Akademia Górniczo-Hutnicza
w Krakowie
Katedra Elektroniki
WIET

Laboratorium
Techniki Mikroprocesorowej 2

Ćwiczenie 7

Moduł I2C -
sposób na komunikację między układami

Autorzy: Sebastian Koryciak

ver. 26.11.2020

1. Cel ćwiczenia

Celem tego ćwiczenia, jest zaznajomienie uczestników zarówno z protokołem do komunikacji między układami, jak i modułem realizującym go dostępnym w mikrokontrolerze Kinetis KL05Z. Do realizacji tego celu posłużą przykłady wykorzystania tego modułu do odpytywania jakie urządzenia podłączone są do magistrali oraz wysyłania i odbierania danych od układu akcelerometru znajdującego się na platformie FRDM-KL05Z oraz ekspandera wejść/wyjść.

1.1. Wymagania

- Komputer PC z zainstalowanym środowiskiem Keil uVision
- Zestaw uruchomieniowy FRDM-KL05Z
- Znajomość podstaw programowania układu FRDM-KL05Z w języku C

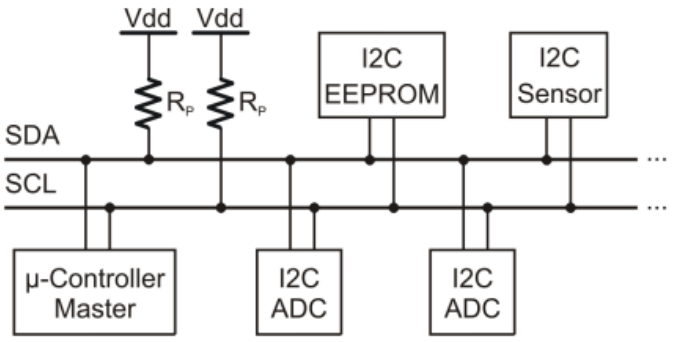
1.2. Literatura

- **I2C-bus specification and user manual, UM10204, 2014**
- **MMA8451Q data sheet**
- Joseph Yiu, The Definitive Guide to the ARM Cortex-M0, Elsevier, 2011
- KL05 Sub-Family Reference Manual, Freescale Semiconductor
- Kinetis L Peripheral Module Quick Reference, Freescale Semiconductor

2. Magistrala I2C

Interfejs I2C, w przeciwieństwie do UART, działa w sposób synchroniczny. Oznacza to, że sygnał zegarowy jest przesyłany między układami, a więc nie musi polegać na zegarach wbudowanych w komunikujące się układy.

Interfejs I2C, w przeciwieństwie do SPI, ma ograniczoną liczbę niezbędnych linii do dwóch, ale jednocześnie umożliwia podłączanie wielu układów do wspólnej magistrali. Urządzenia podłączone możemy podzielić na nadrzędne (*master*) i podrzędne (*slave*). Wersja podstawowa zakłada jedno urządzenie nadrzędne na magistralę.

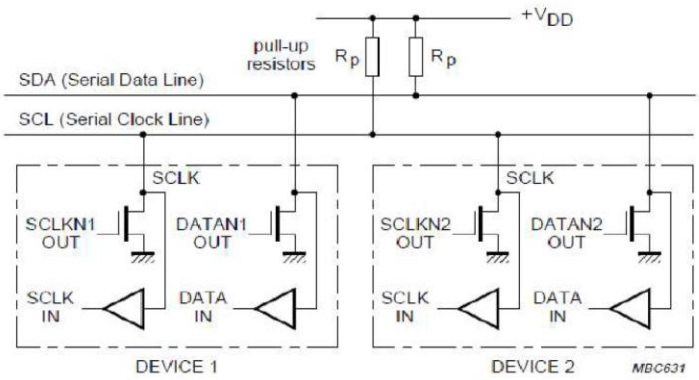


Rysunek 1. Sposób podłączenia urządzeń do magistrali I2C.

