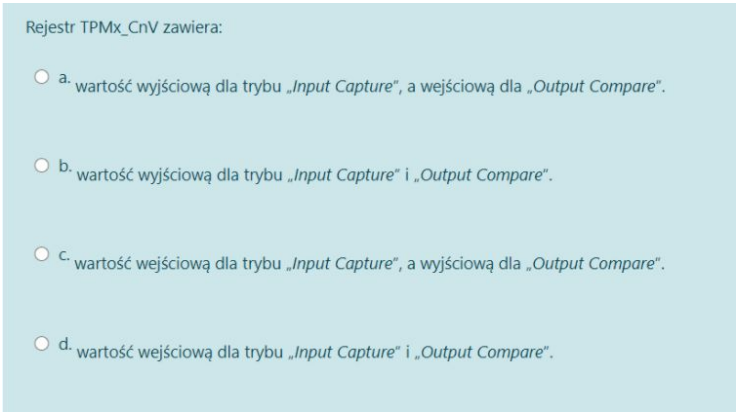
1. Rejestr TPMx_CNV zawiera

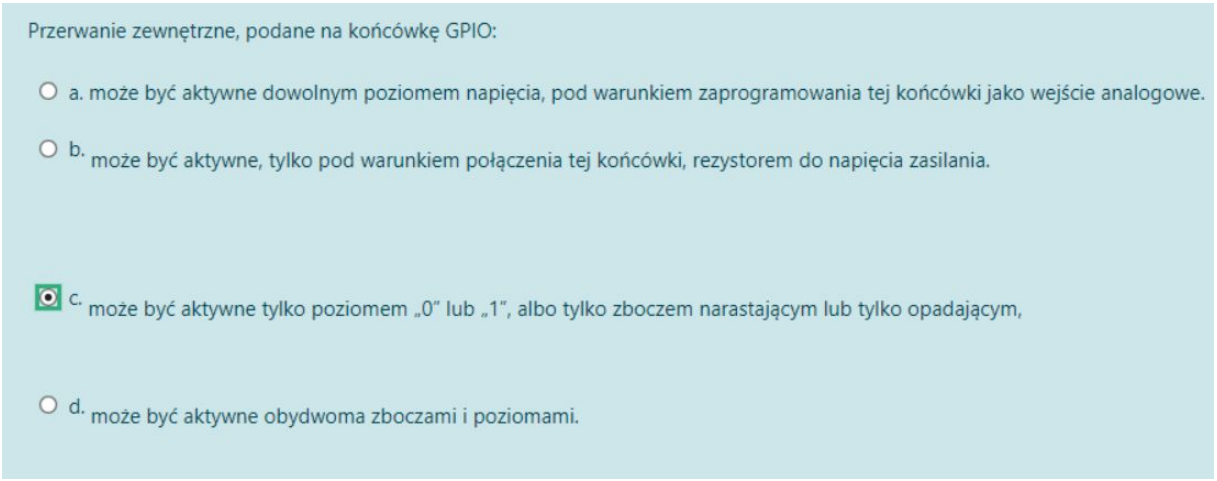


„Input Capture" - sygnał podany na wejście „channel N input" lub pochodzący z wyjścia komparatora CMP0 (tylko TPM1, ustawiane w SIM_SOPT4[TPM1CH0SRC]), zapamiętuje („zatrzaskuje"), aktualny stan licznika głównego, w rejestrze CNV. Aktywne może być

„Output Compare" - następuje ciągłe porównywanie zawartości licznika głównego z wcześniej zapisaną zawartością rejestru CNV. W momencie, gdy obydwie wartości się

rejestr zawiera wartość wyjściową dla input capture i wartość wejściową dla output compare

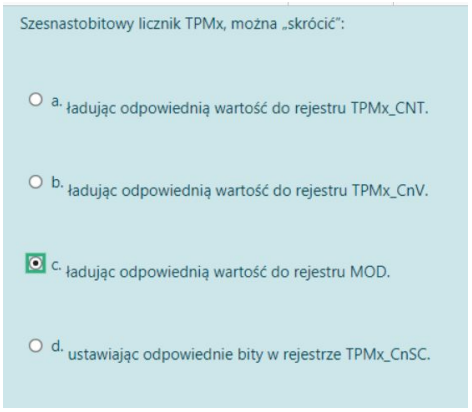
2. Przerwanie zewnętrzne, podane na końcówkę GPIO:



może być aktywne obydwoma zboczami (rising, falling, either) i poziomami (logic zero, logic one) - niżej tabelka z manuala

Reserved	This read-only field is reserved and always has the value 0.
19–16 IRQC	Interrupt Configuration This field is read only for pins that do not support interrupt generation. The pin interrupt configuration is valid in all digital pin muxing modes. The corresponding pin is configured to generate interrupt/DMA request as follows: 0000 Interrupt/DMA request disabled. 0001 DMA request on rising edge. 0010 DMA request on falling edge. 0011 DMA request on either edge. 1000 Interrupt when logic zero. 1001 Interrupt on rising edge. 1010 Interrupt on falling edge. 1011 Interrupt on either edge. 1100 Interrupt when logic one. Others Reserved.
15–11	This field is reserved.

3. Szesnastobitowy licznik TPMx można skrócić



Rejestr MOD służy do „skracania" 16-bitowego licznika głównego.

Ładując odpowiednią wartość do rejestru MOD

4.. Port szeregowy UART0 pracuje w trybie

Port szeregowy UART0 pracuje w trybie:

a. quasi-chronicznym.

b. mieszanym.

c. synchronicznym.

d. asynchronicznym.

Rozwinięcie skrótu UART to universal asynchronous receiver-transmiter więc odpowiedziałbym że asynchronicznym

