Akademia Górniczo-Hutnicza
w Krakowie
Instytut Elektroniki
WIET

Laboratorium
Technika Mikroprocesorowa 2

Ćwiczenie 8

Port szeregowy I²C
Akcelerometr

Autor: Mariusz Sokołowski

wer. 28.09.2021

1. WSTĘP

1.1. CEL

Celem ćwiczenia jest:

- zapoznanie studenta z podstawami techniki poprawnej inicjalizacji i obsługi synchronicznego portu szeregowego I^2C,
- poznanie możliwości wykorzystania portu I^2C do współpracy z czujnikami cyfrowymi, na przykładzie akcelerometru:
  - pomiar przyspieszenia w trzech kierunkach: X, Y i Z,
  - sterowanie urządzeniami zewnętrznymi za pomocą gestów.

1.2. WYMAGANIA

Sprzętowe:

- komputer klasy PC, spełniający wymagania sprzętowe aplikacji KEIL v5,
- zestaw FRDMKL05Z

Programowe:

- system operacyjny Windows 7 lub wyższy (wszystkie instrukcje powstały w oparciu o Windows 7 Pro x64),
- środowisko Keil / uVision 5 MDK-ARM

Doświadczenie:

- podstawowa umiejętność obsługi komputera klasy PC,
- podstawowa znajomość systemów operacyjnych rodziny Windows,
- umiejętność programowania w języku C.

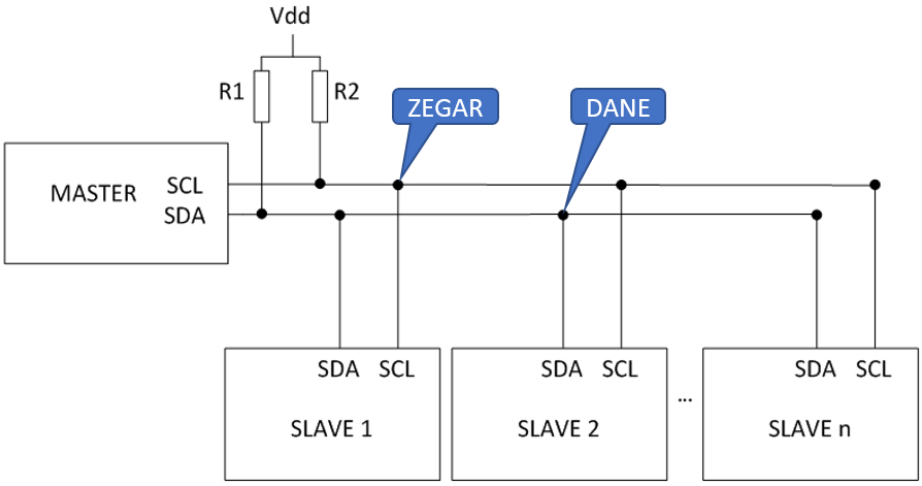
Literatura:

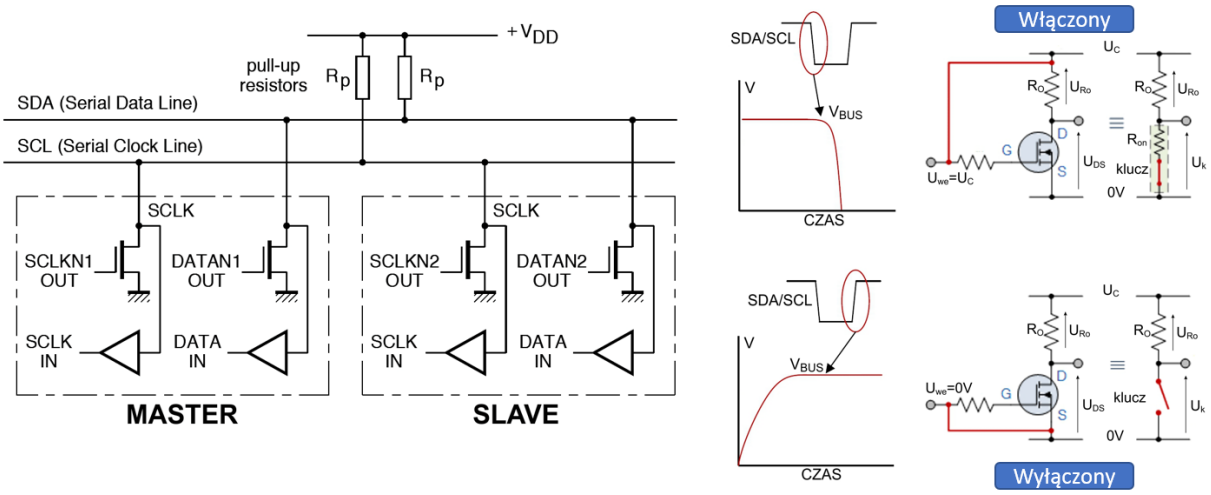
- KL05 Sub-Family Reference Manual, Freescale Semiconductor
- Kinetis L Peripheral Module Quick Reference, Freescale Semiconductor
- Joseph Yiu, The Definitive Guide to the ARM Cortex-M0, Elsevier, 2011
- ARM „Cortex-M0+ Devices Generic User Guide”
- MMA8451Q, 3-axis, 14-bit/8-bit digital accelerometer, NXP Semiconductors Technical data
- Kinetis KL05 32 KB Flash - 48 MHz Cortex-M0+ Based Microcontroller, Data Sheet
- PuTTY User Manual (terminal)

2. STANDARD I²C

Interfejs I²C ma ograniczoną liczbę niezbędnych linii do dwóch i jednocześnie umożliwia podłączenie wielu układów do wspólnej magistrali, które można podzielić na nadrzędne (*master*) i podrzędne (*slave*). Wersja podstawowa zakłada jedno urządzenie nadrzędne na magistralę.