Magistrala I2C składa się z dwóch linii:

- SCL - zegar - linia kontrolowana przez układ master
  (*slave* może "przeciągać" zegar - *clock stretching*)
- SDA - dane - linia dwukierunkowa (*half-duplex*)

Sprzętowo wykorzystano w tym interfejsie wyjścia typu otwarty kolektor (*open drain*). Oznacza to, że gdy mikrokontroler wystawia na pinie logiczne 0, linia jest zwierana do masy. Natomiast logiczna 1 zamiast zwierać wyście do zasilania, pozostawia linię niepodłączoną - dokładnie tak samo, jak podczas pracy linii jako wejścia. Wymaga to zastosowania zewnętrznych rezystorów podciągających do zasilania na obu liniach. Istnieją metody podłączenia na jednej magistrali układów o różnych poziomach logicznej 1.



Rysunek 2. Sposób wykorzystania wyjść typu otwarty kolektor w magistrali I2C.

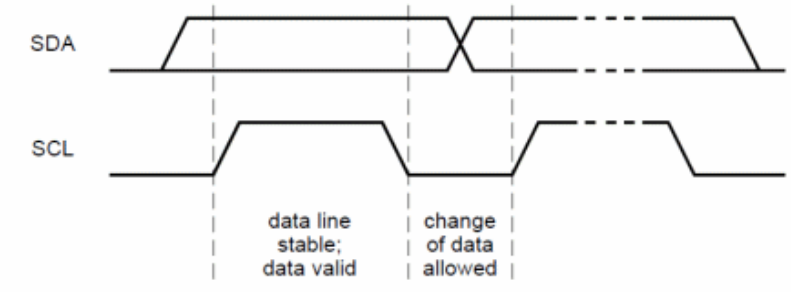
Częstotliwość z jaką pracuje magistrala jest ograniczona przez ilość podłączonych do niej układów i sposobu prowadzenia ścieżek (pojemności pasożytnicze), a wynieść może maksymalnie 400kHz (standardowo ustala się ją na wartość 100kHz).

Ponieważ magistrala pozwala na podłączenie wielu urządzeń, a nie posiada linii kontrolnej dedykowanej dla każdego z nich (jak ma to miejsce w SPI), standard przewiduje adresowanie urządzeń (w wersji podstawowej na 7 bitach - 127 urządzeń). Adresy są przypisywane urządzeniom przez ich producentów i tylko w nielicznych przypadkach istnieje możliwość ich modyfikacji (dodatkowe wejścia do układu pozwalające na modyfikację paru bitów adresu).

3. Protokół I2C

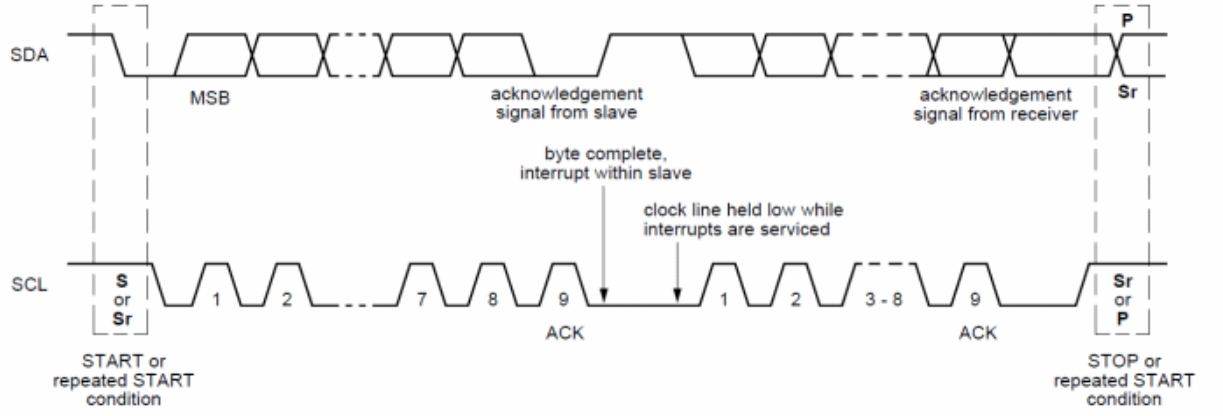
Protokół składa się z szeregu zasad, które zostaną opisane poniżej. Więcej szczegółów można znaleźć w dokumentacji standardu.