5. Po wykonaniu następującego programu, zmienna z będzie miała wartość:

Po wykonaniu następującego programu, zmienna „z" będzie miała wartość:
z=0x55;
w=0xaa;
z=z||w;
z=z|w;

a. 0xff

b. 0xab

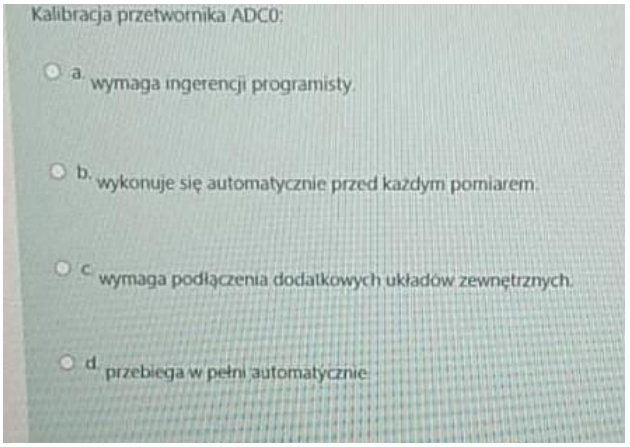
c. 1

d. 0xaa

Z = 0x55 || 0xaa i z tego wyjdzie 1, bo jest logiczny or, a nie bitowy

Potem z = 0x1 | 0xaa i wynik tego bitowego ora wynosi 0xab

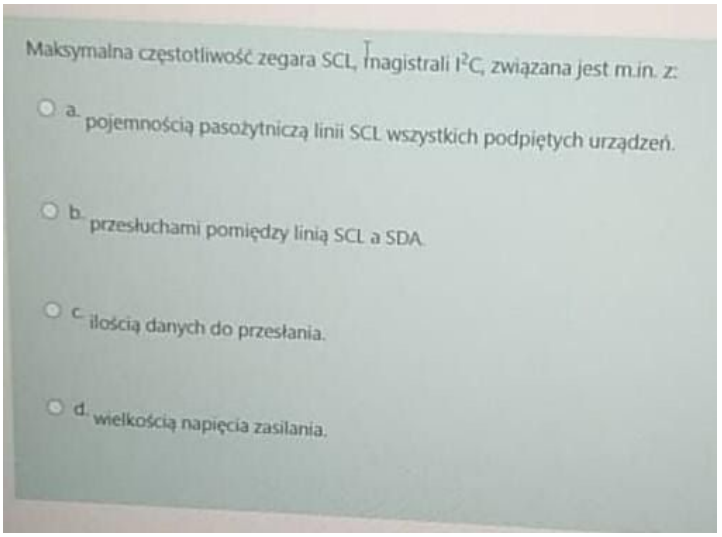
6. Kalibracja przetwornika ADC0



rozpocząć kalibrację, poprzez ustawienie bitu CAL na wartość 1, w rejestrze ADC0->SC3.

Wymaga ingerencji programisty

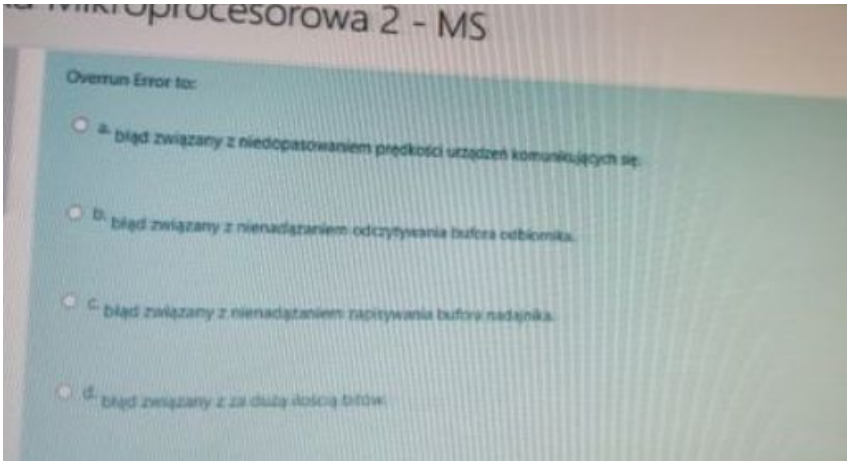
7. Maksymalna częstotliwość zegara SCL, magistrali I2C związana jest m. in. z:



Jak jest dużo urządzeń to przez ich rezystory podciągające ładują się pojemności pasożytnicze, one są równolegle do siebie więc pojemność wypadkowa się zwiększa i wtedy zamiast przebiegu prostokątnego może zrobić się piła nie dochodząca do stanu aktywnego (z labów)

Pojemnością pasożytniczą linii SCL wszystkich podpiętych urządzeń

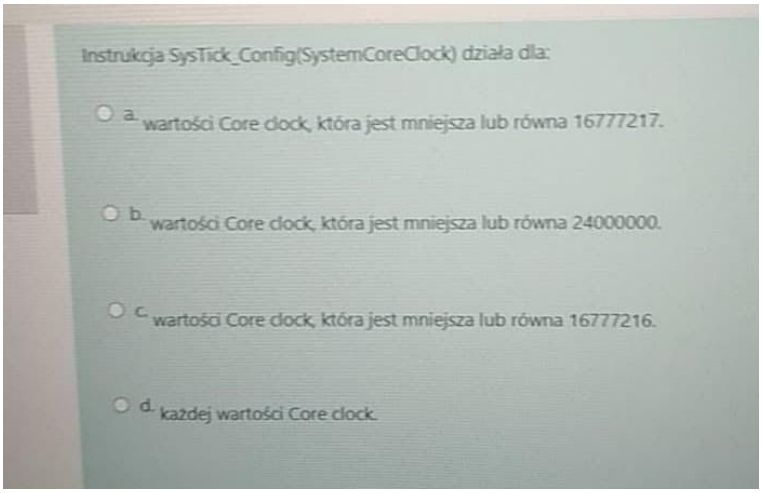
8. Overrun error to



Overrun Error - brak możliwości zapisania do bufora odbiornika nowej danej (skompletowanej w odbiorniku), spowodowany nieodczytaniem poprzedniej wartości z bufora,

Błąd związany z nienadążaniem odczytywania bufora odbiornika (bo odbiornik nie zdążył odczytać poprzedniej wartości z bufora, a chce mu się zapisać nową daną i wtedy występuje overrun error)

9. Instrukcja SysTick_Config(SystemCoreClock) działa dla:



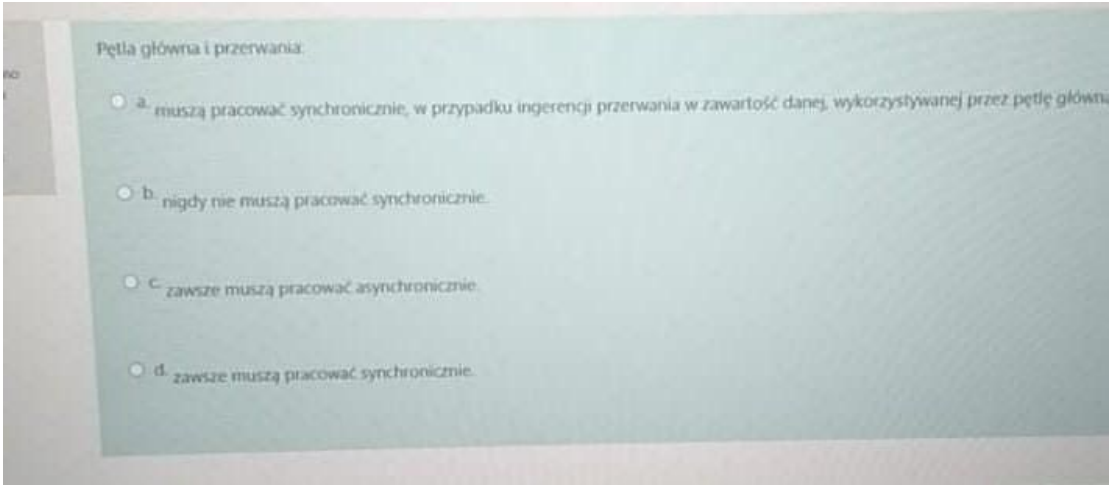
Z prezentacji co była na zajęciach:

