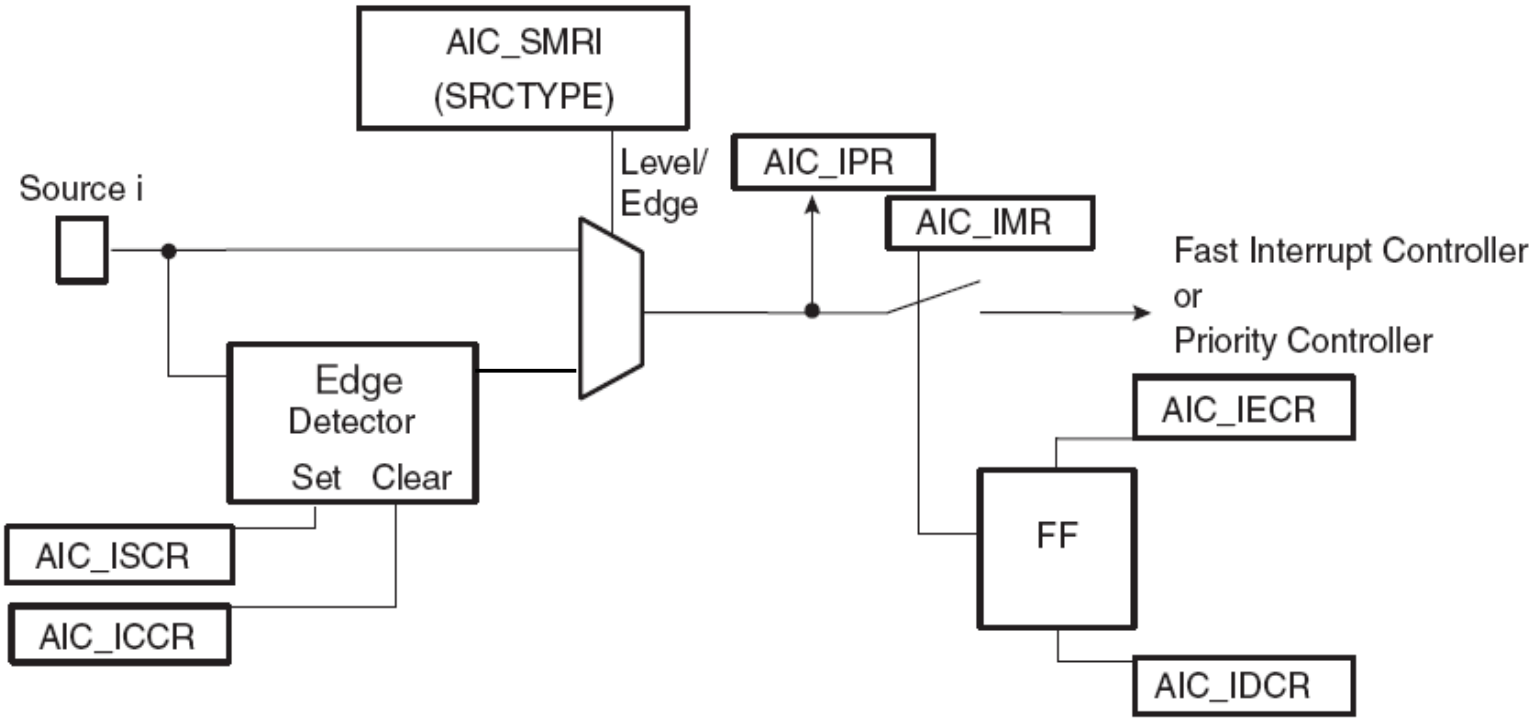
- timery PIT, RTT, WDT,
- interfejs diagnostyczny DBGU,
- Sterownik DMA PMC,
- Układ zerowania procesora RSTC,
- Sterownik pamięci MC.

W procedurze obsługi przerwania SYS należy sprawdzić kolejno stan wszystkich urządzeń, czy występują przerwania odmaskowane. Jeżeli przerwanie jest aktywne należy sprawdzić flagę sygnalizującą przerwanie w rejestrze statusu danego urządzenia. Jeżeli flaga jest ustawiona należy wykonać program związany z obsługą przerwania od danego urządzenia.

# Szczegółowy schemat blokowy sterownika AIC

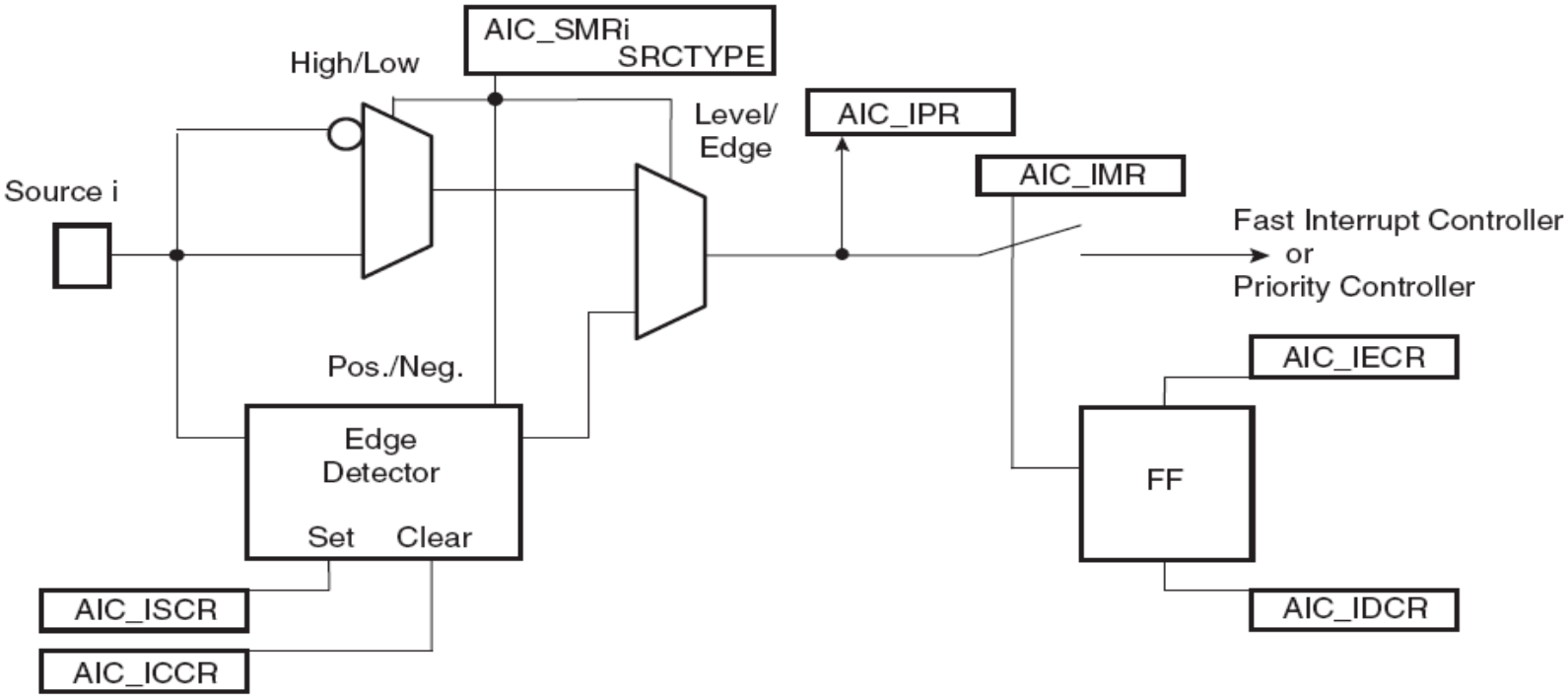


# Przerwania wewnętrzne




- Maska przerwań – AIC_IECR/IDCR (status → AIC_IMR),
- Wyczyszczenie flagi przerwania podczas odczytu rejestru AIC_IVR (przerwania FIQ → AIC_FVR),
- Status przerwań dostępny w rejestrze AIC_IPR

# Przerwania zewnętrzne




- Możliwość wyboru zbocza opadającego/narastającego lub poziomu niskiego/wysokiego do generacji przerwań.

# Definicja numerów urządzeń peryferyjnych

```
// ********************************************************************************
//               PERIPHERAL ID DEFINITIONS FOR AT91SAM9263
// ********************************************************************************
#define AT91C_ID_FIQ    ( 0) // Advanced Interrupt Controller (FIQ)
#define AT91C_ID_SYS    ( 1) // System Controller
#define AT91C_ID_PIOA   ( 2) // Parallel IO Controller A
#define AT91C_ID_PIOB   ( 3) // Parallel IO Controller B
#define AT91C_ID_PIOCDE ( 4) // Parallel IO Controller C, Parallel IO Controller D, Parallel IO Controller E
#define AT91C_ID_US0    ( 7) // USART 0
#define AT91C_ID_US1    ( 8) // USART 1
#define AT91C_ID_US2    ( 9) // USART 2
#define AT91C_ID_MCI0   (10) // Multimedia Card Interface 0
#define AT91C_ID_MCI1   (11) // Multimedia Card Interface 1
#define AT91C_ID_CAN    (12) // CAN Controller
#define AT91C_ID_TWI    (13) // Two-Wire Interface
#define AT91C_ID_SPI0   (14) // Serial Peripheral Interface
```