Na każdy takt zegarowy przesyłany jest jeden bit. Sygnał SDA może się zmienić tylko wtedy, gdy sygnał SCL jest niski. Podczas gdy zegar jest wysoki, dane powinny być stabilne.



Rysunek 3. Przebieg zależności między sygnałami SDA i SCL.

Każda transmisja I2C zainicjowana przez urządzenie nadrzędne rozpoczyna się od stanu START i kończy stanem STOP. W obu przypadkach stan SCL musi być wysoki. Przejście SDA od wysokiego do niskiego jest uważane za START, a przejście od niskiego do wysokiego jako STOP. Po spełnieniu warunku START, urządzenie nadrzędne może wygenerować powtórny START (RESTART). Jest to odpowiednik normalnego Startu i zwykle następuje po nim adres I2C urządzenia podrzędnego.

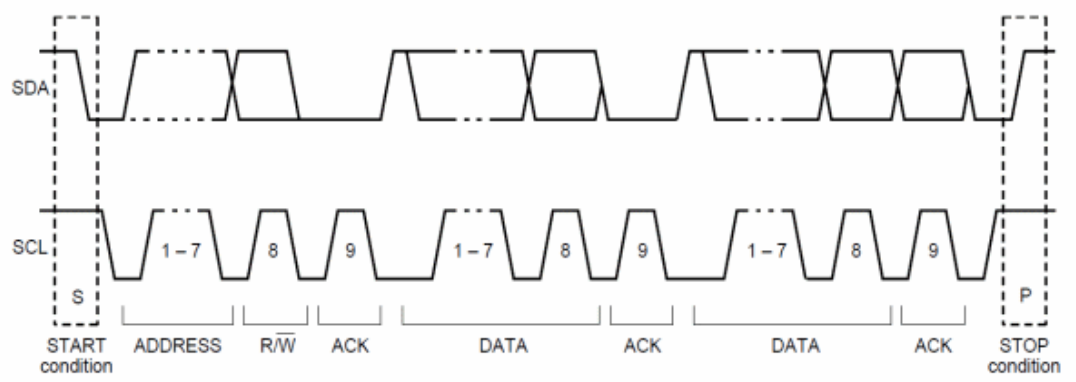


Rysunek 4. Przebieg pokazujący stany START i STOP, oraz miejsce potwierdzania danych.

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.

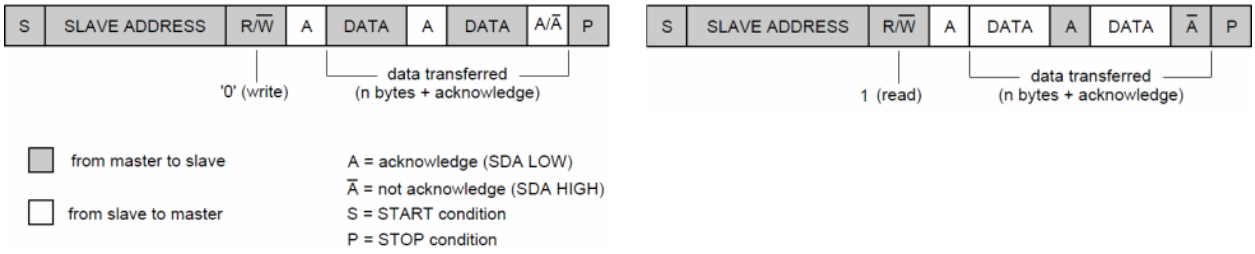
Każde urządzenie typu *slave* na magistrali powinno mieć unikalny adres (7-bitowy). Komunikacja rozpoczyna się od warunku START, po którym następuje adres *slave* i bit kierunku

danych. Jeśli ten bit ma wartość 0, wówczas *master* chce zapisać dane do urządzenia *slave*. W przeciwnym razie, jeśli bit kierunku danych ma wartość 1, *master* chce dokonać odczytu. Komunikacja jest zakończona warunkiem STOP, co oznacza również, że magistrala I2C jest wolna. Jeśli *master* potrzebuje komunikacji z innymi urządzeniami podrzędnymi, może wygenerować RESTART z innym adresem *slave* bez generowania warunku STOP. Wszystkie bajty są przesyłane począwszy od bitu MSB.