Zegar SysTick jest elementem jądra procesora. Taktowany jest zegarem systemowym SystemCoreClock (patrz system_MKL05Z4.c). Zlicza w tył, od wartości wcześniej załadowanej. Jego pojemność to 24 bity, tzn., że maksymalna liczba zliczonych taktów zegarowych wynosi 16777216. Zawsze ładuje się wartość początkową o jeden mniejszą. Przykład: jeśli chcemy, aby zostało zliczonych 100 taktów zegara, musimy załadować wartość początkową 99. Funkcja SysTick_Config(uint32_t ticks), zdefiniowana w zbiorze core_cm0plus.h zapewnia już powyższą modyfikację. Należy tylko zadbać, aby argument tej funkcji nie przekraczał 24-ech bitów, ponieważ zegar nie wystartuje, a funkcja zwróci kod błędu równy 1. Nie ma bezpośredniej funkcji zatrzymującej ten zegar. Aby go zatrzymać, należy wyzerować bit ENABLE w rejestrze Systick->CTRL.

[AS] W zasadzie wydaje mi się że można wpisać tylko wartość mniejszą niż 16 777 216, bo jak byśmy wpisali dokładnie 16 777 216 to to już jest 25 bitów no ale takiej odpowiedzi nie ma więc pewnie ma być 16 777 216.

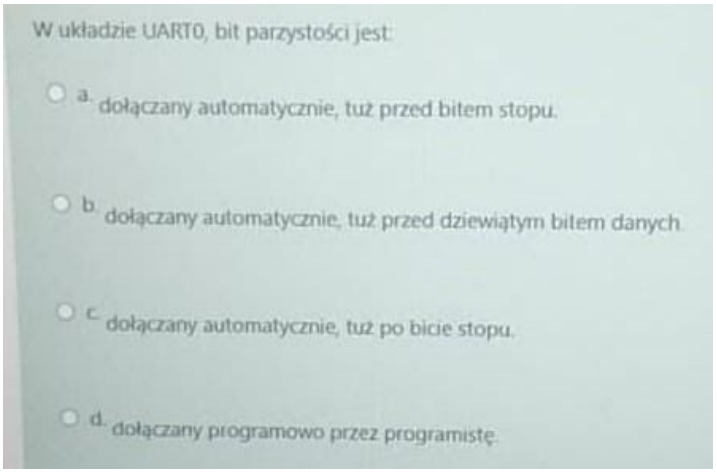
To z tego wynika że 16 777 216

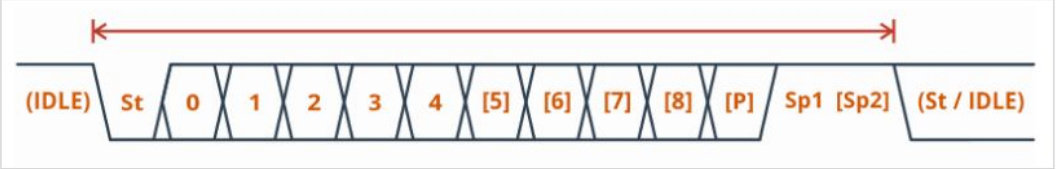
10.. Pętla główna i przerwania



Muszą pracować synchronicznie, w przypadku ingerencji przerwania w zawartość danej, wykorzystywanej przez pętlę główną

11. W układzie UART0 bit parzystości jest





Ramka UART.

- **Stan bezczynności (Idle)** - oznacza, że aktualnie nie przebiega żadna transmisja. Komunikowany jest przez wymuszenie **stanu wysokiego** na linii danych.
- **Bit startu (St)** - rozpoczyna każdą ramkę danych. W tym wypadku jest to wymuszenie stanu niskiego na linii danych. Bit startu jest obowiązkowym elementem każdej ramki.
- **Bity danych (0-8)** - bity danych reprezentują aktualne informacje, które chcemy przesłać. Standardowo przesyła się ramki złożone z 8 bitów danych. W teorii może być ich od 5 do 9.
- **Bit parzystości (P)** - służy jako podstawowa, niskopoziomowa forma sprawdzenia poprawności przesłanych danych. Jest on opcjonalny i jego obecność jest jednym z konfigurowalnych parametrów transmisji. Bit parzystości może działać w dwóch trybach:
    1. **Parzysty** - celem jest uzyskanie parzystej sumy bitów danych i bitu parzystości. Jeżeli liczba "jedynek" w bitach danych była nieparzysta, bit przyjmie wartość 1. Jeżeli liczba ta była parzysta, bit parzystości przyjmie wartość 0.
    2. **Nieparzysty** - analogicznie do trybu parzystego. Bit parzystości zawsze przyjmie taką wartość, aby sumarycznie liczba bitów w stanie wysokim była liczbą nieparzystą.
- **Bit stopu (Sp1, Sp2)** - bit oznaczający zakończenie całej ramki danych. Komunikowany jest jako wymuszenie stanu wysokiego na linii danych. Liczba bitów stopu jest konfigurowalna i może wynosić 1 lub 2.

Bit parzystości jest dołączany programowo przez programistę

12. Po przyjęciu przerwania od końcówki PTAx, pierwszą czynnością podjętą przez program obsługi to:

Po przyjęciu przerwania od końcówki PTAx, pierwszą czynnością podjętą przez program obsługi to:

- a. sprawdzenie stanu bitów w rejestrze PORTA_ISFR.
- b. skasowanie bitu ISFx, w rejestrze PORTA_ISFR.
- c. odczytanie stanu portu A.
- d. wykonanie instrukcji *NVIC_ClearPendingIRQ(PORTA_IRQn)*.