Interfejs I²C, w przeciwieństwie do UART, działa w sposób synchroniczny. Sygnał zegarowy SCL jest generowany przez urządzenie typu *master* i przesyłany osobną linią do wszystkich urządzeń typu *slave*. Linia danych SDA jest dwukierunkowa, co oznacza, że transfer jest typu *Half Duplex* (nadawanie i odbiór nie mogą zachodzić w tym samym czasie). Obydwie linie muszą być podłączone do napięcia zasilania za pomocą rezystorów (R_1 i R_2).





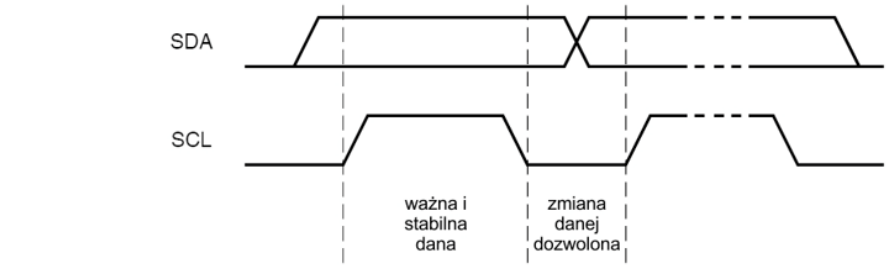
Jeśli któreś urządzenie ustawi wartość „0”, to cała linia będzie w stanie „0”. Nazywa się to „*wire AND*”, ponieważ zachowuje się jak funkcja logiczna AND.

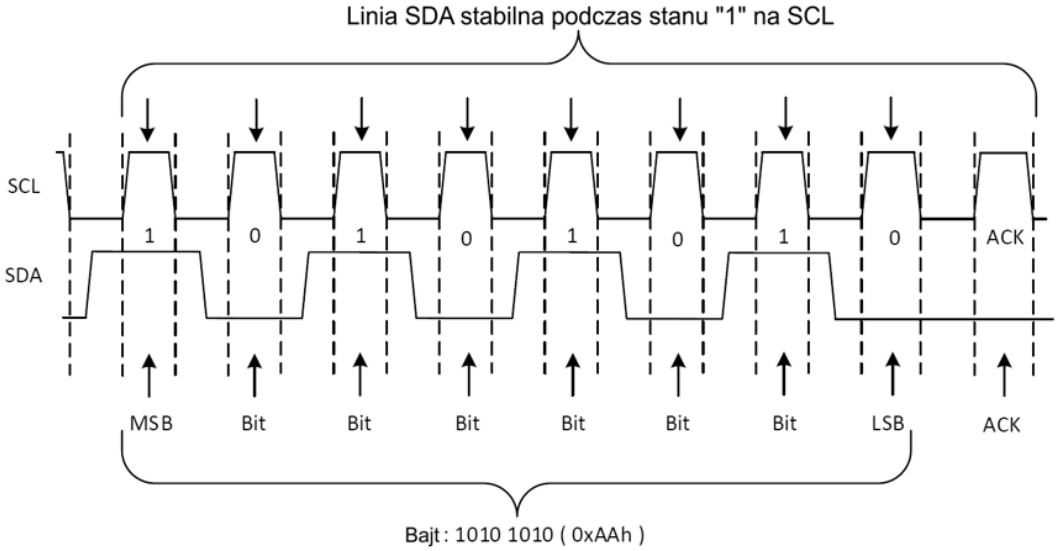
Częstotliwość, z jaką pracuje magistrala jest ograniczona przez ilość podłączonych do niej układów i sposobu prowadzenia ścieżek (pojemności pasożytnicze), a wynieść może maksymalnie do 3.4 Mb/s (standardowo ustala się ją na wartość 100kb/s), do 4-ech metrów (pojemność do 400pF).

Układy typu *master* przejmują kontrolę nad magistralą, zarządzając transmisją danych, czyli inicjalizują połączenie, kończą je oraz ustalają taktowanie. Układy typu *slave* są wybierane przez układ *master*.

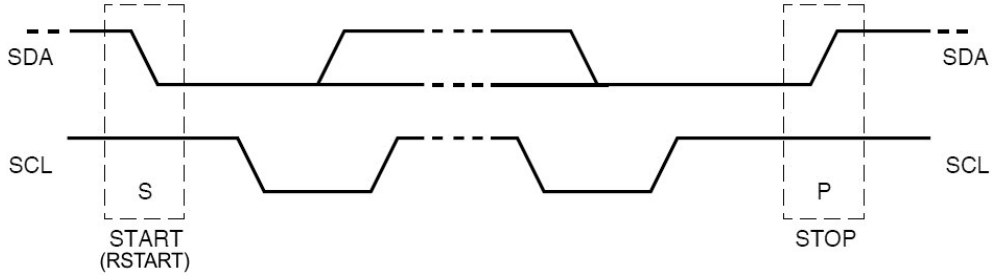
Ponieważ magistrala pozwala na podłączenie wielu urządzeń, a nie posiada linii kontrolnej dedykowanej dla każdego z nich (jak ma to miejsce w przypadku magistrali SPI), więc jednym z etapów protokołu wymiany informacji jest adresowanie urządzeń (w wersji standardowej adres jest 7-bitowy). Adresy są przypisywane urządzeniom przez ich producentów i tylko w nielicznych przypadkach istnieje możliwość ich modyfikacji (dodatkowe wejścia do układu pozwalające na modyfikację paru bitów adresu). Teoretycznie może być do 127 urządzeń, ale pewna pula adresów jest zarezerwowana, więc praktycznie może być do 119 urządzeń.