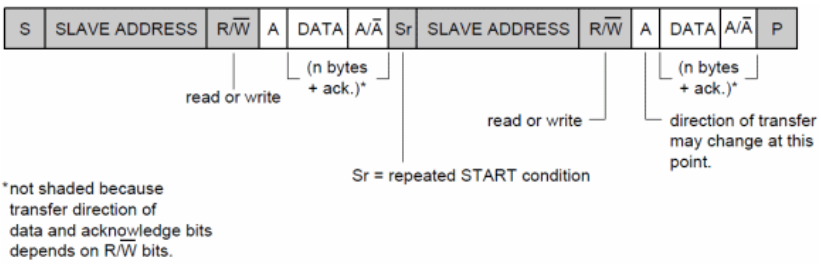
Rysunek 5. Przebieg pokazujący stany START i STOP, oraz miejsce potwierdzania danych.

Jeśli urządzenie *master* tylko zapisuje lub tylko odczytuje z urządzenia *slave*, kierunek przesyłania danych nie ulega zmianie. Różnica polega jedynie na prawidłowym ustawieniu bitu R/W. Potwierdzenie adresu w obu przypadkach wysyła *slave*. Zmienia się sposób potwierdzania odbioru danych – potwierdza zawsze odbierający. Należy pamiętać, że potwierdzenie ostatniego pakietu jest negowane.



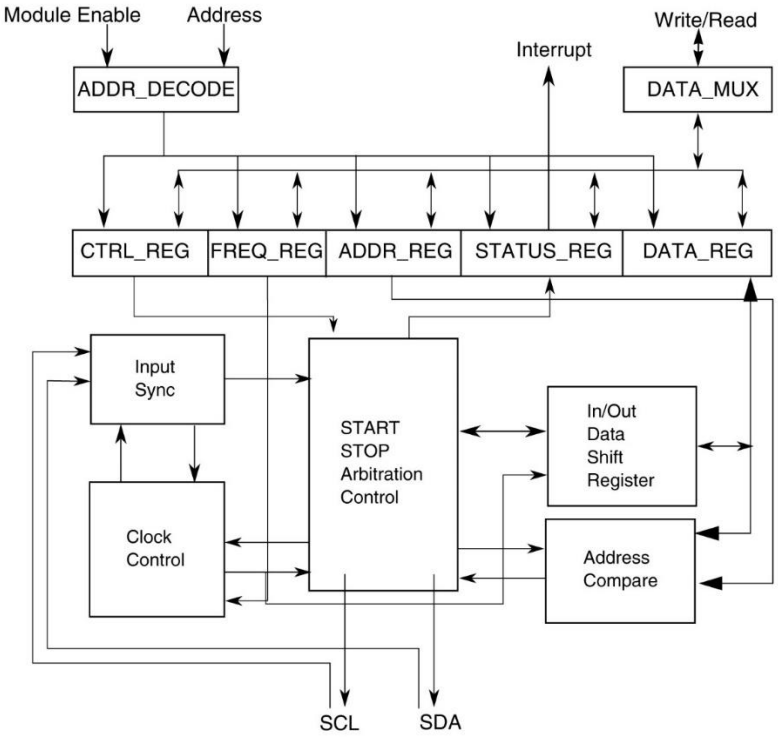
Rysunek 6. Ramki transmisji I2C: zapis i odczyt.

Istnieją sytuacje, w których *master* musi zapisać dane, a następnie odczytać z urządzenia *slave*. Oznacza to wysłanie adresu I2C z bitem R/W ustawionym na zapis, a następnie wysłanie dodatkowych danych, takich jak adres rejestru. Po zakończeniu zapisu urządzenie nadrzędne generuje warunek RESTART i wysyła adres I2C z bitem R/W ustawionym na odczyt. Po tym następuje zmiana kierunku przesyłania danych i urządzenie nadrzędne zaczyna odczytywać dane.



Rysunek 7. Ramka transmisji I2C: zmiana kierunku transmisji.