PORTx_ISFR field descriptions

Field	Description
31–0 ISF	Interrupt Status Flag Each bit in the field indicates the detection of the configured interrupt of the same number as the field. 0 Configured interrupt is not detected. 1 Configured interrupt is detected. If the pin is configured to generate a DMA request, then the corresponding flag will be cleared automatically at the completion of the requested DMA transfer. Otherwise, the flag remains set until a logic one is written to the flag. If the pin is configured for a level sensitive interrupt and the pin remains asserted, then the flag is set again immediately after it is cleared.

```
void PORTB_IRQHandler(void){
if( PORTB->ISFR & (1 << BUT1) ){      /* Check in ISFR register if button BUT1 is pressed */
        fsmFastSlow();                /* handle interrupt */
        PORTB->PCR[BUT1] |= PORT_PCR_ISF_MASK; /* clear interrupt service flag */
        }
}
```

Handler najpierw sprawdza od którego pinu przyszło przerwanie w rejestrze ISF, potem wykonuje obsługę przerwania a na końcu czyści flagę w ISF. Czyli byłoby sprawdzenie stanu bitów w rejestrze PORTA_ISFR.

13. Port B pracuje jako wyjście. Poniższa instrukcja:

Port B pracuje jako wyjście. Poniższa instrukcja:

PTB->PCOR &= 0xffff0000

a. szesnaście najmłodszych bitów ustawi w stan 0, a resztę w stan 1 .

b. szesnaście najstarszych bitów ustawi w stan 0, a reszta pozostanie bez zmian.

c. pozostawi bez zmian stan wszystkich bitów portu B.

d. szesnaście najmłodszych bitów ustawi w stan 0, a reszta pozostanie bez zmian.

Clear (to 0):
Write 1 to PCOR

WSZYSTKIE BITY POZOSTANĄ BEZ ZMIAN !!!! ODP C , mamy tutaj & więc będziemy miec same 0 więc nic się nie zmieni, gdyby zamiast & było | to faktycznie poprawną odpowiedzią byłoby B

14. Wydłużenie czasu próbkowania:

Wydłużenie czasu próbkowania:

a. wpływa na zmniejszenie się dokładności pomiaru źródeł o małej rezystancji wewnętrznej.

b. nie wpływa na szybkość pomiaru.

c. pozwala zwiększyć szybkość przetwarzania.

d. pozwala na dokładniejsze pomiary źródeł o dużej rezystancji wewnętrznej.

Mówił na zajęciach coś takiego, że jak jest układ sample and hold po źródle napięciowym, to jak źródło napięciowe będzie miało za dużą rezystancję wewnętrzną, to kondensator (który jest częścią układu sample and hold) nie naładuje się odpowiednio. Z tego wynikałoby że pozwala na dokładniejsze pomiary źródeł o dużej rezystancji wewnętrznej

15. Do komunikacji terminalowej (z komputerem):

Do komunikacji terminalowej (z komputerem):

a. używa się kodu ASCII, ponieważ część wartości bajtowych to znaki sterujące.

b. używa się dziewięciobitowych danych, gdzie dziewiąty bit decyduje czy jest to dana czy znak sterujący.

c. można używać dowolnych wartości bajtowych.

d. używa się dowolnych wartości bajtowych, a komputer sam wie, które z nich to znaki sterujące.

Używa się kodu ascii, ponieważ część wartości bajtowych to znaki sterujące

Takie znaki sterujące to np LF('\n') i CF('\r')

16. Aby dwa liczniki PIT, połączone łańcuchowo, doliczyły do 100 000 000 000, do poszczególnych rejestrów PIT_LDVALx należy wpisać:

Aby dwa liczniki PIT, połączone łańcuchowo, doliczyły do 100 000 000 000, do poszczególnych rejestrów PIT_LDVALx należy wpisać:

a. PIT_LDVAL1=10 000 000 000 i PIT_LDVAL2=10.

b. PIT_LDVAL1=10 000 000 000 i PIT_LDVAL2=9.

c. PIT_LDVAL1=999 999 999 i PIT_LDVAL2=99.

d. PIT_LDVAL1=9 999 999 999 i PIT_LDVAL2=9.

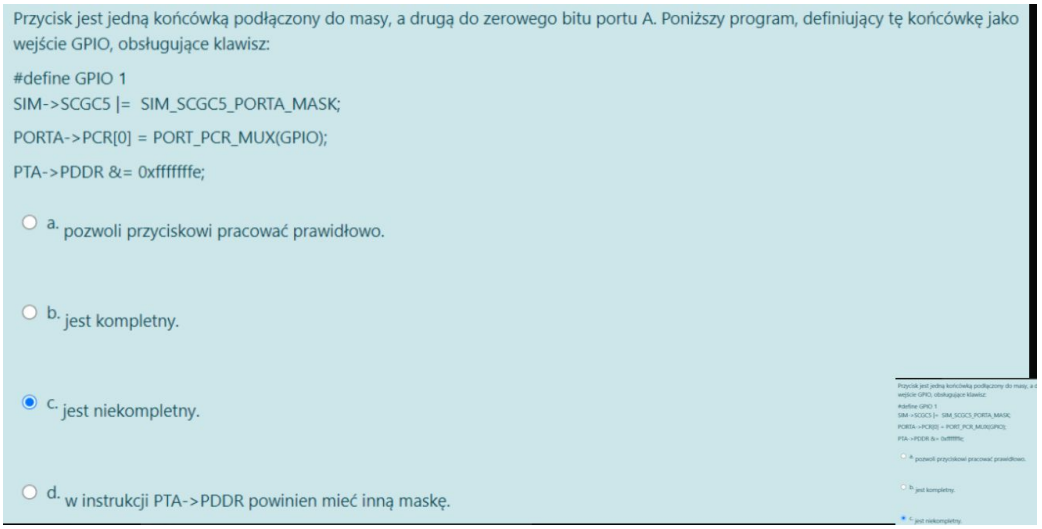
LDVAL – rejestr 32-bitowy w którym można ustawić okres generowania zdarzeń

$$T = \frac{LDVAL - 1}{freq_{bus\ clock}}$$

Czyli do LDVAL możemy załadować wartość maksymalnie 2^32 -1 = 4 294 967 295. To oznacza że odpowiedzi A B i D nam odpadają (bo po prostu nie możemy załadować takich dużych wartości do PIT_LDVAL1). Nie wiem czy to co jest w odpowiedzi C doliczy do tego co chcemy, ale to jest jedyna opcja jaką w ogóle da się zrealizować.

PIT_LDVAL1 = 999 999 999 i PIT_LDVAL2 = 99

17. Przycisk jest jedną końcówką podłączony do masy, a drugą do zerowego bitu portu A. Poniższy program, definiujący tę końcówkę jako wejście GPIO, obsługujące klawisz:



Maska jest ok – bo jak się zrobi coś takiego

```
unsigned int pta = 0;
pta &= 0xfffffffe;
cout << pta << endl;
return 0;
```

To wynik pta = 0 – czyli jest ustawione jako wejście.