Jak w przypadku każdego portu szeregowego, I^2C również posiada swój protokół wymiany informacji. Aby zapisać lub odczytać daną, musi być wykonanych szereg czynności, w ściśle określonej kolejności. Inaczej wygląda transfer danych do/z urządzenia o prostej budowie, czyli zawierającego tylko jeden rejestr danych wejściowych i jeden rejestr danych wyjściowych (np. wspominany, przy okazji omawiania wyświetlacza LCD, ekspander I^2C). Natomiast jeszcze inaczej będzie to wyglądało w przypadku urządzenia złożonego (np. akcelerometr, który zawiera wiele rejestrów wewnętrznych, posiadających swoje, wewnętrzne adresy, działające tylko w ramach urządzenia). Wszystkie „akcje" przeprowadzane na magistrali odbywają się w takt zegara SCL. Każdy bit danych musi się pojawić w ściśle określonym czasie trwania taktu zegarowego. Na szczęście nie musimy się tym martwić, bo robi to za nas sterownik magistrali.

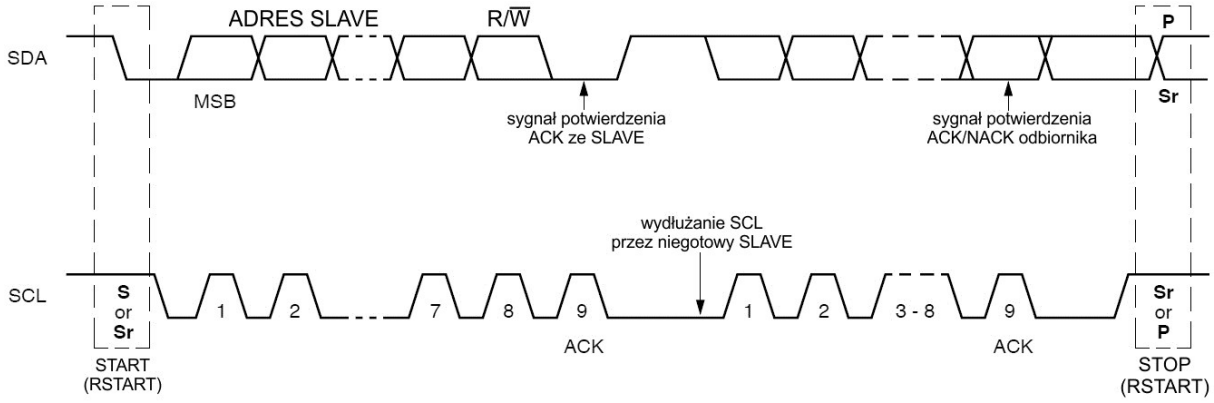




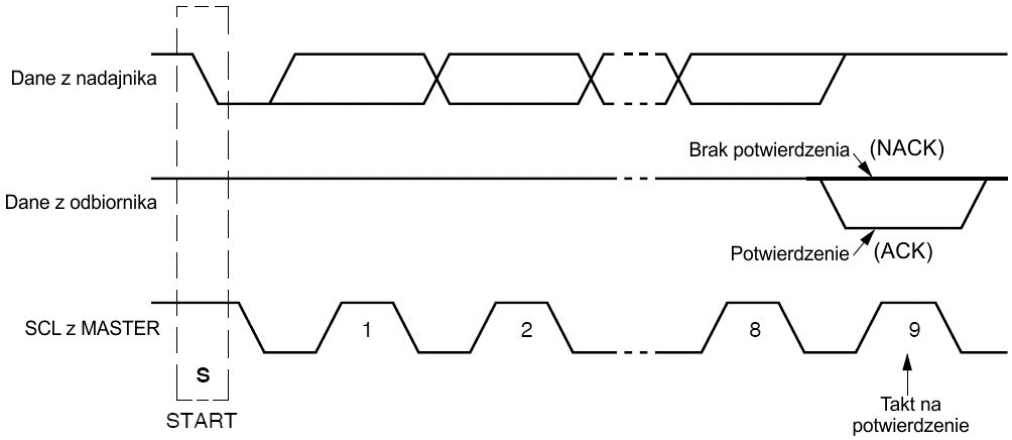
Każdy transfer zaczyna się od kombinacji START (S), a kończy kombinacją STOP (P), które generuje MASTER na SDA i SCL.



Jeśli MASTER chce odczytać coś ze SLAVE, to w trakcie transferu generuje kombinację powtórnego START (Sr) (ang. Repeated START – RSTART), aby móc zmienić np. kierunek transferu. Na początku każdego transferu zawsze znajduje się sekwencja zapisu adresu SLAVE. Adres ten jest czytany przez wszystkie urządzenia i „odzywa" się tylko to, którego jest to adres. Jeśli SLAVE jest niegotowy, to wymusza i wydłuża stan „0" na SCL.



Każda sekwencja zapisu lub odczytu kończy się potwierdzeniem, przesyłanym albo przez MASTER albo SLAVE, w zależności od kierunku przepływu danych. Sygnał potwierdzający może być na „tak" (potwierdzenie – ACK „0") lub na „nie" (brak potwierdzenia – NACK „1"). Potwierdzenie ACK oznacza gotowość na następną daną.



Zawsze potwierdza odbiornik. Który, MASTER czy SLAVE, to zależy od kierunku transferu.

Jeśli adres nie zostanie potwierdzony przez właściwe urządzenie SLAVE, to oznacza, że:

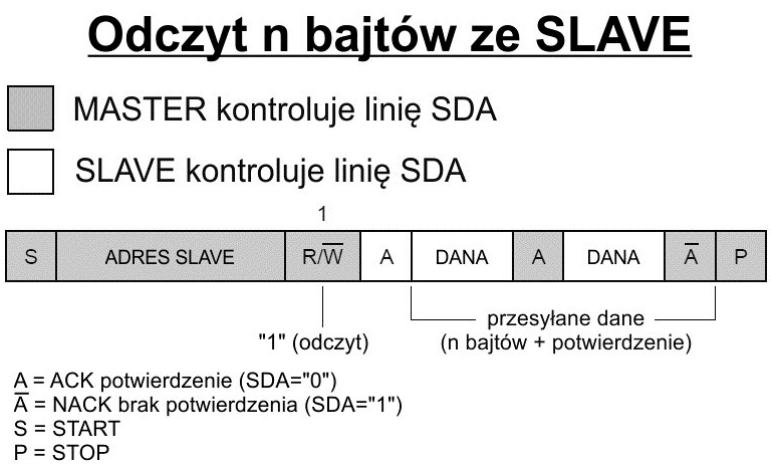
- adres nie został poprawnie odebrany przez SLAVE,
- brak urządzenia o danym adresie na magistrali. Efekt ten jest wykorzystywany do „odpytywania" magistrali na obecność poszczególnych urządzeń (adresów).