4. Moduł I2C w procesorze KL05



Rysunek 8. Schemat blokowy modułu I2C

Do głównych funkcji modułu możemy zaliczyć:

- generowanie i wykrywanie sygnałów START i STOP,
- generowanie i wykrywanie sygnału RESTART,
- generowanie i wykrywanie bitu ACKNOWLEDGE,
- możliwość pracy na magistrali z wieloma masterami,
- obsługa przerwań,
- wykrywanie zajętej magistrali,
- praca w trybie adresów 10 bitowych,
- możliwość wykorzystania DMA.

Moduł może pracować w jednym z trzech trybów: Run / Wait / Stop. Tryb Run jest podstawowym i będzie wykorzystany w tej instrukcji.

Podczas pracy z modułem będą wykorzystywane rejestry:

I2C Control Register 1 (I2C0_C1)

Bit	7	6	5	4	3	2	1	0
Read	IICEN	IICIE	MST	TX	TXAK	0	WUEN	DMAEN
Write						RSTA		
Reset	0	0	0	0	0	0	0	0

Główny rejestr kontrolny składa się przede wszystkim z bitów włączających zarówno sam moduł jak i jego przerwania (IICEN, IICIE). Kolejne bity będą wykorzystywane w trakcie

transmisji:

- MST – generowanie warunków START i STOP,
- TX – nadawanie albo odbieranie,
- TXAK – włączenie nadawania bitu potwierdzenia,
- RSTA – generowanie warunku RESTART.

I2C Status Register (I2C0_S)

Bit	7	6	5	4	3	2	1	0
Read	TCF	IAAS	BUSY	ARBL	RAM	SRW	IICIF	RXAK
Write				w1c			w1c	
Reset	1	0	0	0	0	0	0	0

Rejestr głównie do odczytu stanu modułu. Bit TCF informuje o wykonanej transmisji pakietu wraz z bitem potwierdzającym. Bit IICIF informuje o pojawieniu się przerwania (zapis 1 kasuje tą flagę). Tymczasem bit RXAK będzie ustawiony na '1' jeżeli nie został wykryty bit potwierdzenia. Tym samym bit RXAK można wykorzystać do sprawdzenia czy urządzenie *slave* jest obecne na magistrali.

I2C Data I/O register (I2C0_D)

Bit	7	6	5	4	3	2	1	0
Read Write	DATA							
Reset	0	0	0	0	0	0	0	0

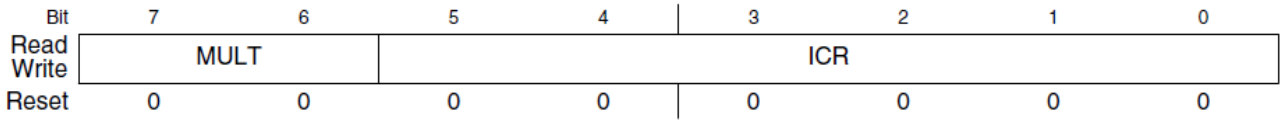
Rejestr służący jednocześnie do zapisu i odczytu 8 bitowych danych. Czytanie z tego rejestru wywołuje procedurę odbierania danych, dlatego właściwy pakiet pojawia się w tym rejestrze dopiero przy drugim odczycie

Aby przygotować układ do pracy należy wykonać następujące czynności:

- podłączyć sygnał taktujący do modułu I2C0, w rejestrze SIM_SCGC4,
- dołączyć sygnał taktujący do odpowiedniego portu, którego końcówki realizują funkcje SCL (PTB3) i SDA (PTB4), w rejestrze SIM_SCGC5,
- ustawić odpowiednią funkcję dla wykorzystywanych końcówek portu, w rejestrze PORTB_PCRx[MUX=2]. PTB3 – zegar SCL, PTB4 – dane SDA,
- upewnić się, że moduł I2C0 jest wyłączony podczas zmiany zawartości jego rejestrów, w rejestrze I2C0_C1 skasować bit IICEN,
- w rejestrze I2C0_F ustawić częstotliwość pracy magistrali I2C (generowanego przez *master* sygnału SCL),
- włączyć przerwania od modułu przez ustawienie bitu IICIE w rejestrze I2C0_C1.