3.10.1.2 Port Control and Interrupt Summary

The following table provides more information regarding the Port Control and Interrupt configurations .

Table 3-47. Ports Summary

Feature	Port A	Port B
Pull Select control	No	No
Pull Select at reset	PTA0=Pull down, Others=Pull up	Pull up
Pull Enable control	Yes	Yes
Pull Enable at reset	PTA0/PTA2/RESET_b=Enabled; Others=Disabled	PTB5=Enabled; Others=Disabled

Jak był testowany to działa ale te odpowiedzi są serio niejednoznaczne i nie wiadomo co zaznaczyć

18. Wykorzystanie trybu ciągłego Multiple-Byte Read w przypadku odczytu z wyświetlacza LCD, podłączonego do ekspandera I2C PCF8574

Wykorzystanie trybu ciągłego „*Multiple-Byte Read*", w przypadku odczytu z wyświetlacza LCD, podłączonego do ekspandera I^2C PCF8574:

a. jest możliwe tylko wtedy, gdy odpowiednie bity ekspandera zostaną zdefiniowane jako wyjścia.

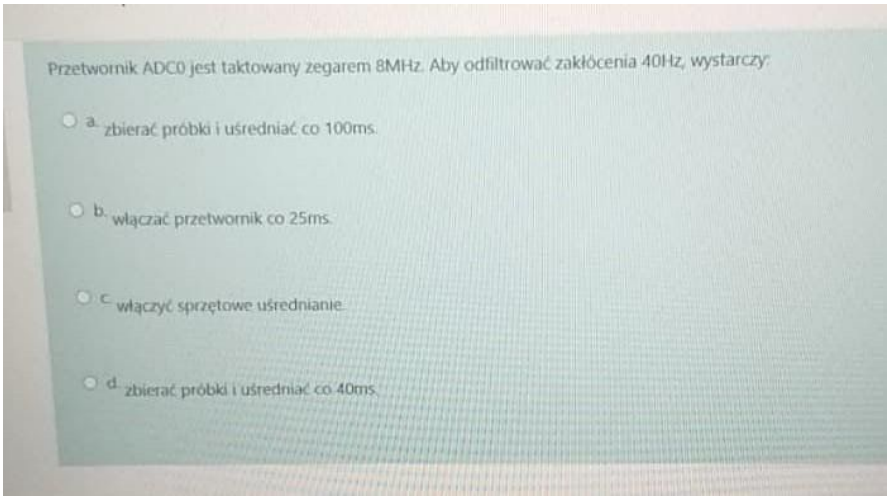
b. jest w ogóle niemożliwe.

c. jest zawsze możliwe.

d. jest możliwe tylko wtedy, gdy odpowiednie bity ekspandera zostaną zdefiniowane jako wejścia.

jest zawsze możliwe

19. Przetwornik ADC0 jest taktowany zegarem 8 MHz. Aby odfiltrować zakłócenia 40 Hz, wystarczy:



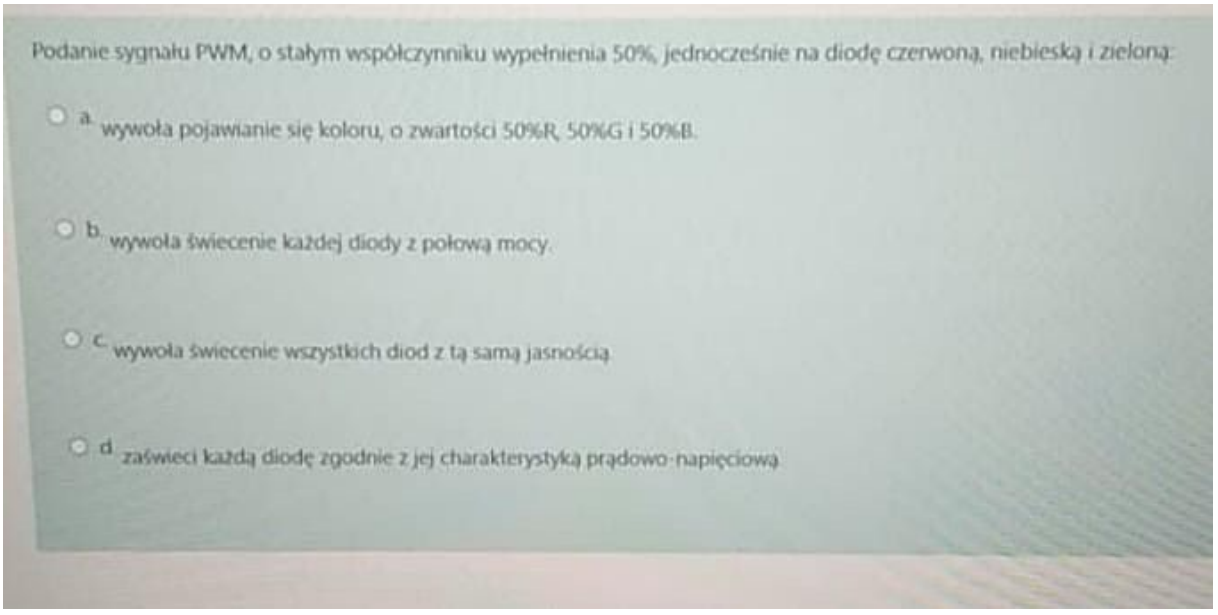
zbierać próbki i uśredniać co 100ms

Jak było zadanie 1 z ćwiczenia 5, to ten program (przed tym jak napisaliśmy uśrednianie), wariował po tym jak zaświeciliśmy na niego żarówką zasilaną napięciem o częstotliwości 50 Hz (okres 20 ms). Uśrednianie robiliśmy co 100 ms czyli 5 okresów sieci.

To teraz jak mamy zakłócenia 40 Hz (okres 25 ms) to możliwe że trzeba zrobić to samo – wziąć uśrednianie co 100 ms czyli 4 okresy zakłóceń.

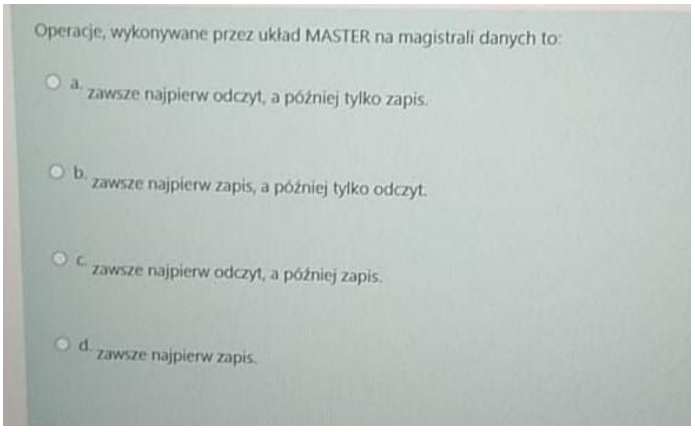
Inne odpowiedzi mi nie pasują – włączanie przetwornika co 25 ms to totalnie bez sensu, sprzętowe uśrednianie włącza się tak czy tak ale to nie wystarczy żeby odfiltrować zakłócenia, a z uśrednianiem 40 ms to też bez sensu bo byśmy brali niepełny okres zakłóceń.