**ID=0, ID=30-31 przerwania zewnętrzne, pozostałe to przerwania wewnętrzne.**

# Rejestry sterownika przerwań (1)

| Offset | Register | Name | Access | Reset |
|---|---|---|---|---|
| 0x00 | Source Mode Register 0 | AIC_SMR0 | Read-write | 0x0 |
| 0x04 | Source Mode Register 1 | AIC_SMR1 | Read-write | 0x0 |
| --- | --- | --- | --- | --- |
| 0x7C | Source Mode Register 31 | AIC_SMR31 | Read-write | 0x0 |
| 0x80 | Source Vector Register 0 | AIC_SVR0 | Read-write | 0x0 |
| 0x84 | Source Vector Register 1 | AIC_SVR1 | Read-write | 0x0 |
| --- | --- | --- | --- | --- |
| 0xFC | Source Vector Register 31 | AIC_SVR31 | Read-write | 0x0 |
| 0x100 | Interrupt Vector Register | AIC_IVR | Read-only | 0x0 |
| 0x104 | FIQ Interrupt Vector Register | AIC_FVR | Read-only | 0x0 |
| 0x108 | Interrupt Status Register | AIC_ISR | Read-only | 0x0 |
| 0x10C | Interrupt Pending Register(3) | AIC_IPR | Read-only | 0x0(1) |
| 0x110 | Interrupt Mask Register(3) | AIC_IMR | Read-only | 0x0 |
| 0x114 | Core Interrupt Status Register | AIC_CISR | Read-only | 0x0 |
| 0x118 - 0x11C | Reserved | --- | --- | --- |
| 0x120 | Interrupt Enable Command Register(3) | AIC_IECR | Write-only | --- |
| 0x124 | Interrupt Disable Command Register(3) | AIC_IDCR | Write-only | --- |
| 0x128 | Interrupt Clear Command Register(3) | AIC_ICCR | Write-only | --- |
| 0x12C | Interrupt Set Command Register(3) | AIC_ISCR | Write-only | --- |
| 0x130 | End of Interrupt Command Register | AIC_EOICR | Write-only | --- |
| 0x134 | Spurious Interrupt Vector Register | AIC_SPU | Read-write | 0x0 |
| 0x138 | Debug Control Register | AIC_DCR | Read-write | 0x0 |
| 0x13C | Reserved | --- | --- | --- |
| 0x140 | Fast Forcing Enable Register(3) | AIC_FFER | Write-only | --- |
| 0x144 | Fast Forcing Disable Register(3) | AIC_FFDR | Write-only | --- |
| 0x148 | Fast Forcing Status Register(3) | AIC_FFSR | Read-only | 0x0 |
| 0x14C - 0x1E0 | Reserved | --- | --- | --- |
| 0x1EC - 0x1FC | Reserved | | | |

# Rejestry sterownika przerwań

```
typedef struct _AT91S_AIC {
    AT91_REG    AIC_SMR[32];    // Source Mode Register
    AT91_REG    AIC_SVR[32];    // Source Vector Register
    AT91_REG    AIC_IVR;        // IRQ Vector Register
    AT91_REG    AIC_FVR;        // FIQ Vector Register
    AT91_REG    AIC_ISR;        // Interrupt Status Register
    AT91_REG    AIC_IPR;        // Interrupt Pending Register
    AT91_REG    AIC_IMR;        // Interrupt Mask Register
    AT91_REG    AIC_CISR;       // Core Interrupt Status Register
    ...
} AT91S_AIC, *AT91PS_AIC;
#define AT91C_BASE_AIC      (AT91_CAST(AT91PS_AIC)   0xFFFFF000) // (AIC)
    Base Address
```

# Rejestry sterownika przerwań (2)

**AIC_SMR[32]; // Source Mode Register – konfiguracja poziomu oraz sposobu wyzwalania**

**AIC_SVR[32]; // Source Vector Register – uchwyty obsługujące przerwania**

**AIC_IVR; // IRQ Vector Register – adres uchwytu do obsługiwanego przerwania**

AIC_FVR; // FIQ Vector Register – adres uchwytu do obsługiwanego przerwania

AIC_ISR; // Interrupt Status Register – numer obsługiwanego przerwania

AIC_IPR; // Interrupt Pending Register – rejestr z flagami oczekujących przerwań 0-31

AIC_IMR; // Interrupt Mask Register – rejestr z maskami przerwań 0-31

AIC_CISR; // Core Interrupt Status Register – stan przerwań rdzenia IRQ/FIQ

**AIC_IECR; // Interrupt Enable Command Register – rejestr uaktywniający przerwania**

AIC_IDCR; // Interrupt Disable Command Register – rejestr wyłączający przerwania

**AIC_ICCR; // Interrupt Clear Command Register – rejestr kasujący flagi przerwań**

AIC_ISCR; // Interrupt Set Command Register – rejestr ustawiający flagi przerwań

**AIC_EOICR; // End of Interrupt Command Register – koniec obsługi przerwania**

AIC_SPU; // Spurious Vector Register – handler do przerwania fałszywego

# Procedura obsługująca przerwanie od timera PIT i klawiatury

**Ustawienie adresu funkcji (handlera) obsługującego przerwanie (adres 32-bitowy)**

AT91C_BASE_AIC->AIC_SVR[AT91C_ID_SYS] = (unsigned long) TIMER_handler;

**Procedura obsługi przerwania od timera**

void TIMER_handler (void) {

Odczyt rejestru statutowego PITC_PISR

jeżeli flaga od timera INT_ENABLE jest ustawiona (rejestr PITC_PIMR) to odczyt rejestru PITC_PIVR - skasowanie flagi przerwania

jeżeli nie to inne urządzenie peryferyjne zgłosiło przerwanie – odpowiednia reakcja

}

**Procedura obsługi przerwania od klawiatury**

void BUTTON_handler (void) {

Odczyt rejestru statutowego PIO_ISR - skasowanie flagi przerwania

jeżeli flaga na odpowiednim bicie rejestru PIO_ISR jest ustawiona to oznacza to wciśnięcie przycisku

}

# Konfiguracja przerwań od klawiatury

**Przyciski dołączone są do portu C – przerwania generowane przez układy wejściowe portu C/D/E (maska AT91C_ID_PIOCDE)**

**Konfiguracja przerwań od portów C/D/E:**

1. Konfiguracja portów procesora jako porty wejściowe (przycisk lewy i prawy)
2. Wyłączenie przerwań generowanych przez porty C/D/E (rejestr AIC_IDCR, AT91C_ID_PIOCDE)
3. Ustawienie wskaźnika do procedury obsługującej przerwanie dla portów C/D/E w tablicy wektorów SVR (AIC_SVR[AT91C_ID_PIOCDE])
4. Konfiguracja poziomu i metody wyzwalania przerwania (rejestr AIC_SMR, wyzwalanie wysokim poziomem AT91C_AIC_SRCTYPE_EXT_HIGH_LEVEL oraz priorytet AT91C_AIC_PRIOR_HIGHEST)
5. Wyczyszczenie flagi przerwania portów C/D/E (rejestr AIC_ICCR)
6. Włącznie przerwań dla obu przycisków (rejestr PIO_IER)
7. Włączenie przerwania od portów C/D/E (AIC_IECR)
8. Włączenie portu IO

# Konfiguracja przrwań od Timera PIT

**Timer PIT generuje przerwania o numerze 1 – przerwania od urządzeń peryferyjnych (System Controller, maska AT91C_ID_SYS)**

## Konfiguracja przerwań od Timera PIT

1. Konfiguracja okresu timera, np. 5 ms
2. Wyłączenie przerwania od Timera PIT na czas konfiguracji (AIC_IDCR, przerwanie nr 1 - urządzenia peryferyjne procesora, stała AT91C_ID_SYS)
3. Ustawienie wskaźnika do procedury obsługującej przerwanie dla urządzeń peryferyjnych w tablicy wektorów AIC_SVR (AIC_SVR[AT91C_ID_SYS])
4. Konfiguracja poziomu i metody wyzwalania przerwania (rejestr AIC_SMR, wyzwalanie AT91C_AIC_SRCTYPE_INT_LEVEL_SENSITIVE, dowolny poziom, np. AT91C_AIC_PRIOR_LOWEST)
5. Wyczyszczenie flagi przerwania urządzeń peryferyjnych (rejestr AIC_ICCR)
6. Włącznie przerwania urządzeń peryferyjnych AT91C_ID_SYS (rejestr AIC_IECR)
7. Włączenie przerwania od Timera (AT91C_PITC_PITIEN)
8. Włączenie Timera PIT (AT91C_PITC_PITEN)
9. Wyzerowanie tzw. licznika lokalnego Timera (zmienna Local_Counter)

# Funkcje obsługujące przerwania w języku C (1)

Deklaracja procedur obsługujących przerwania (kompilator GCC) wymaga użycia dyrektywy preprocesora __attribute__ ((interrupt("IRQ"))):

```
void INTButton_handler()__attribute__ ((interrupt("IRQ")));
void INTPIT_handler()__attribute__ ((interrupt("IRQ")));
void Soft_Interrupt_handler()__attribute__ ((interrupt("SWI")));
void Abort_Exception_handler()__attribute__ ((interrupt("ABORT")));
void Undef_Exception_handler()__attribute__ ((interrupt("UNDEF")));
```

Funkcja obsługująca przerwanie nie różni się od klaszycznej funkcji w języku C

```
void INTButton_handler() {
      // standard C function
}
```

Podczas pisania programów na laboratorium nie należy używać atrybutu __attribute__ ((interrupt("IRQ"))) – przerwania zagnieżdzone obsługiwane są przez funkcję umieszczoną w pliku startowym (startup.S).

# Funkcje obsługujące przerwania w języku C (2)

```
.size     Open_INTButtons, .-Open_INTButtons
.align    2
.global   INTButton_handler
INTButton_handler:
@ Interrupt Service Routine.
@ args = 0, pretend = 0, frame = 8
@ frame_needed = 1, uses_anonymous_args = 0
str  ip, [sp, #-4]!      // store R12
mov  ip, sp              // move sp to ip
stmfd     sp!, {r1, r2, r3, fp, ip, lr, pc}
sub  fp, ip, #4          // allocate dataframe
sub  sp, sp, #8          // on stack for local
                         // data
```

**Właściwy program obsługujący przerwanie**

```
sub  sp, fp, #24         // rem. frame from st
ldmfd     sp, {r1, r2, r3, fp, sp, lr}
ldmfd     sp!, {ip}      // recover R12
subs pc, lr, #4          // return from INT
```

```
.size     Open_INTButtons, .-Open_INTButtons
.align    2
.global   INTButton_handler
.type     INTButton_handler, %function
INTButton_handler:
@ args = 0, pretend = 0, frame = 8
@ frame_needed = 1, uses_anonymous_args = 0

mov  ip, sp              // store R12
stmfd     sp!, {fp, ip, lr, pc}
sub  fp, ip, #4          // allocate dataframe
sub  sp, sp, #8          // on stack for local
                         //  data
```

**Właściwy program obsługujący przerwanie**

```
sub  sp, fp, #12         // rem. frame from stac
ldmfd     sp, {fp, sp, pc} // recover registers
                           // and return from
                           // function
```

# Interfejs szeregowy SPI
# Serial Peripheral Interface

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

# Serial Peripheral Interface

**Cechy interfejsu SPI:**

- Szeregowa transmisja synchroniczna,
- Transfer full duplex, master-slave lub master-multi-slave,
- Duża szybkość transmisji (>12 Mbit/s),
- Zastosowanie:
  - układy peryferyjne (ADC, DAC, RTC, EEPROM, termometry, itp),
  - sterowanie pomocnicze (matryca CCD z szybkim interfejsem równoległym),
  - karty pamięci z interfejsem szeregowym SD/SDHC/MMC.