SLAVE powinien również potwierdzić odbiór danych. Jeśli zamiast ACK, pojawi się jego brak (NACK), to znaczy, że:

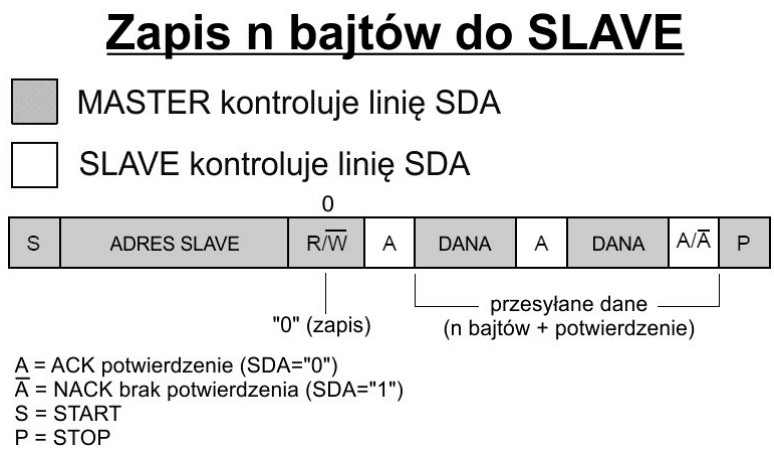
- urządzenie SLAVE nie jest gotowe na dane lub nie może ich więcej przyjąć,
- dana nie została odebrana poprawnie.

Odbiornik MASTER generuje potwierdzenie ACK po każdej poprawnie odebranej danej. Brak potwierdzenia (NACK) oznacza, że skończył pobieranie danych i jest znakiem dla SLAVE, że ma przestać nadawać.

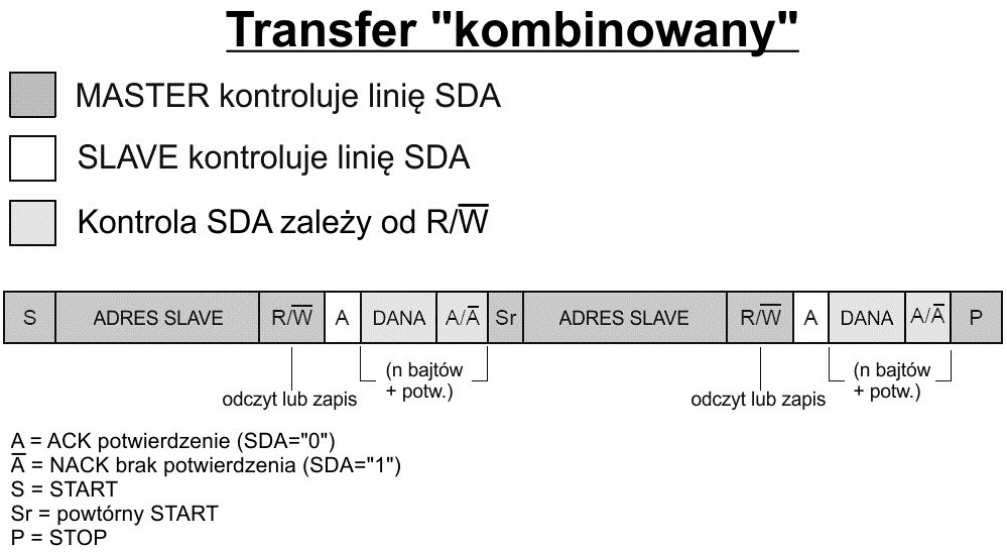
2.1. Odczyt ze SLAVE (bez struktury rejestrowej)



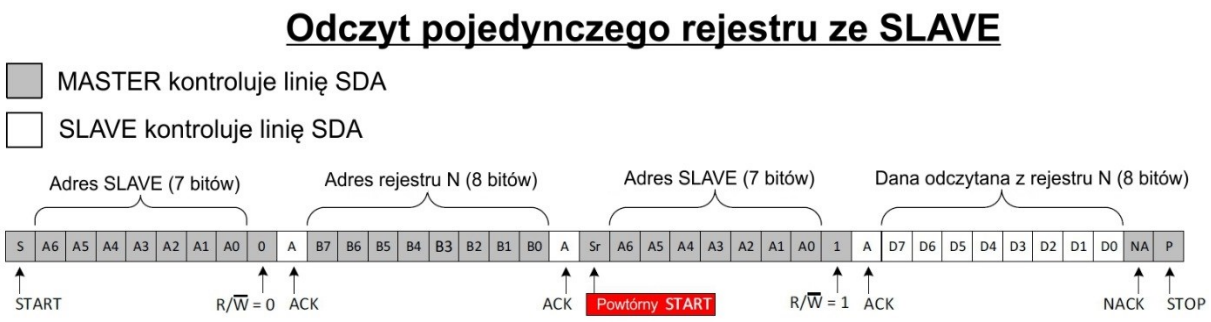
2.2. Zapis do SLAVE (bez struktury rejestrowej)