20. Podanie sygnału PWM o stałym współczynniku wypełnienia 50 % jednocześnie na diodę czerwoną, niebieską i zieloną



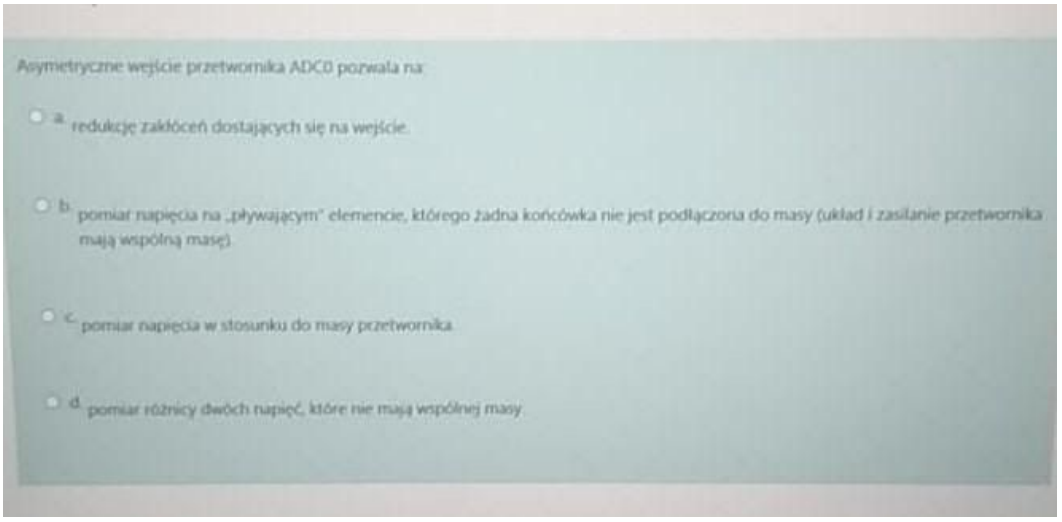
Połowa mocy

21. Operacje wykonywane przez układ MASTER na magistrali danych to:



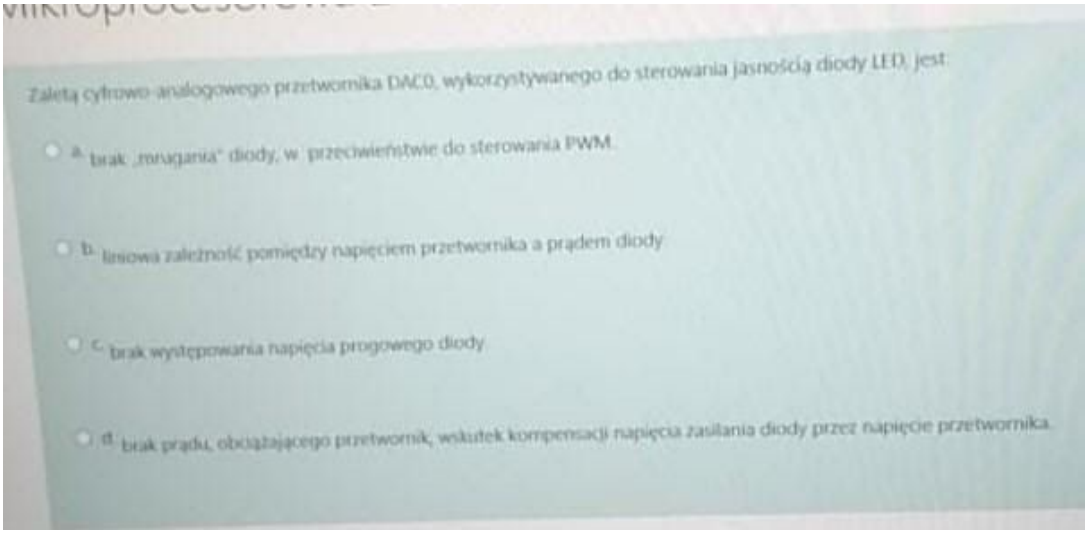
Zawsze najpierw zapis - bo master musi odczytać adres urządzenia slave

22. Asymetryczne wejście przetwornika ADC0 pozwala na:



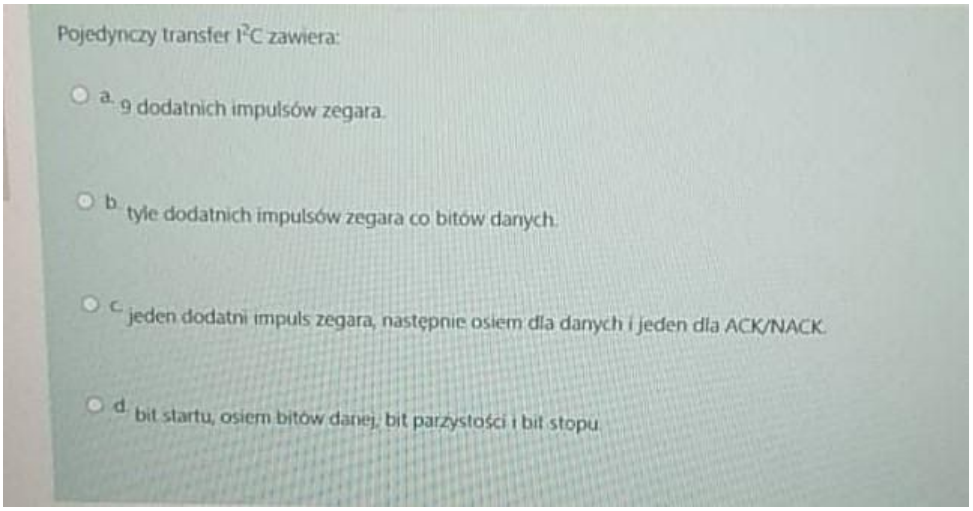
Pomiar napięcia w stosunku do masy przetwornika