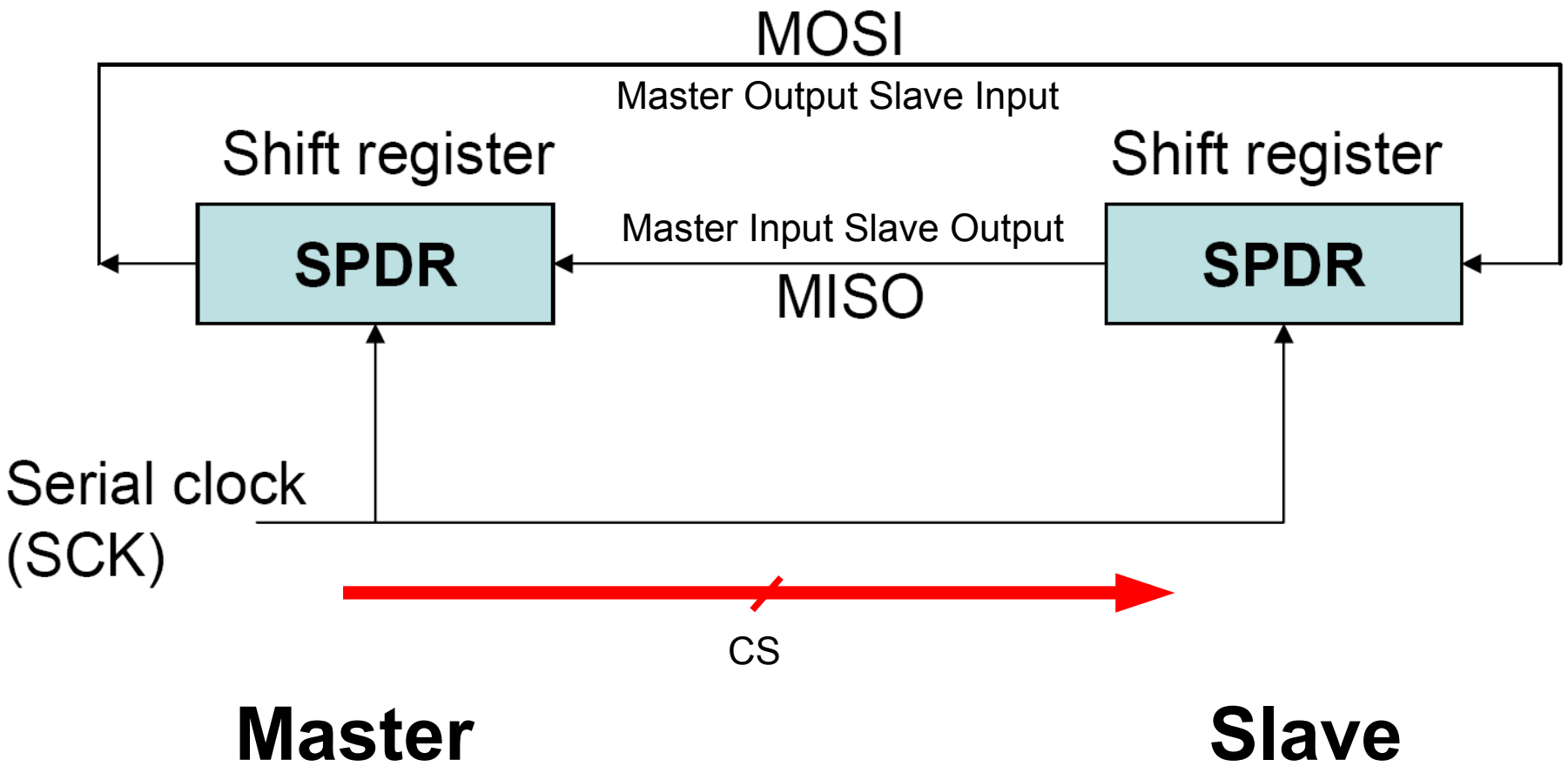
# Zalety standardu SPI

- Wymaga użycia tylko 3 lub 4 pinów procesora,
- Komunikacja Full duplex,
- Większa szybkość transmisji niż I2C (TWI, SMBUS),
- Swoboda wyboru wielkości pakietu (8, 16, 32,... -bity),
- Prosta budowa transceivera SPI,
- Sygnały jednokierunkowe – łatwo zrealizować izolację galwaniczną,
- Stosunkowa duża szybkość transmisji (rzędu 10 Mbit/s).

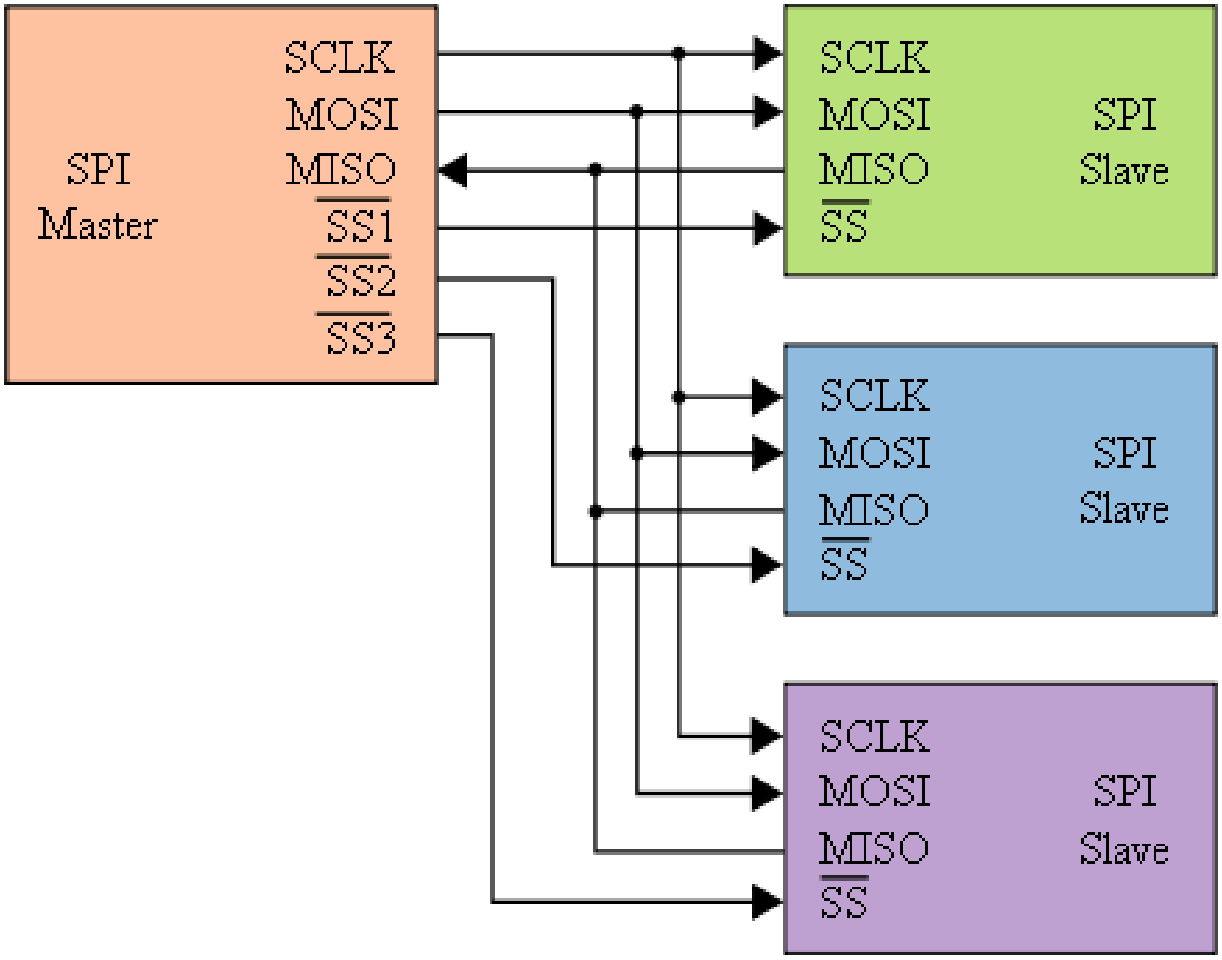
# Wady standardu SPI

- Wymaga większej liczby pinów niż one-wire lub I2C,
- Ograniczona liczba urządzeń dołączonych do magistrali (sygnały CS wymagają więcej pinów),
- Brak sprzętowej kontroli transmisji danych,
- Brak potwierdzenia transmisji – Master może wysyłać dane do urządzenia, które nie istnieje,
- Brak możliwości pracy w trybie Multi-Master,
- Transmisja na niewielkie odległości (< 50 cm).