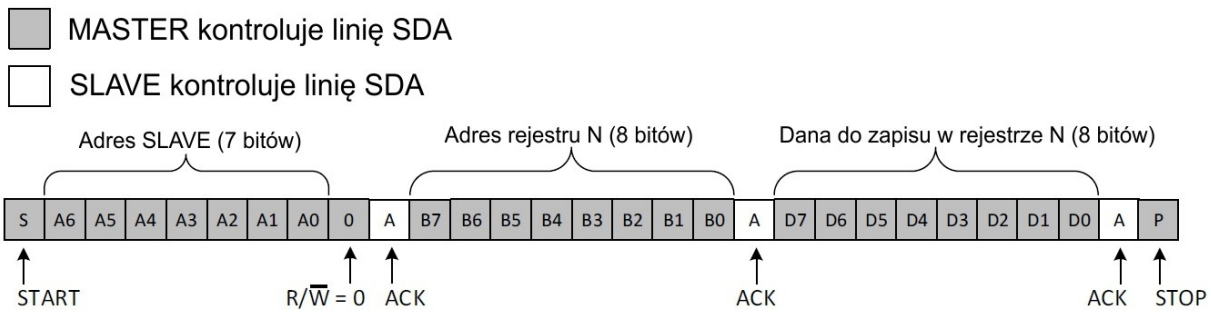
2.3. TRANSFER KOMBINOWANY (BEZ STRUKTURY REJESTROWEJ)



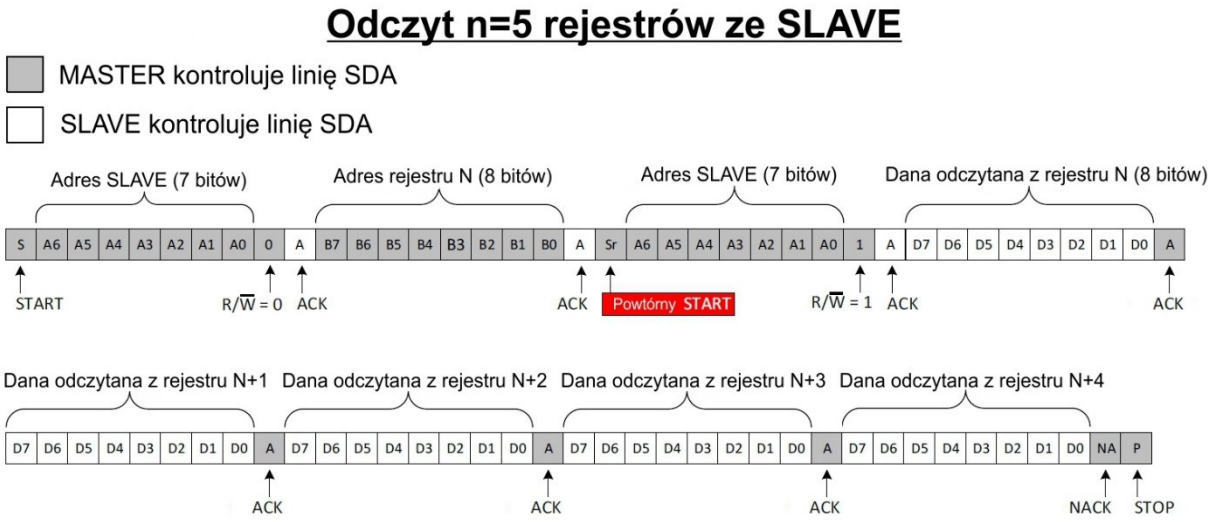
2.4. ODCZYT POJEDYNCZEGO REJESTRU ZE SLAVE (STRUKTURA REJESTROWA)