4.1. Wyliczanie częstotliwości pracy magistrali

I2C Frequency Divider register (I2C0_F)



Bit	7	6	5	4	3	2	1	0
Read / Write	MULT		ICR					
Reset	0	0	0	0	0	0	0	0

Rejestrem odpowiedzialnym za ustawienie częstotliwości jest I2C0_F. Składa się on z dwóch pozycji:

- MULT – mnożnik, którego wartość może wynosić 0, 1, lub 2,
- ICR – przeskalowuje zegar magistrali w celu wyboru szybkości transmisji.

Wzór na szybkość transmisji:

$$baud\ rate_{I2C} = \frac{\text{bus frequency}}{2^{MULT} \times SCL\ divider}$$

Wartość *SCL divider* wybierana jest przez bity ICR.

Liczbę należy dobrać z tabeli *36-28 - I2C divider and hold values* (Reference Manual).

Przykładowo, dla MULT=0x01 i ICR=0x19:

$$\frac{20.97\ [MHz]}{2^1 \times 96} = \ 109\ [kHz]$$

5. Puk puk ... sprawdzenie listy obecności na magistrali

ToDo

1.1 Podłącz do płytki FRDM-KL05 jedynie kabel USB. Na razie nie podłączaj wyświetlacza LCD.

1.2 Ze strony laboratorium pobierz archiwum z projektem.

1.3 Rozpakuj plik do nowo utworzonego folderu, a następnie otwórz projekt (*demo.uvprojx*).

1.4 Projekt podzielony jest na foldery 'src' i 'inc'. Projekt składa się z plików bibliotecznych z pierwszego laboratorium oraz pliku:

- *i2c.c* - zawiera inicjalizację magistrali I2C i funkcje pomocnicze.

1.5 Zbuduj projekt i załaduj program na płytkę FRDM-KL05.

1.6 Uruchom ulubiony program terminala (*putty*, *termite*, *teraterm*, ...), zlokalizuj numer portu COM, pod którym zameldował się nasz moduł, ustaw prędkość transmisji na 9600 i otwórz port.

Po prawidłowym wykonaniu powyższych czynności układ po naciśnięciu przycisku RESET powinien zachowywać się w następujący sposób:

- dioda RGB - po sekwencji powitalnej (krótkie zaświecenie każdego z kolorów) dioda czerwona miga z częstotliwością około 1Hz, tzw. heartbeat służy do prostego zwizualizowania, że procesor się nie zawiesił,

- terminal – wyświetla informacje powitalne i kończy na słowie „*Sleep ...*”.

Projekt w wersji podstawowej wykorzystuje jedynie biblioteki LED i UART. Program rozpoczyna się on inicjalizacji obu obiektów i pokazania ich podstawowej funkcjonalności.

Główna pętla programu składa się z dwóch zadań (jedno puste), które uruchamiane są w oparciu o zegar systemowy (SysTick Timer - laboratorium 3). Aktywne zadanie to miganie czerwoną diodą.

1.7 W pliku *main.c* odkomentuj wskazane 3 linie kodu. Upewnij się czy dołączona jest biblioteka przez załączenie pliku nagłówkowego *i2c.h* na początku pliku, a następnie otwórz plik *i2c.c*. Zlokalizuj funkcję *I2C_Init()* i odkomentuj jej zawartość. Uzupełnij konfigurację rejestru F (informacje w punkcie 4.1), tak aby otrzymać szybkość transmisji zbliżoną do 100kbps.

1.8 Zbuduj i uruchom program. Sprawdź jakie urządzenie zgłosiło się na magistrali. Zapisz jego adres – to akcelerometr umieszczony na FRDM-KL05Z.

1.9 Podłącz wyświetlacz LCD zgodnie ze wskazówkami z pierwszej instrukcji z tą różnicą, że zasilanie wyjątkowo podłącz do 3.3V. W ten sposób sam wyświetlacz jest nieaktywny, ale działa układ ekspandera.

1.10 Uruchom program jeszcze raz (przycisk RESET). Sprawdź jakie urządzenie dodatkowo zgłosiło się na magistrali. Zapisz jego adres – to ekspander wejść/wyjść.

1.11 Zlokalizuj uruchomioną funkcję (*bus_scan()*) na końcu pliku *main.c*. Prześledź kolejność działań wykonywanych w funkcji *I2C_Ping()*.