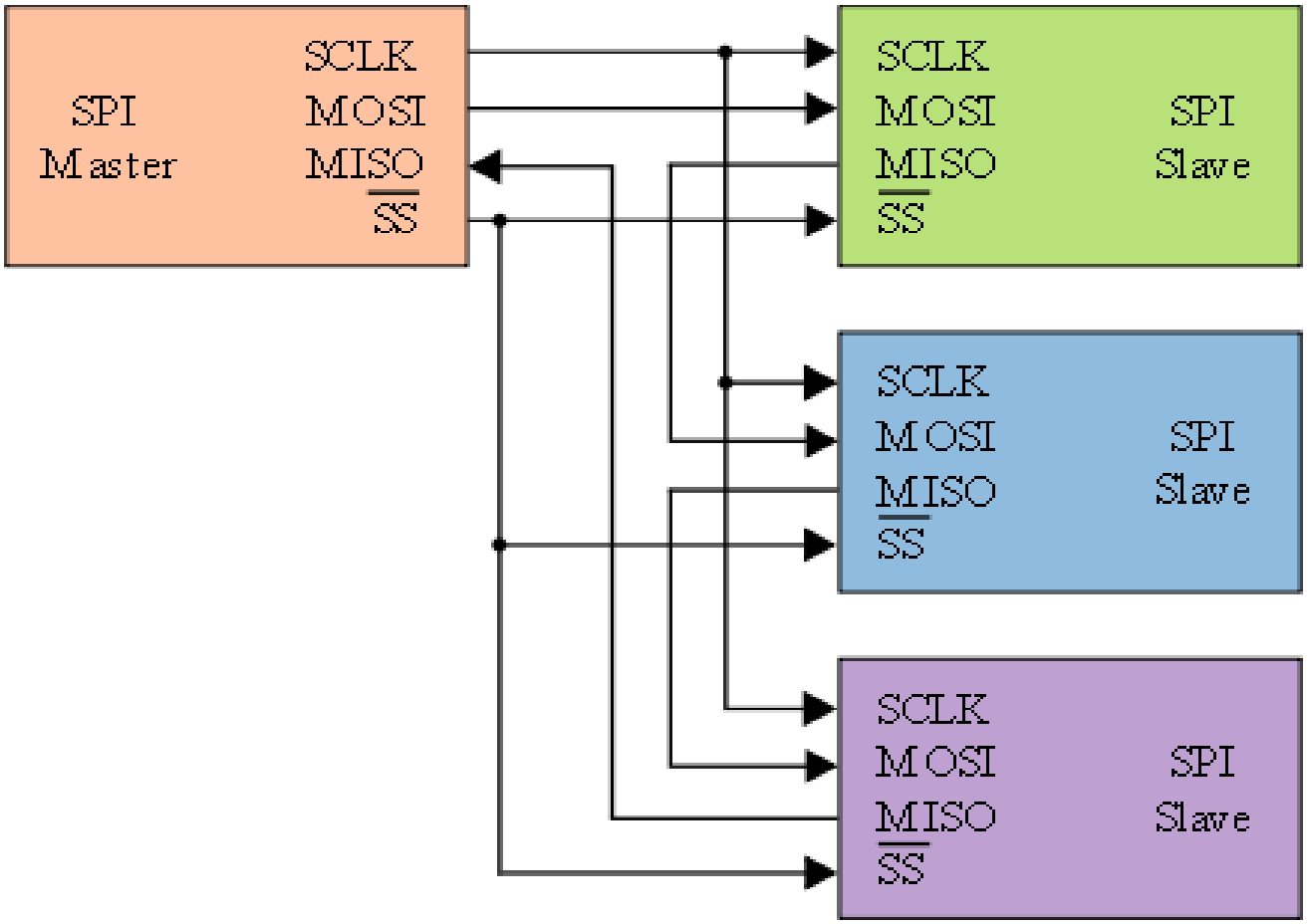
# Serial Peripheral Interface (1)



# Serial Peripheral Interface (2)



# SPI w połączeniu łańcuchowym



# Protokół interfejsu SPI

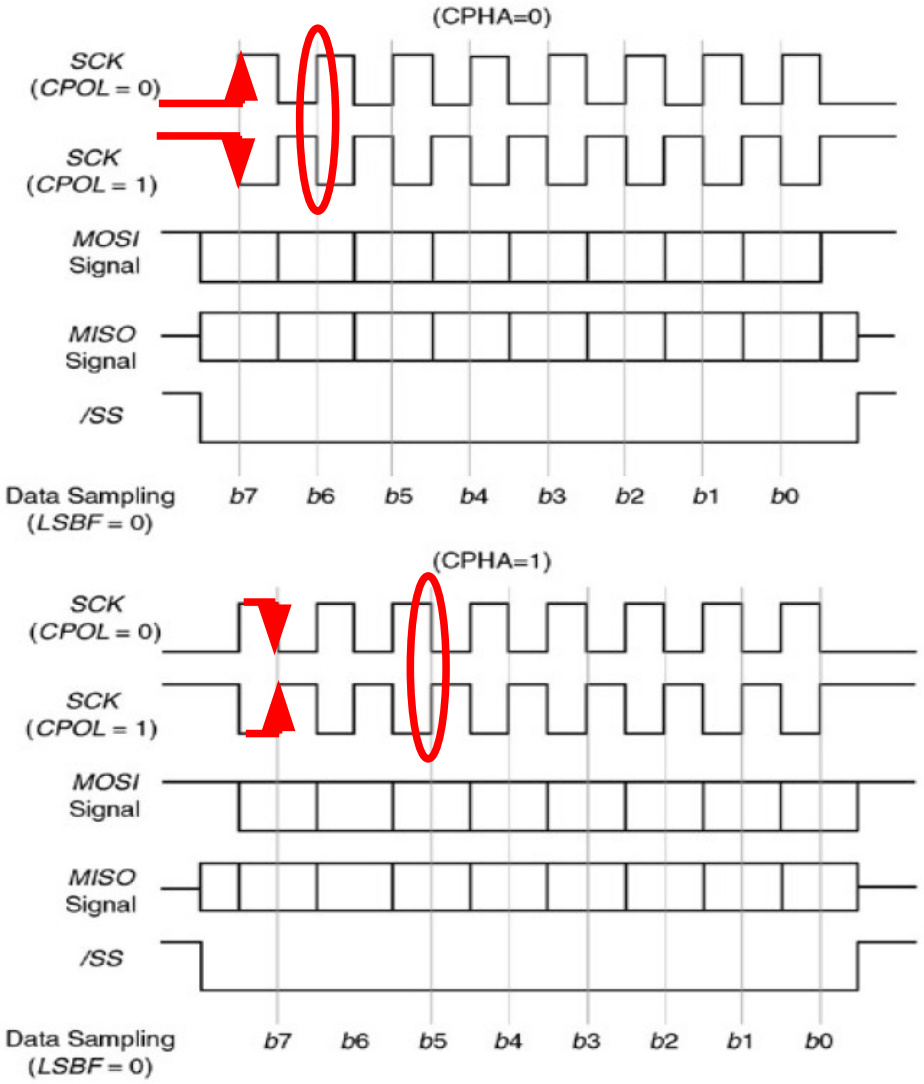
**Konfiguracja sygnału zegarowego:**

**Polaryzacja zegara:**
Polaryzacja **ujemna CPOL = 0** (**stan niski**, 8 impulsów zegara),

Polaryzacja **dodatnia CPOL = 1** (**stan wysoki**, 8 ujemnych impulsów zegara).

**Faza zegara:**
Zerowa faza zegara (próbkowanie na pierwszym zboczu zegara),

Opóźniona faza zegara (próbkowanie na drugim zboczu zegara).



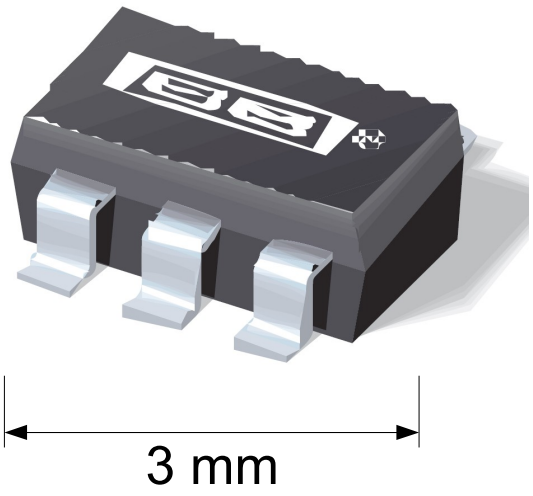
# Układu wykorzystujące interfejs SPI

- Czujniki temperatury, ciśnienia,
- Zegary czasu rzeczywistego (RTC),
- Przetworniki ADC, DAC,
- Wyświetlacze LCD,
- Ekrany czułe na dotyk (ang. Touch screen),
- Pamięci FLASH, EEPROM,
- Karty pamięci MMC, SD lub SDIO,
- Protokół JTAG (Joint Test Action Group) – wykorzystywany do testowania połączeń elektrycznych układów w obudowach BGA i płytek drukowanych,
- Protokół QSPI (Queued Serial Peripheral Interface).

# Termometr z interfasem SPI

**TMP 121:**

- Obudowa SOT 23-6,
- fclk mak. = 15 MHz
- Interfejs: SPI-Compatible Interface
- Rozdzielczość: 12-Bit + Sign, 0,0625°C
- Dokładność: ±1.5°C od −25°C do +85°C
- Pobór prądu w stanie uśpienia: 50µA (mak.)
- Zasilanie: 2,7V to 5,5V




| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|
| T12 | T11 | T10 | T9 | T8 | T7 | T6 | T5 |

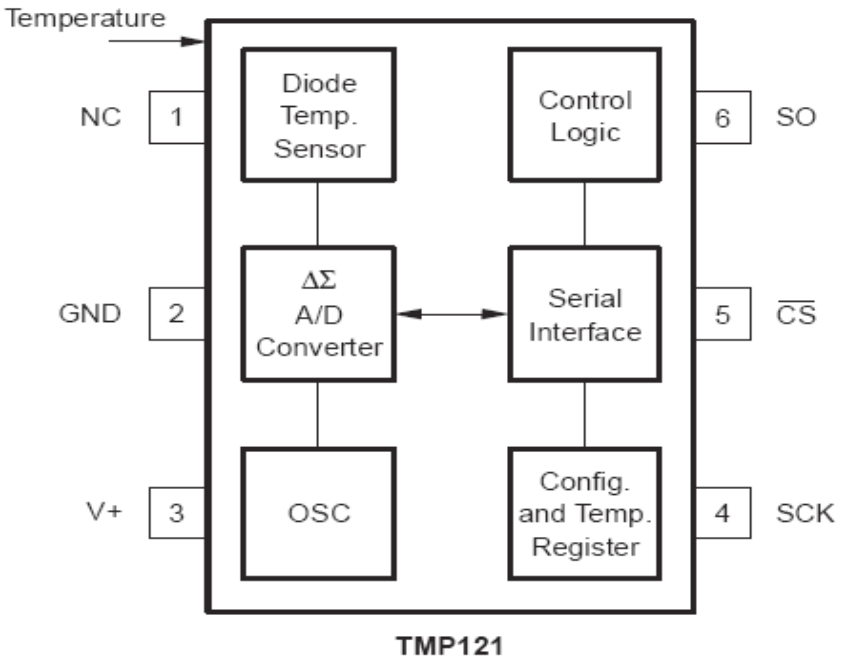
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| T4 | T3 | T2 | T1 | T0 | 0 | Z | Z |

Table 1. Temperature Register

| TEMPERATURE (°C) | DIGITAL OUTPUT(1) (BINARY) | HEX |
|---|---|---|
| 150 | 0100 1011 0000 0000 | 4B00 |
| 125 | 0011 1110 1000 0000 | 3E80 |
| 25 | 0000 1100 1000 0000 | 0C80 |
| 0.0625 | 0000 0000 0000 1000 | 0008 |
| 0 | 0000 0000 0000 0000 | 0000 |
| −0.0625 | 1111 1111 1111 1000 | FFF8 |
| −25 | 1111 0011 1000 0000 | F380 |
| −55 | 1110 0100 1000 0000 | E480 |
| (1) The last two bits are high impedance and are shown as 00 in the table. | | |