2.5. ZAPIS POJEDYNCZEGO REJESTRU W SLAVE (STRUKTURA REJESTROWA)

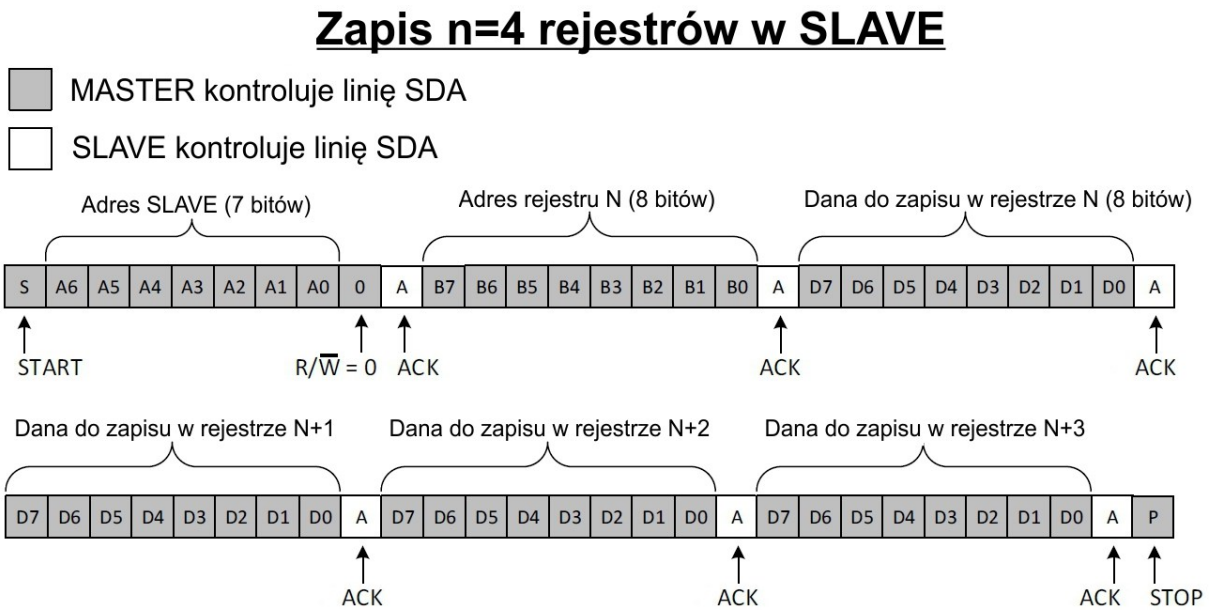


2.6. Odczyt n rejestrów ze SLAVE (struktura rejestrowa)



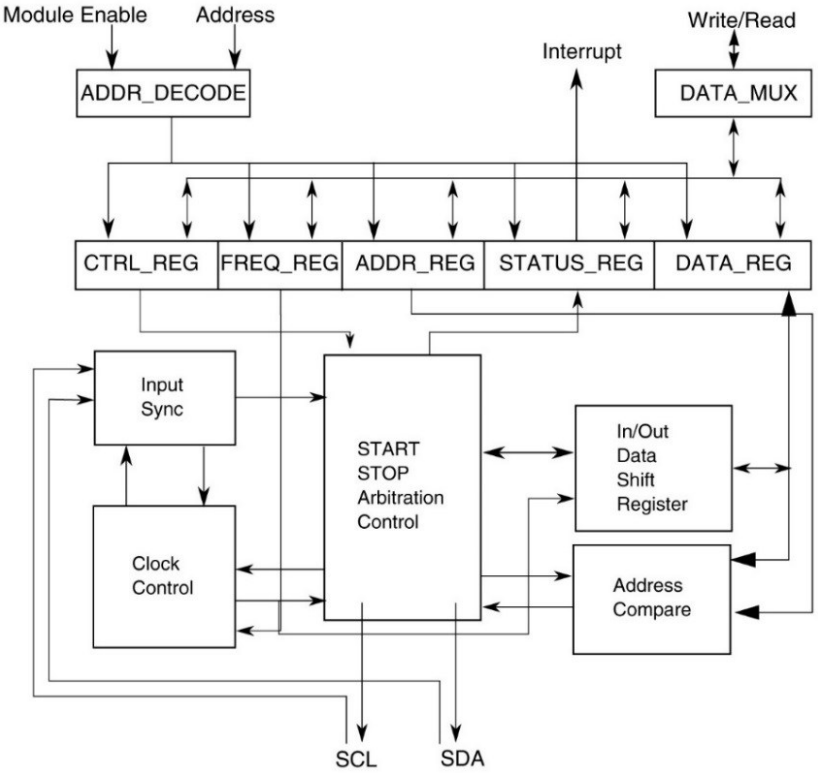
W urządzeniu SLAVE adresy kolejnych rejestrów będą się automatycznie aktualizować (zwiększać o 1) po każdym odczycie, do czasu wysłania sygnału NACK przez MASTER.

2.7. Zapis n rejestrów w SLAVE (struktura rejestrowa)



3. MODUŁ PORTU SZEREGOWEGO I²C

Układ MKL05Z32VLC4, będący centralnym mikrokontrolerem zestawu FRDM-KL05Z, posiada w swoich zasobach sprzętowych, jeden synchroniczny port szeregowy I²C – I2C0 (Rys. 1).



Rys. 1 Schemat blokowy portu I2C0

Niniejszy port spełnia warunki specyfikacji, zdefiniowane dla magistrali I²C.

Aby przygotować układ I2C0 do pracy, należy wykonać następujące czynności:

- Sprawdzić, jaką wartość ma stała CLOCK_SETUP, w zbiorze *system_MKL05Z4.c*. Informacja ta będzie miała wpływ na ustawienia rejestrów, odpowiedzialnych za szybkość transmisji. W zależności od stałej CLOCK_SETUP, parametry podstawowych sygnałów zegarowych mają następujące wartości:
  - CLOCK_SETUP=0 (wartość domyślna):
    - zegar referencyjny dla modułu MCG - 32768Hz,
    - Core clock - 41943040Hz,
    - BusClock - 20971520Hz.
  - CLOCK_SETUP=1:
    - zegar referencyjny dla modułu MCG - 32768Hz,
    - Core clock - 47972352Hz,
    - BusClock - 23986176Hz.
  - CLOCK_SETUP=2:
    - zegar referencyjny dla modułu MCG – 4MHz,
    - Core clock – 4MHz,
    - BusClock – 2MHz.

❖ dołączyć sygnał taktujący do modułu I2C0, w rejestrze SIM_SCGC4 [I2C0=1],

System Clock Gating Control Register 4 (SIM_SCGC4)

Address: 4004_7000h base + 1034h offset = 4004_8034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	1				0				0	SPI0	0		CMP	0	0	
W																
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		0	0	0	UART0	0		0	I2C0	1		0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

❖ dołączyć sygnał taktujący do odpowiedniego portu, którego końcówki realizują funkcje SCL (PTB3) i SDA (PTB4), w rejestrze SIM_SCGC5[PORTB=1],

❖ ustawić odpowiednią funkcję dla wykorzystywanych końcówek portu, w rejestrze PORTB_PCRx[MUX=2]. PTB3 – zegar SCL, PTB4 – dane SDA,

w rejestrze I2C0_F, ustawić wartości pól MULT i ICR tak, aby dla danej wartości częstotliwości BusClock zrealizować założoną częstotliwość transferu BR. Wartość ICR wybieramy z tabeli Tab. 1, po uprzednim wyliczeniu wartości „SCL divider”, wg poniższego wzoru:

$$SCL\ divider = \frac{BusClock}{mul \cdot BR}\ [Hz]$$

Wartość pola MULT jest związana z czynnikiem *mul*, poniższą zależnością:
mul=1 MULT=00
mul=2 MULT=01
mul=4 MULT=10

I2C Frequency Divider register (I2Cx_F)

Address: 4006_6000h base + 1h offset = 4006_6001h

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

❖ w przypadku wykorzystywania przerwań, odblokować przerwania od portu I2C0, w rejestrze I2C0_C1[IICIE=1].

W tym momencie układ I2C0 jest gotowy do pracy.