23. Zaletą cyfrowo-analogowego przetwornika DAC0 wykorzystywanego do sterowania jasnością diody LED jest



Brak “mrugania” diody

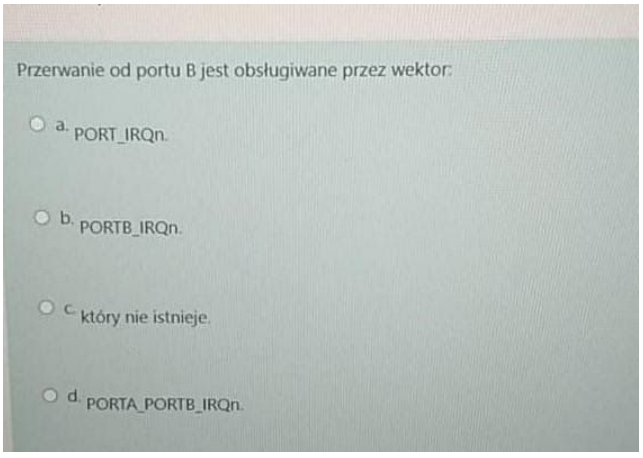
24. Pojedynczy transfer I2C zawiera:



9 dodatnich impulsów zegara

Dane na magistrali I2C są przesyłane w 8-bitowych pakietach. Nie ma ograniczenia liczby bajtów, jednakże po każdym bajcie musi następować bit potwierdzenia (ACK). Ten bit sygnalizuje, czy urządzenie jest gotowe do przejścia do następnego pakietu. Dla wszystkich bitów danych, w tym bitu potwierdzenia, master musi generować impulsy zegarowe. Jeśli urządzenie podrzędne nie potwierdza przesyłania, oznacza to, że nie ma więcej danych lub urządzenie nie jest jeszcze gotowe do przesyłania.

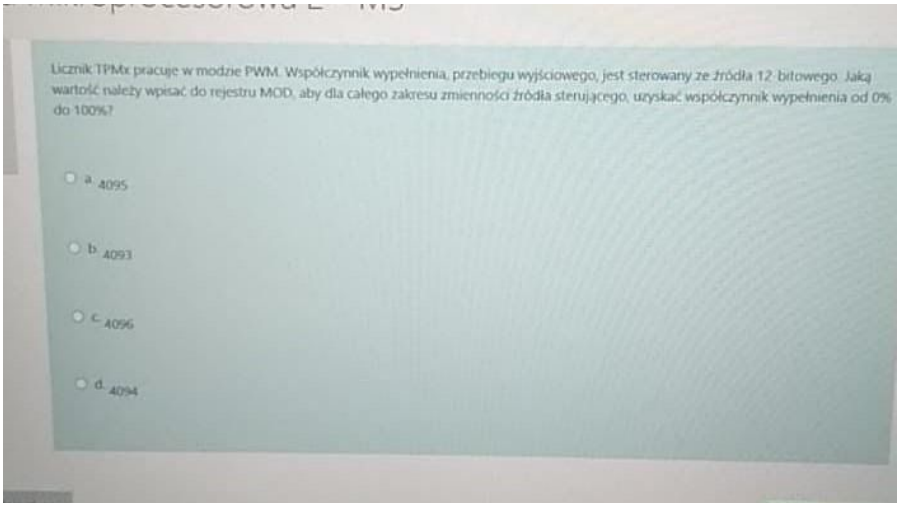
25. Przerwanie od portu B jest obsługiwane przez wektor




```
PORTB_IRQn                = 31                /**< Port B interrupt */
```

PORTB_IRQn

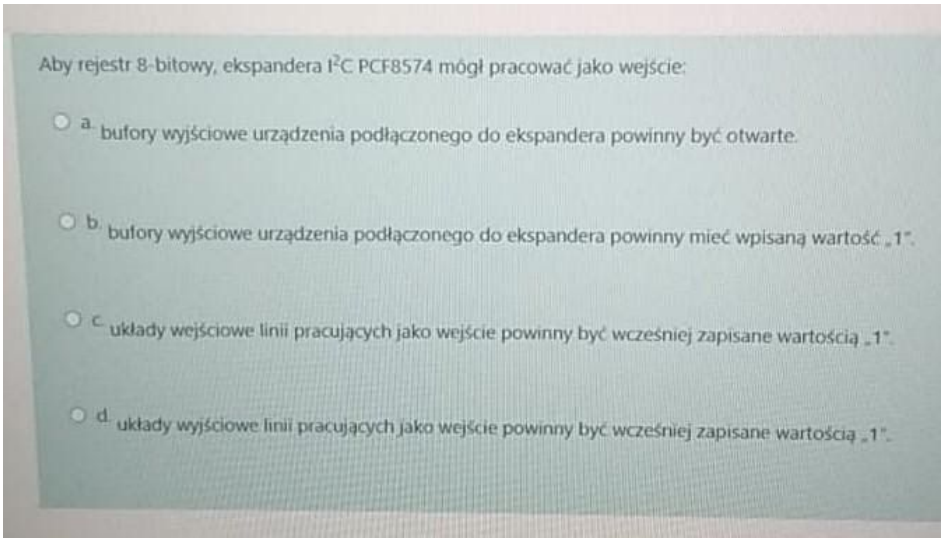
26. Licznik TPMx pracuje w modzie PWM. Współczynnik wypełnienia przebiegu wejściowego jest sterowany ze źródła 12-bitowego. Jaką wartość należy wpisać do rejestru MOD, aby dla całego zakresu zmienności źródła sterującego uzyskać współczynnik wypełnienia od 0 % do 100 %?



jest zmniejszana o 2, co daje przedział zmian 0÷95. Proszę zwrócić uwagę na ustawienie rejestru MOD, licznika TPM0, w zbiorze *TPM.c*. Ma on wartość 94, czyli o 1 mniej niż zakres zmian. Tę zasadę stosuje się, w celu umożliwienia osiągnięcia współczynnika wypełnienia

Czyli do MOD wpisujemy 4094

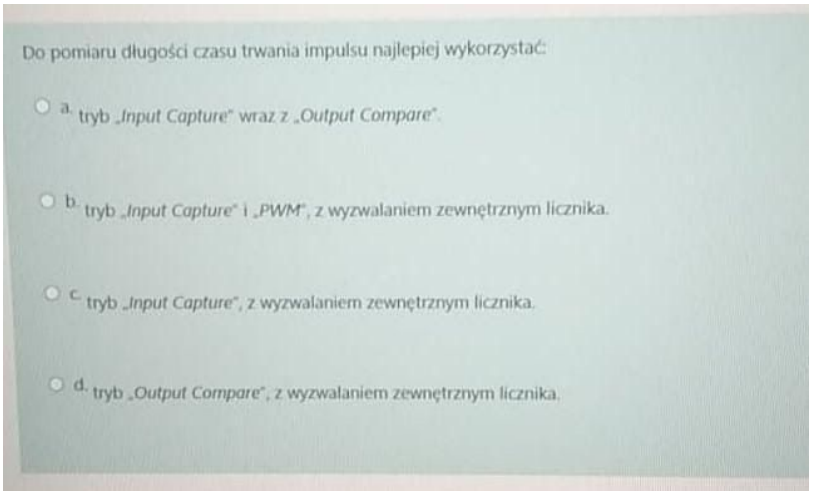
27. Aby rejestr 8-bitowy ekspandera I2C PCF8574 mógł pracować jako wejście



d układy wyjściowe linii pracujących jako wejście(?)