Table 2. Temperature Data Format

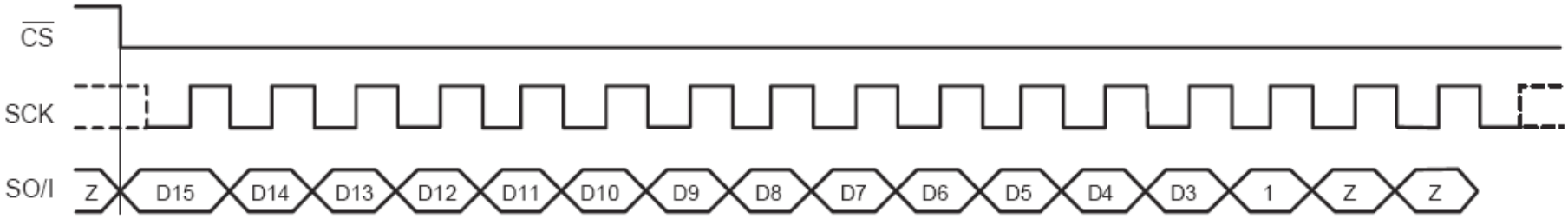
# Ramka SPI termometru TMP121




| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|
| T12 | T11 | T10 | T9 | T8 | T7 | T6 | T5 |

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| T4 | T3 | T2 | T1 | T0 | 0 | Z | Z |

Table 1. Temperature Register

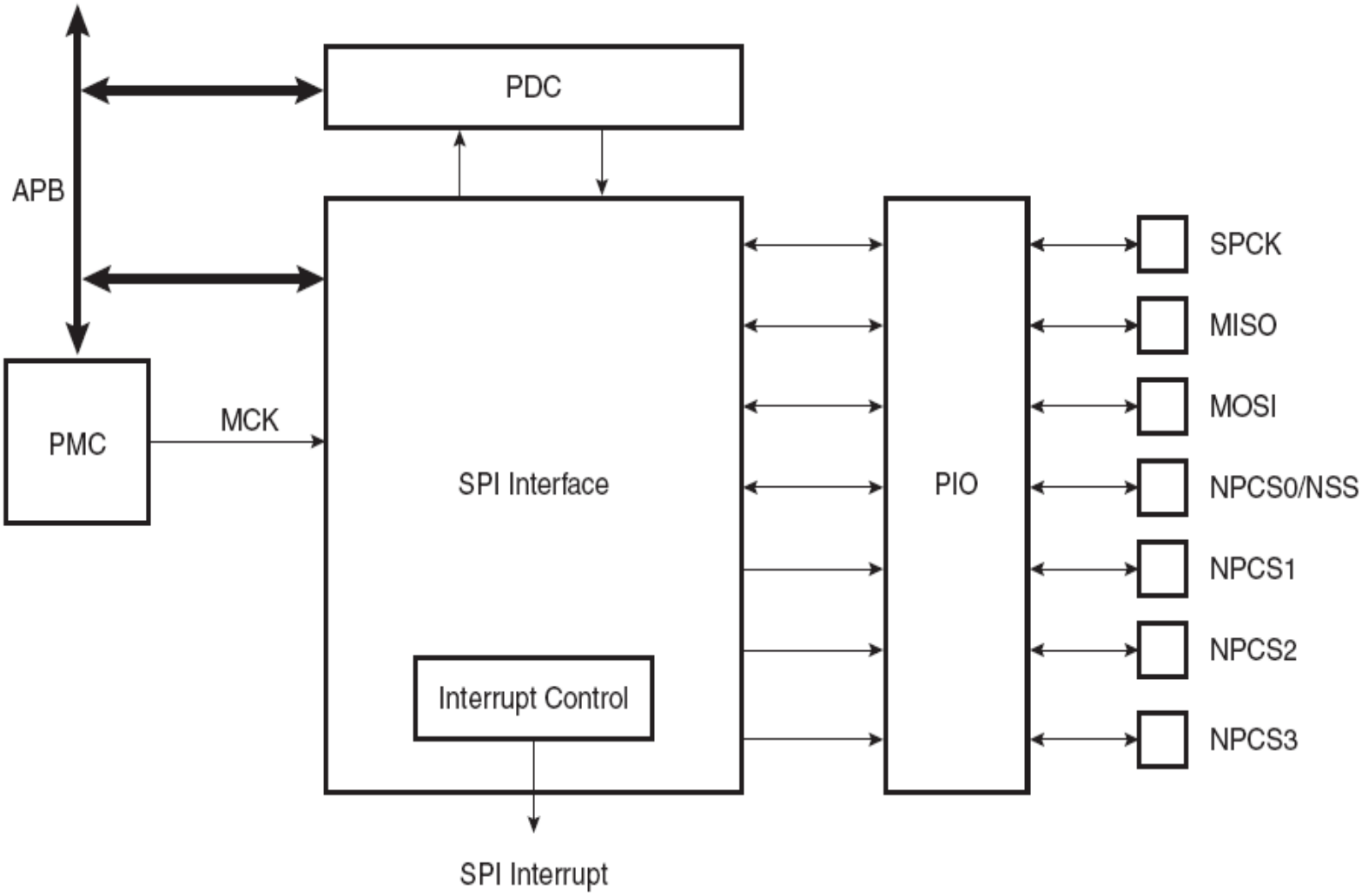


# Moduł SPI procesora ARM AT91SAM9263 (1)

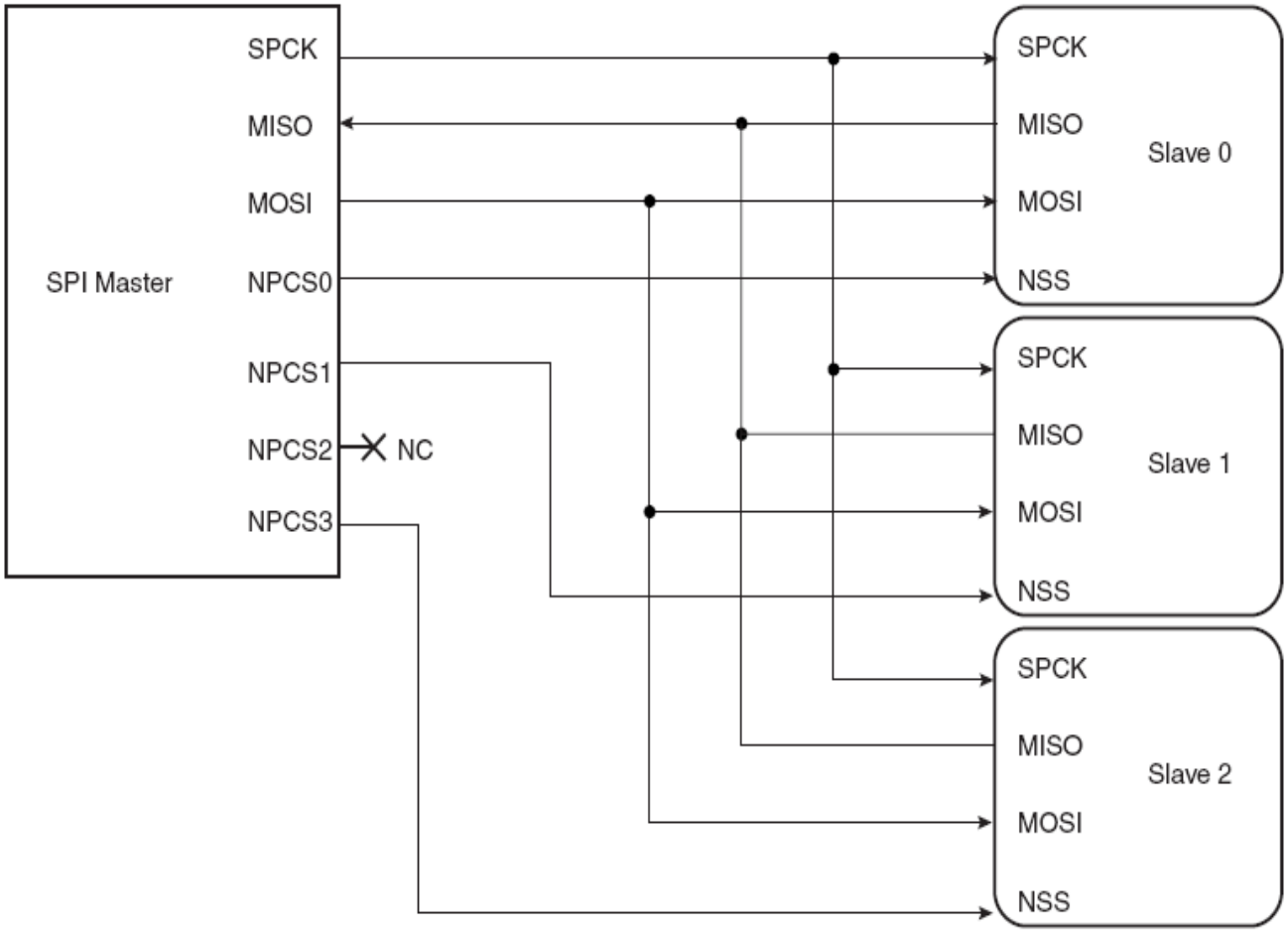
**Cechy modułu SPI:**

- Obsługa transferów w trybie Master lub Slave,
- Bufor nadawczy, odbiorczy oraz bufor transceivera,
- Transfery danych od 8 do 16 bitów,
- Cztery programowalne wyjścia aktywujące urządzenia dołączone do SPI (obsługa do 15 urządzeń),
- Programowalne opóźnienia pomiędzy transferami,
- Programowalna polaryzacja i faza zegara,
- 13 rejestrów do konfiguracji modułu SPI.