ICR (hex)	SCL divider	SDA hold value	SCL hold (start) value	SCL hold (stop) value		ICR (hex)	SCL divider (clocks)	SDA hold (clocks)	SCL hold (start) value	SCL hold (stop) value
00	20	7	6	11		20	160	17	78	81
01	22	7	7	12		21	192	17	94	97
02	24	8	8	13		22	224	33	110	113
03	26	8	9	14		23	256	33	126	129
04	28	9	10	15		24	288	49	142	145
05	30	9	11	16		25	320	49	158	161
06	34	10	13	18		26	384	65	190	193
07	40	10	16	21		27	480	65	238	241
08	28	7	10	15		28	320	33	158	161
09	32	7	12	17		29	384	33	190	193
0A	36	9	14	19		2A	448	65	222	225
0B	40	9	16	21		2B	512	65	254	257
0C	44	11	18	23		2C	576	97	286	289
0D	48	11	20	25		2D	640	97	318	321
0E	56	13	24	29		2E	768	129	382	385
0F	68	13	30	35		2F	960	129	478	481
10	48	9	18	25		30	640	65	318	321
11	56	9	22	29		31	768	65	382	385
12	64	13	26	33		32	896	129	446	449
13	72	13	30	37		33	1024	129	510	513
14	80	17	34	41		34	1152	193	574	577
15	88	17	38	45		35	1280	193	638	641
16	104	21	46	53		36	1536	257	766	769
17	128	21	58	65		37	1920	257	958	961
18	80	9	38	41		38	1280	129	638	641
19	96	9	46	49		39	1536	129	766	769
1A	112	17	54	57		3A	1792	257	894	897
1B	128	17	62	65		3B	2048	257	1022	1025
1C	144	25	70	73		3C	2304	385	1150	1153
1D	160	25	78	81		3D	2560	385	1278	1281
1E	192	33	94	97		3E	3072	513	1534	1537
1F	240	33	118	121		3F	3840	513	1918	1921

Tab. 1

4. TRANSFER MIĘDZY „MASTER” A „SLAVE”

Ponieważ cała instrukcja będzie się odnosiła do akcelerometru, znajdującego się na wyposażeniu modułu KL05Z, dlatego omówione zostaną transfery odnoszące się tylko do struktury rejestrowej.

4.1. WYBRANE FUNKCJE BIBLIOTEKI *I2C.C*

Biblioteka ta zawiera podstawowe funkcje, pozwalające na inicjalizację pracy portu oraz mini funkcje, podobne do „klocków”, z których można „poskładać” każdy rodzaj transferu między MASTER a SLAVE.

4.1.1. Mini funkcje

i2c_enable() – aktywowanie portu I2C0
i2c_disable() – deaktywowanie portu I2C0
i2c_m_start() – sygnał START
i2c_m_rstart() – sygnał ponowny START
i2c_m_stop() – sygnał STOP
i2c_tran() – ustaw transfer z MASTER do SLAVE
i2c_rec() – ustaw transfer ze SLAVE do MASTER
i2c_send(uint8_t) – wysłanie danej z MASTER do SLAVE
i2c_recv() – odczyt danej ze SLAVE do MASTER
i2c_wait() – oczekiwanie na koniec transferu i potwierdzenie ACK
i2c_nack() – wyślij NACK po odczycie (od MASTER)
i2c_ack() – wyślij ACK po odczycie (od MASTER)
i2c_clr_IICIF() – skasowanie flagi przerwania, pojawiającej się po transferze

4.1.2. Wybrane funkcje

I. **I2C_Init()** – inicjalizacja pracy portu I2C0.
II. **I2C_Ping(uint8_t address)** – sprawdzanie obecności na magistrali urządzenia o adresie address.
Instrukcja zwraca kod błędu, w przypadku niepowodzenia:
1 – przeterminowanie (za długi czas obsługi),
2 – brak potwierdzenia ACK (brak urządzenia o adresie address),
3 – obydwa błędy równocześnie (brak urządzenia o adresie address).
III. **I2C_WriteReg(uint8_t address, uint8_t reg, uint8_t data)** – zapis jednego rejestru w SLAVE:
address – adres urządzenia na magistrali I^2C (akcelerometr 0x1D),
reg – adres wewnętrznego rejestru urządzenia,
data – dana do zapisania w rejestrze „reg”.
Instrukcja zwraca kod błędu, w przypadku niepowodzenia:
1 – przeterminowanie (za długi czas obsługi),
2 – brak potwierdzenia ACK,
3 – obydwa błędy równocześnie.
IV. **I2C_ReadReg(uint8_t address, uint8_t reg, uint8_t* data)** – odczyt jednego rejestru ze SLAVE:
address – adres urządzenia na magistrali I^2C (akcelerometr 0x1D),
reg – adres wewnętrznego rejestru urządzenia,
*data – wskaźnik do zmiennej (adres zmiennej), w której będzie zapisana dana z rejestru „reg”.
Instrukcja zwraca kod błędu, w przypadku niepowodzenia:
1 – przeterminowanie (za długi czas obsługi),

2 – brak potwierdzenia ACK,

3 – obydwa błędy równocześnie.

V. **I2C_ReadRegBlock(uint8_t address, uint8_t reg, uint8_t size, uint8_t* data)** – odczyt bloku rejestrów ze SLAVE:

address – adres urządzenia na magistrali I^{2}C (akcelerometr 0x1D),

reg – adres pierwszego, wewnętrznego rejestru urządzenia,

size – ilość kolejnych rejestrów do odczytania (wartość musi być większa od 1),

*data – wskaźnik tablicy, w której będą zapisane dane z rejestrów, począwszy od „reg" a skończywszy na „reg+size-1".