6. Zapis i odczyt z ekspandera I/O – loop test

ToDo

2.1 W pliku *main.c* odkomentuj kolejne 2 linie kodu. Na podstawie odczytu z zadania 1.10 wpisz adres ekspandera jako argument do funkcji *expander_looptest()*.

2.2 Zlokalizuj funkcję *expander_looptest()* na końcu pliku *main.c*. Działa ona na zasadzie kolejnych zapisów i odczytów z magistrali. Prześledź kolejność działań wykonywanych w funkcjach *I2C_Write()* i *I2C_Read()*. W jaki sposób różnią się od siebie wysyłane adresy w obu funkcjach?

2.3 Zbuduj i uruchom program. Sprawdź wyniki testu.

2.4 Wykorzystując zdobytą wiedzę wybierz odpowiednią funkcję i wpisz do niej argumenty, tak aby ustawić 4 bit w ekspanderze – jest on podłączony do anody diody podświetlającej wyświetlacz LCD.

2.5 Zbuduj i uruchom program.

7. Odczyt zawartości rejestru w sensorze – kim jesteś?

ToDo

3.1 W pliku *main.c* odkomentuj kolejne 4 linie kodu. Na podstawie odczytu z zadania 1.8 wpisz adres akcelerometru jako pierwszy argument do funkcji *I2C_ReadReg()*. Kolejnymi argumentami tej funkcji są adres rejestru sensora, który chcemy odczytać (*who_am_i*) i adres w pamięci mikrokontrolera, gdzie chcemy zapisać odczyt. Tabela z adresami rejestrów sensora znajduje się w dokumentacji producenta (*MMA8451Q data sheet*).

3.2 Zbuduj i uruchom program. Ile wynosi wartość *Device ID*?

3.3 Prześledź kolejność działań wykonywanych w funkcji *I2C_ReadReg()*.

8. Zapis do rejestru w sensorze – pierwszy pomiar grawitacji

ToDo

4.1 W pliku *main.c* odkomentuj kolejne 5 linii kodu. Uzupełnij brakujące wartości zgodnie z wcześniejszymi odczytami i dokumentacją akcelerometru. Ustawiamy bit *ACTIVE* w rejestrze *CTRL_REG1* aby uruchomić pomiary, a następnie odczytujemy starsze 8 bitów przyśpieszenia w osi Z.

4.2 W zadaniu będą używane funkcje operujące na wartościach zmiennoprzecinkowych, dlatego zwiększ pojemność stosu, ustawiając go na wartość: *Stack_Size EQU 0x00000300*, w pliku *startup_MKL05Z4.s*.

4.3 Zbuduj i uruchom program.

4.4 Zwróć uwagę na sposób przeliczenia odczytanej wartości na jednostkę [g]. Operacje przesuwania wartości odczytu wynikają z konwersji na liczbę 16 bitową ze znakiem. Dzielenie odczytu przez wartość 4096 wynika z domyślnie ustawionej czułości sensora.

9. Blokowy odczyt wielu rejestrów z sensora – XYZ

ToDo

5.1 W pliku *main.c* odkomentuj kolejne 8 linii kodu. Uzupełnij brakujące wartości zgodnie z wcześniejszymi odczytami i dokumentacją akcelerometru. Tym razem chcemy odczytać wszystkie dostępne pomiary przyśpieszenia z pełną precyzją. Wykorzystujemy inkrementację adresów w trakcie odczytu.

5.2 Zbuduj i uruchom program.

5.3 Zwróć uwagę na sposób przeliczenia odczytanej wartości na jednostkę [g]. Operacje przesuwania wartości odczytu wynikają z konwersji na liczbę 16 bitową ze znakiem. Dzielenie odczytu przez wartość 4096 wynika z domyślnie ustawionej czułości sensora.

10. Zadanie dodatkowe – sejsmograf

ToDo

6.1 Zmień czułość sensora z 2g na 8g. Następnie wykorzystaj puste zadanie w pętli *while()* do ciągłego odczytu wszystkich pomiarów ułożonych w 3 kolumnach. Czy da się te odczyty zwizualizować? Wykorzystaj do tego ulubiony język programowania (Python, Matlab, …).