# Moduł SPI procesora ARM AT91SAM9263 (2)



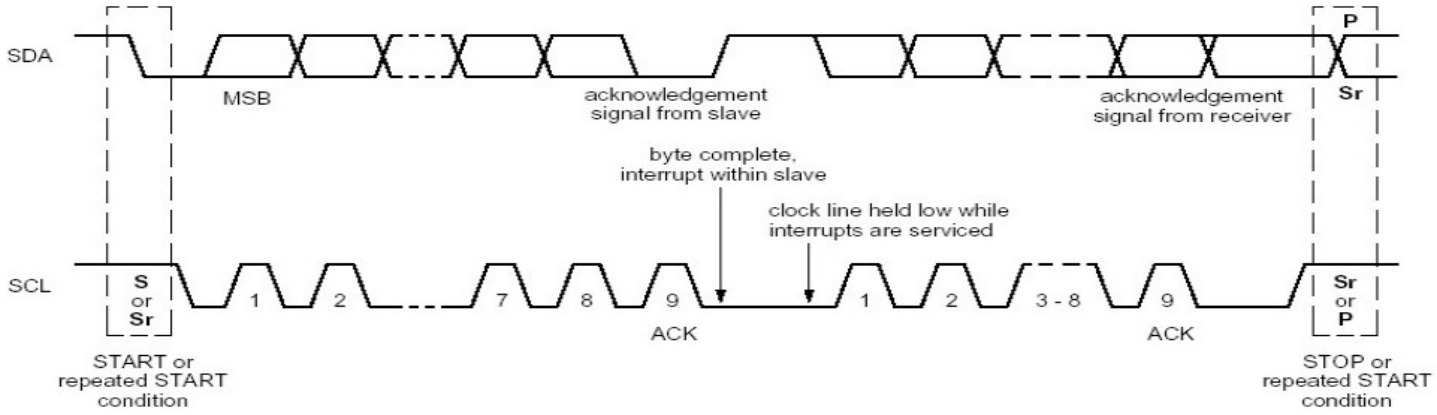
# Moduł SPI procesora ARM (3)



# Magistrala I2C

# Magistrala I2C

- Standard opracowany przez firmę Philips na początku lat 80,
- Dwuprzewodowy interfejs synchroniczny (SDA – linia danych, SCL – linia zegara),
- Transmisja dwukierunkowa, typu master-slave (multi-master), ramki 8-bitowe,
- Szybkość transmisji:
  - 100 kbps (standard mode),
  - 400 kbps (fast mode),
  - 3,4 Mbps (high-speed mode),
- Urządzenia posiadają niepowtarzalne adresy (7-bitów lub 10-bitów),
- Synchronizacja przy pomocy sygnału zegarowego umożliwia pracę urządzeń komunikujących się z różnymi szybkościami,
- Liczba urządzeń dołączonych do magistrali ograniczona jest pojemnością mag. (400 pF),
- Mechanizmy arbitrażu umożliwiające uniknięcie kolizji i utraty danych.

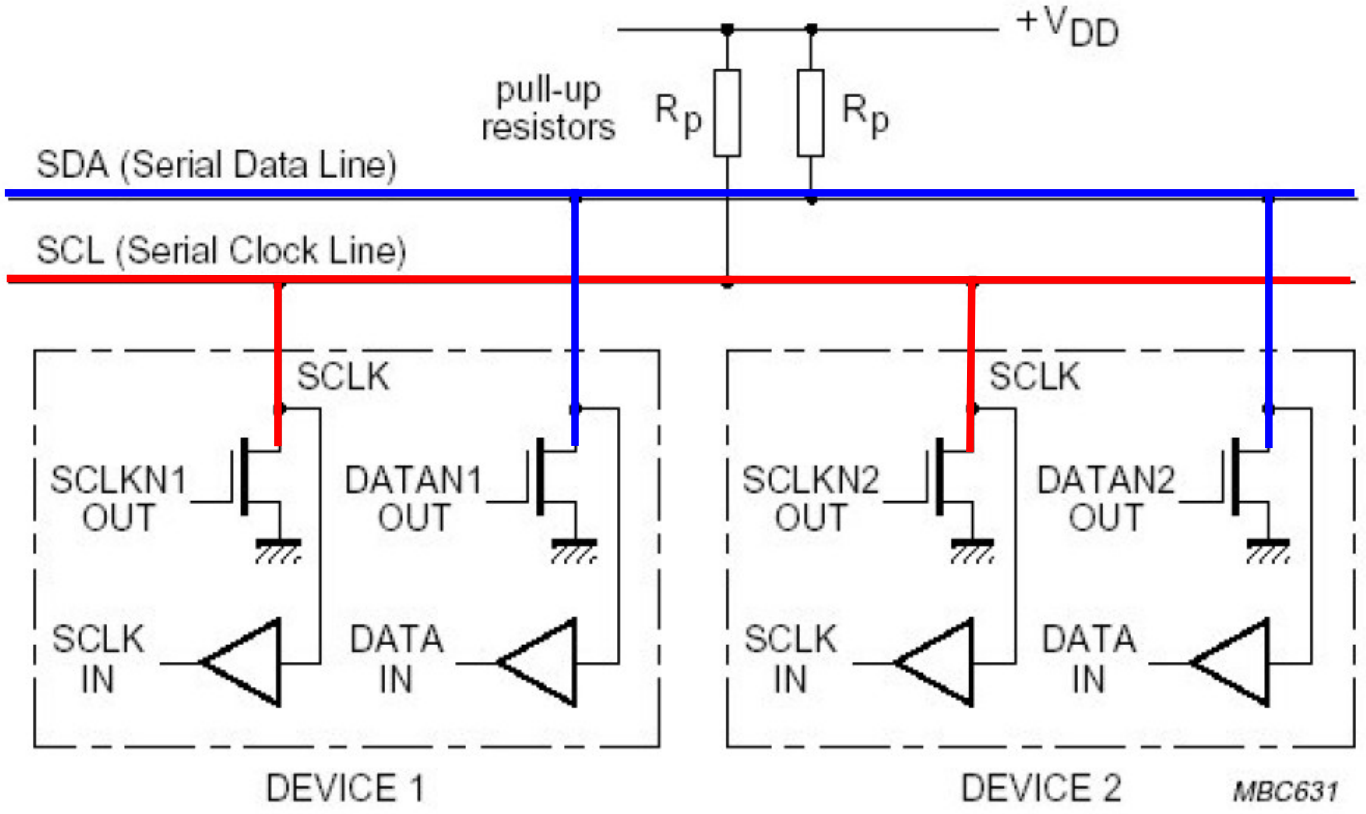


# Zastosowanie interfejsu I2C

W sprzedaży dostępnych jest wiele bardzo tanich układów scalonych sterowanych poprzez I²C:

- PCF8563/8583 - zegar, kalendarz, alarm, timer, dodatkowo może służyć jako RAM
- PCF8574 - pseudo-dwukierunkowy 8-bitowy ekspander
- PCF8576, PCF8577 - sterowniki wyświetlaczy LCD
- PCF8582 - pamięć EEPROM 256 bajtów (1, 2, 4 kB, ... MB)
- PCF8591 - 8-bitowy, 4-kanałowy przetwornik analogowo-cyfrowy i cyfrowo-analogowy

# Magistrala I2C

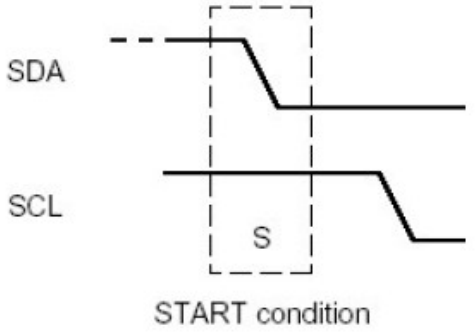



Urządzenie nadrzędne (Master) – inicjuje transmisję, generuje sygnał zegarowy

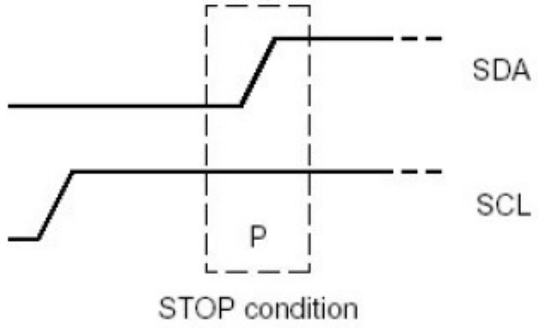
Urządzenie podrzędne (Slave) – analizuje wysłany przez urządzenie adres i transmituje lub odbiera dane.

# Rozpoczęcie oraz zakończenie transmisji

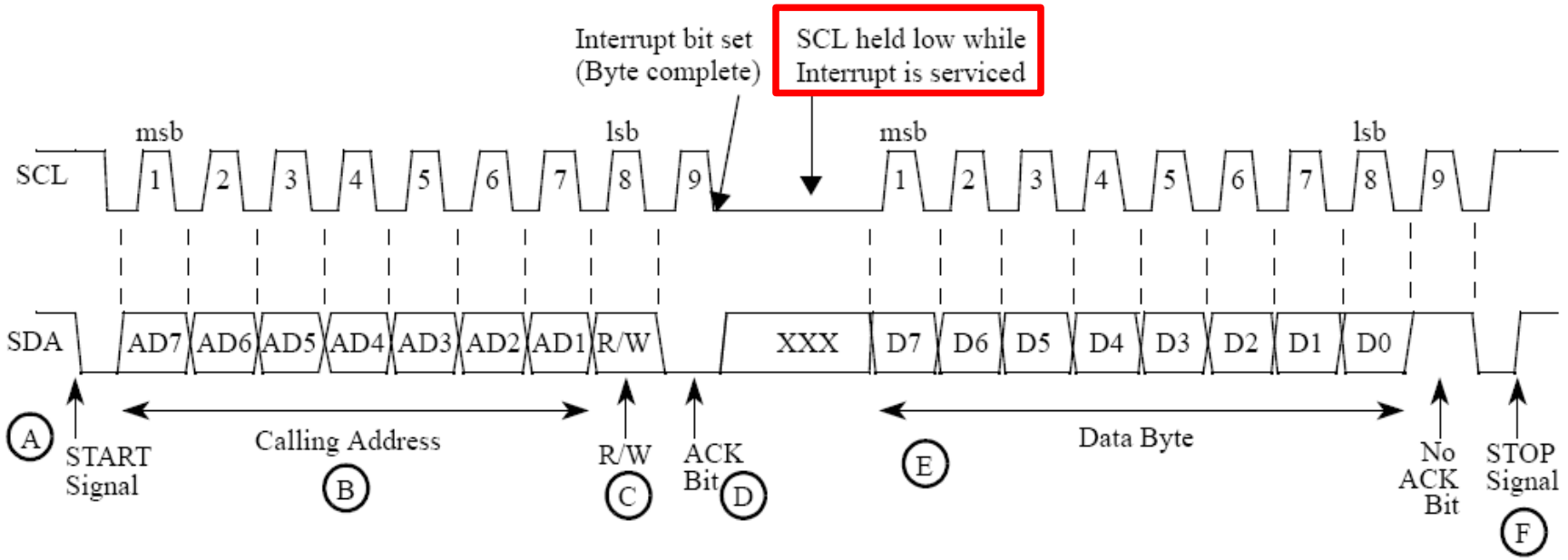
Rozpoczęcie transmisji – generacja sygnału **START** (opadające zbocze na szynie SDA, zmiana stanu z “1” na “0” logiczne, podczas ważnego sygnału SCL = ”1”). Sygnał generuje Master.