Port P jest pseudo dwukierunkowy (quasi-bidirectional – nie jest potrzebny rejestr kierunku, wpisanie 1 oznacza wejście, wpisanie 0 oznacza wyjście). Aby dana końcówka mogła pracować jako wejście, jej bufor wyjściowy musi być zapisany wartością „1". Wejście Px nie może „walczyć" z sygnałem wejściowym.

28. Do pomiaru długości czasu trwania impulsu najlepiej wykorzystać



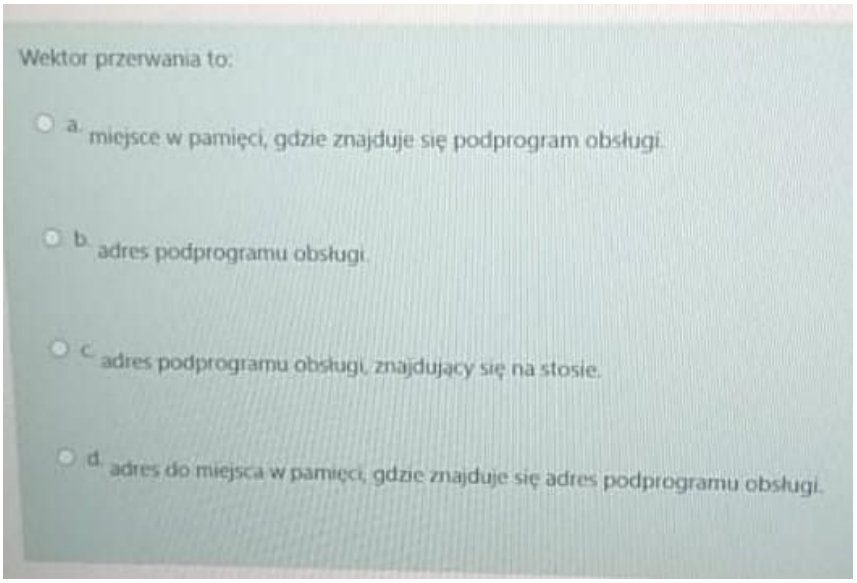
2.2. INPUT CAPTURE

Tryb ten pozwala na przechwytywanie „w locie" aktualnej zawartości licznika głównego. W niniejszym ćwiczeniu skupimy się na wyzwalaniu przechwytywania przez impulsy zewnętrzne, podawane na wejście kanału nr 1 licznika TPM1. W przedstawianym przykładzie, impulsy będą pochodzić z wyjścia kanału nr 0 licznika TPM0. Kanał ten będzie pracował w trybie PWM („High-true pulses"), z pozycjonowaniem impulsu względem początku okresu. Taka konfiguracja pozwoli nam na stworzenie środowiska do pomiaru czasu trwania impulsu PWM i nie tylko. Pomiar będzie polegał na zliczaniu impulsów

Tu w ćwiczeniu było że jeden kanał licznika pracuje w trybie input capture, a drugi kanał pracuje w trybie PWM, ale wydaje mi się że w ogólności impuls nie musi pochodzić z PWMa. Dlatego odpowiedziałbym że tryb input capture z wyzwalaniem zewnętrznym licznika

[AS] Potem mierzyliśmy w tym ćwiczeniu czas wykonywania instrukcji sprintf czy jakiejśtam innej i już nie używaliśmy PWM tylko wyzwalaliśmy sami licznik. Więc tym bardziej odp. C.

29. Wektor przerwania to:

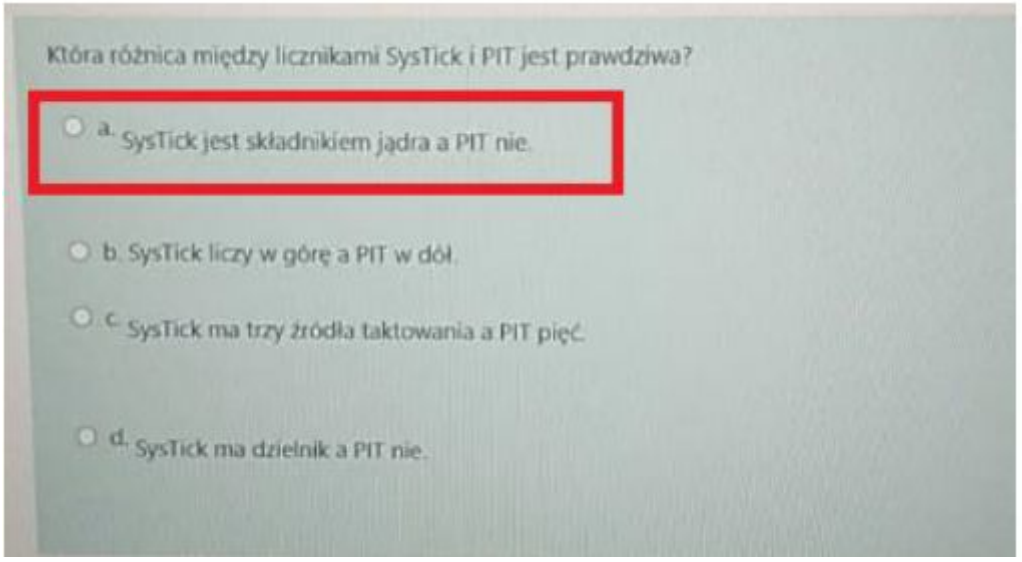


On powiedział na zajęciach że to są adresy podprogramu obsługi przerwań/wyjątków

Adres podprogramu obsługi

Tablica wektorow = {0xff01,0xf02,.....} <-adres początku podprogramu

30. Która różnica pomiędzy licznikami SysTick i PIT jest prawdziwa?



RGB LED. W przeciwieństwie do SysTick, który jest częścią rdzenia przetwarzającego ARM, zegar PIT jest modułem peryferyjnym (jak GPIO). Podobnie jak SysTick, PIT jest prostym

Systick jest częścią rdzenia procesora, a PIT jest modułem peryferyjnym (jak GPIO)