Zakończenie transmisji – generacja sygnału **STOP** (narastające zbocze na szynie SDA, zmiana stanu z “0” na “1” logiczną,podczas ważnego sygnału SCL = ”1”). Sygnał generuje Master.



# Protokół I2C

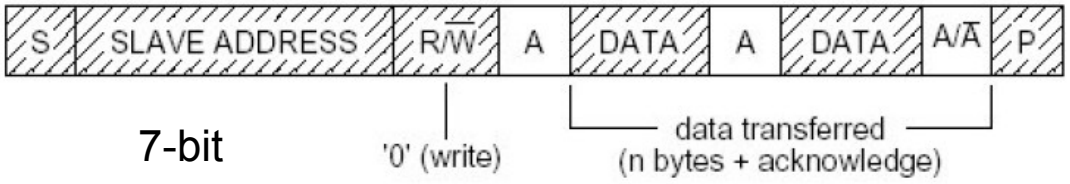



A) Transmisje rozpoczyna Master generując sygnał START.
B) Następnie transmituje 8 bitów danych (7 bitów adresowych, bit R/W).
C) Po transmisji 8 bitów Slave przejmuje magistralę i wymusza odpowiedni poziom na linii SDA (9 takt zegara). Odpowiada w ten sposób bitem potwierdzenia ACK (brak potwierdzenia, ACK = “1”).
E) Po przesłaniu adresu następuje faza odczytu lub zapisu danej do obsługiwanego urządzenia (8 bitów danych).
F) Po przesłaniu danych urządzenie nadrzędne kończy transmisję generując brak potwierdzenia (ACK = “1”) oraz bit stopu.

# Zapis lub odczyt

## Zapis n-bajtów danych

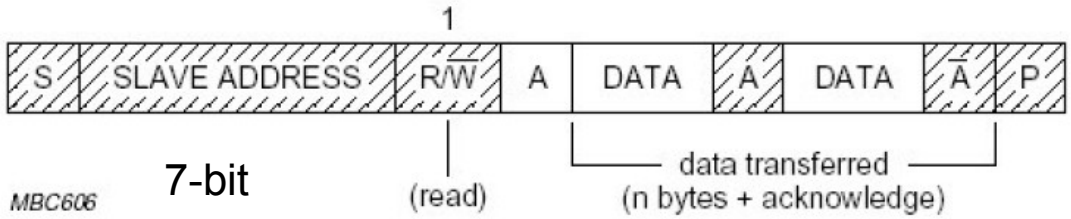
master-transmitter





## Odczyt n-bajtów danych

master-receiver (since second byte)



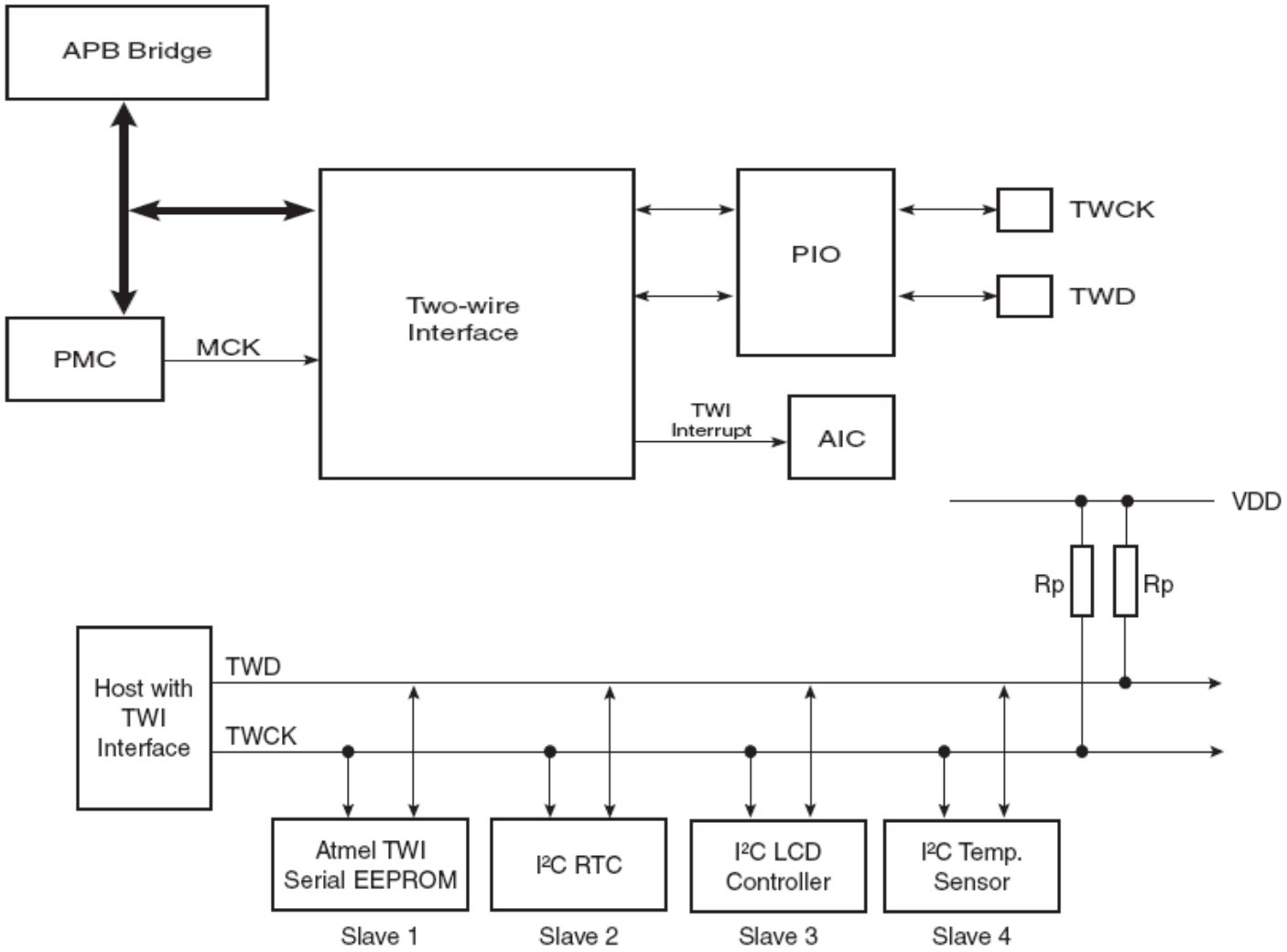
# Two-Wire Interface – standard zgodny z I2C ?

Moduł TWI procesorów ARM jest odpowiednikiem standardu opracowanego przez firmę Philips (firma Philips posiada patent na interfejs I2C).

## Cechy interfejsu SWI procesora AMR firmy ATMEL:

- Zgodny ze standardem I2C,
- Praca w trybie Master, Multimaster lub Slave,
- Umożliwia dołączenie urządzeń zasilanych napięciem 3,3 V,
- Transmisja danych z częstotliwością zegara do 400 kHz,
- Transfery poszczególnych bajtów wyzwalane przerwaniami,
- Automatycznie przejście do trybu Slave w przypadku kolizji na magistrali (Arbitration-lost interrupt),
- Przerwanie zgłaszane, gdy zostanie wykryty adres urządzenia w trybie Slave,
- Automatyczne wykrywanie stanu zajętością magistrali,
- Obsługa adresów 7 i 10-cio bitowych.

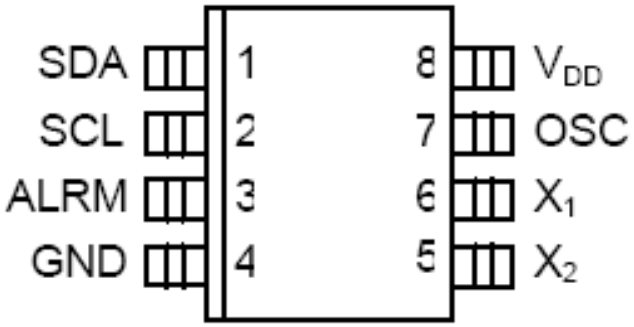
# Schemat blokowy modułu TWI



# Zegar czasu rzeczywistego

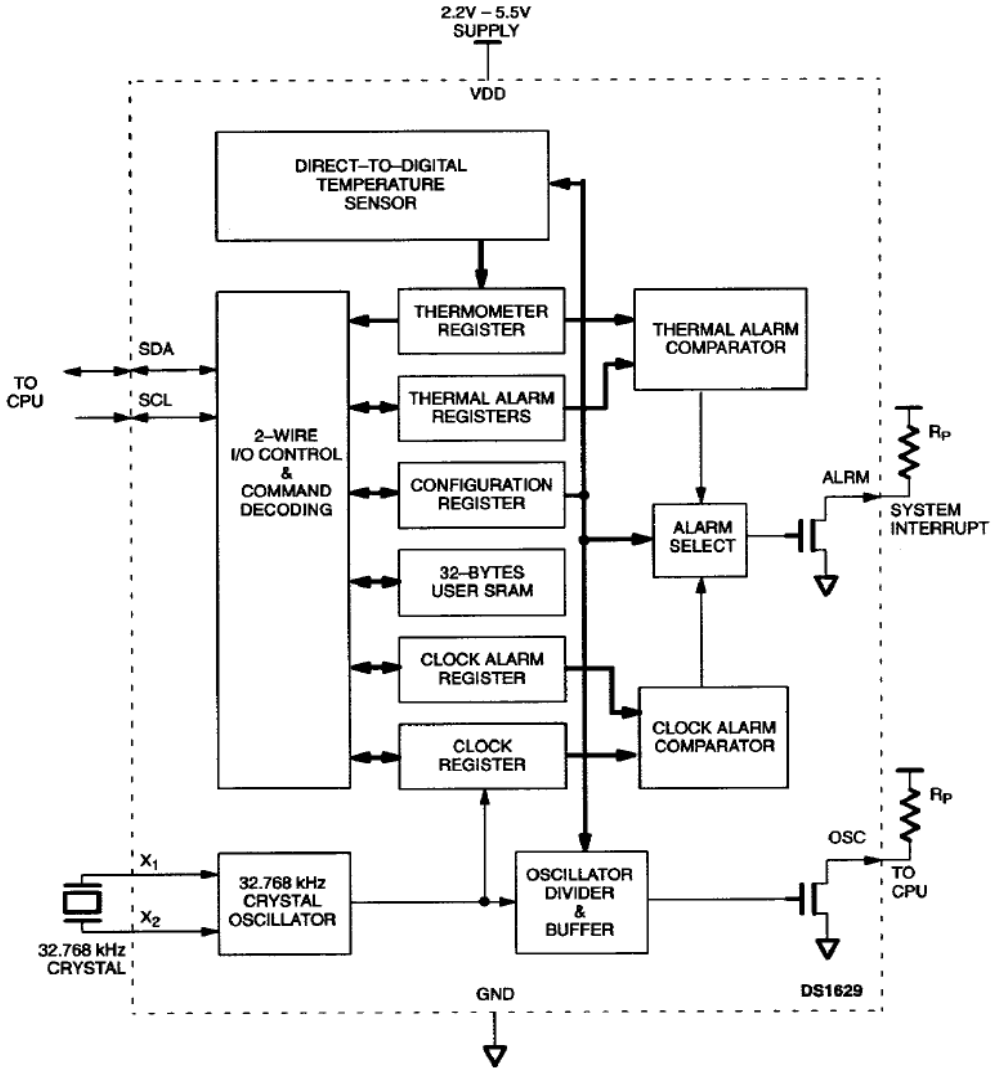
## Cechy układu DS1629:

- Zegar czasu rzeczywistego,
- Pomiar temperatury -55 – 125 C,
- Rozdzielczość termometru: 9 bitów,
- Dokładność termometru +/- 2 C,
- Układ termostatu,
- 32 bajty pamięci SRAM,
- Zasilanie 2,2 – 5,5 V,
- Interfejs zgodny ze standardem I2C (400 kHz).

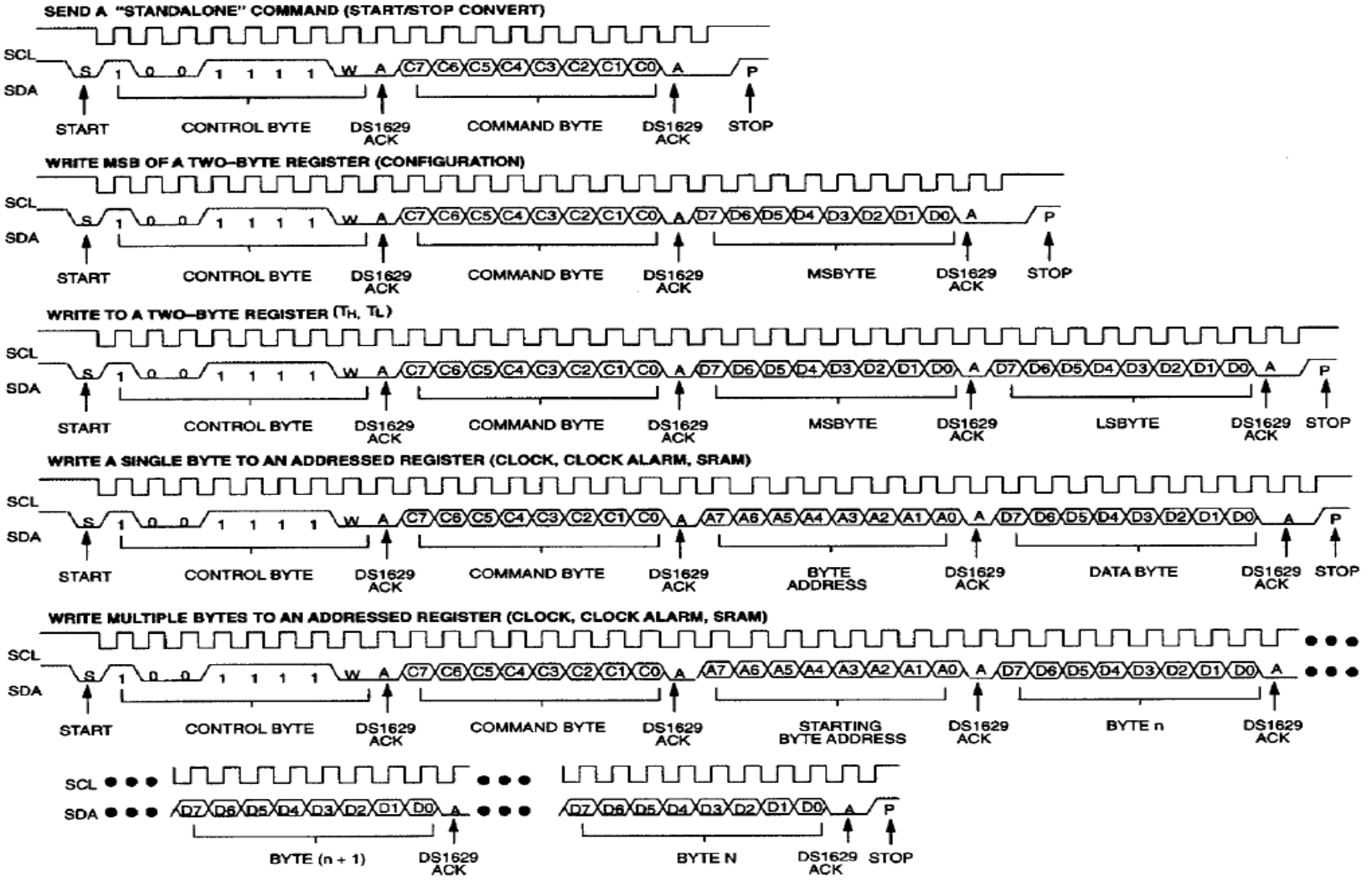



DS1629S 8-Pin SOIC
(150-mil)

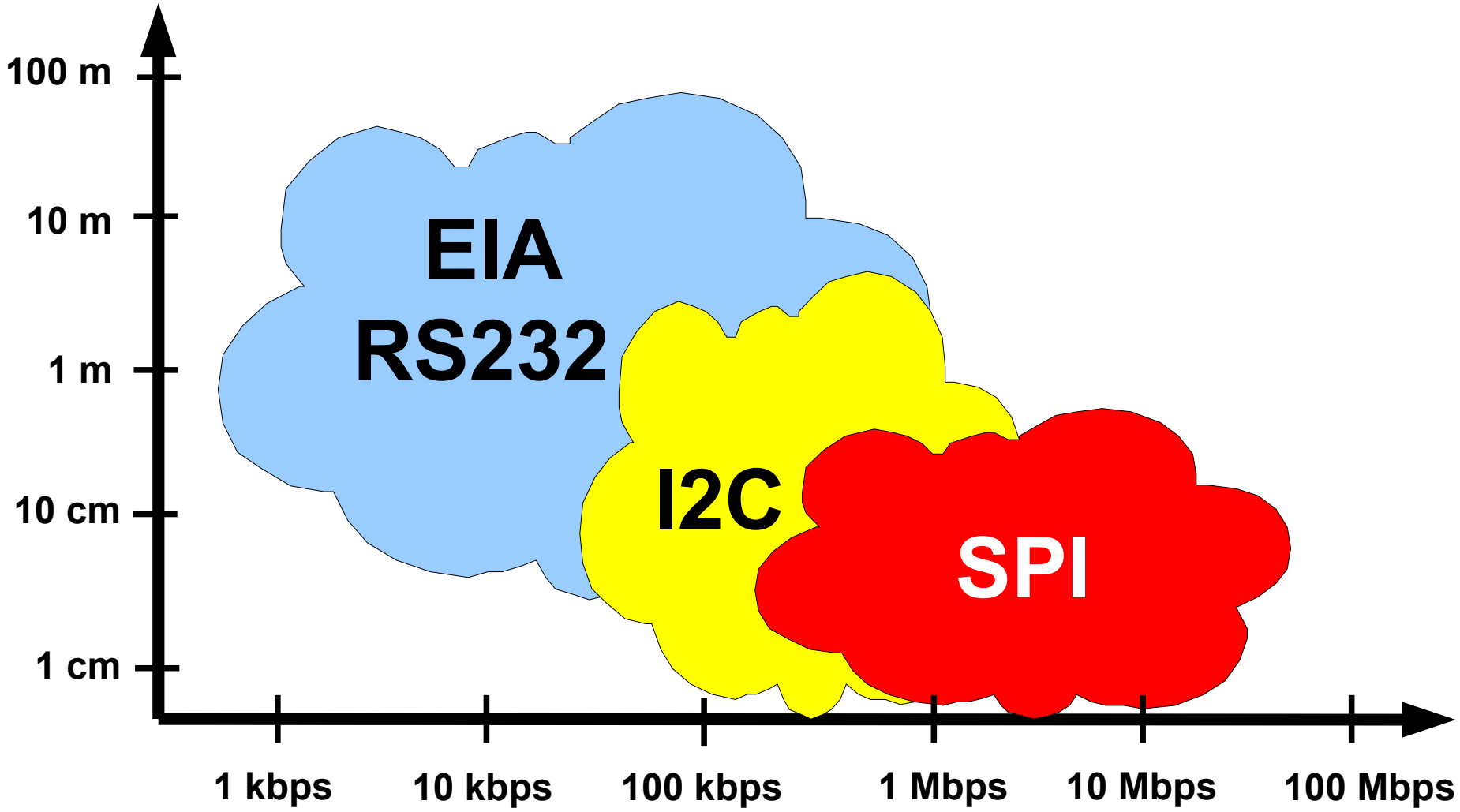
# Zegar czasu rzeczywistego



# Transmisja z wykorzystaniem interfejsu I2C



# Interfejsy szeregowe - podsumowanie



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

# „Pamięci i urządzenia peryferyjne”
# „Wprowadzenie do przedmiotu”

Prezentacja jest współfinansowana przez
Unię Europejską w ramach
Europejskiego Funduszu Społecznego w projekcie pt.

*„Innowacyjna dydaktyka bez ograniczeń - zintegrowany rozwój Politechniki Łódzkiej - zarządzanie Uczelnią, nowoczesna oferta edukacyjna i wzmacniania zdolności do zatrudniania osób niepełnosprawnych”*

Prezentacja dystrybuowana jest bezpłatnie