Instrukcja zwraca kod błędu, w przypadku niepowodzenia:

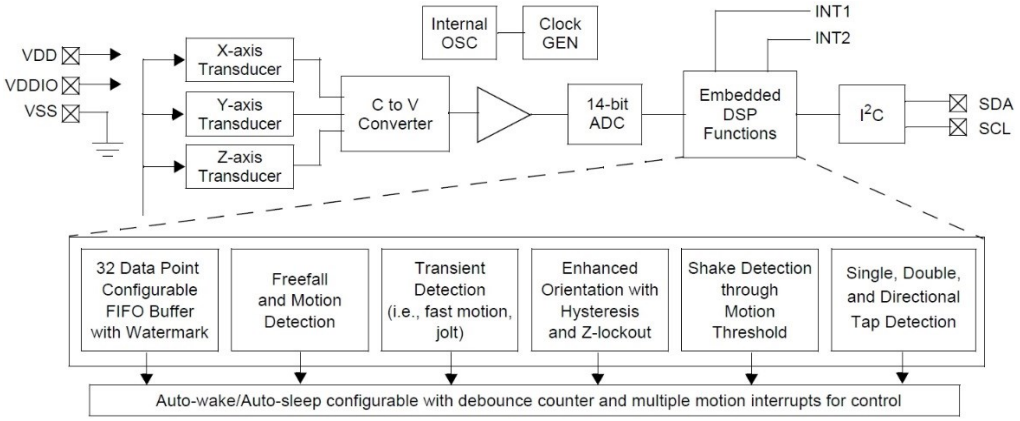
1 – przeterminowanie (za długi czas obsługi),

2 – brak potwierdzenia ACK,

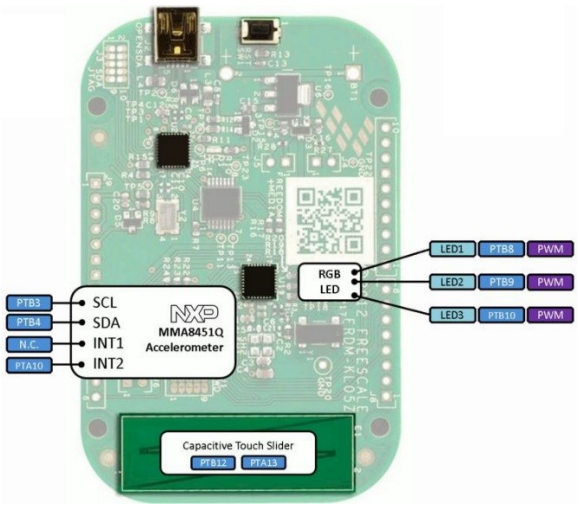
3 – obydwa błędy równocześnie.

5. AKCELEROMETR MMA8451

Układ MMA8451 jest 3-osiowym akcelerometrem cyfrowym (Rys. 2), będącym na wyposażeniu modułu KL05Z (Rys. 3).



Rys. 2 Schemat blokowy układu MMA8451

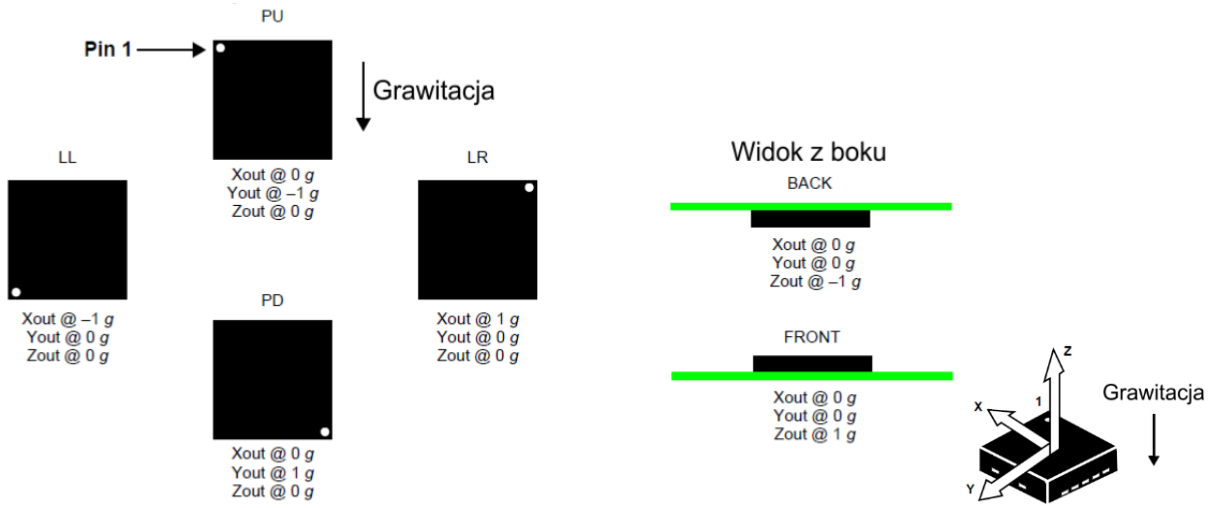


Rys. 3 Położenie akcelerometru na module KL05Z

Układ MMA8451Q służy do pomiaru przyspieszenia liniowego i jest zbudowany w oparciu o mikromaszyny (MEMS). Pomiar dla każdej osi (X,Y,Z) może być z rozdzielczością 14-to lub 8-io bitową. Ze światem zewnętrznym komunikuje się za pomocą portu I^2C, dla którego jest widziany jako struktura rejestrowa.

Jeżeli chodzi o czułość, to są do wyboru trzy zakresy: ± 2g, ± 4g i ± 8g. Jego możliwości to:

- wykrywanie swobodnego upadku lub ruchu, z informacją o jego kierunku
- wykrywanie pojedynczych i podwójnych uderzeń, z informacją o ich kierunku,
- wykrywanie wstrząsów i szybkich ruchów, z informacją o ich kierunku,
- wykrywanie orientacji (portret / krajobraz) z programowalną histerezą.



5.1. Programowanie (tylko podstawowe funkcje)

Układ posiada adres I^2C równy 0x1D. Jego rejestry mogą być odczytywane lub zapisywane pojedynczo lub blokowo.

1) W rejestrze konfiguracyjnym CTRL_REG1 (adres wewnętrzny 0x2A) ustawiamy bit ACTIVE=0 (stan czuwania).

0x2A: CTRL_REG1 register (read/write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ASLP_RATE1	ASLP_RATE0	DR2	DR1	DR0	LNOISE	F_READ	ACTIVE

2) W rejestrze konfiguracyjnym XYZ_DATA_CFG (adres wewnętrzny 0xE) ustawiamy czułość na ±2g (bity FS0=FS1=0).

0x0E: XYZ_DATA_CFG (read/write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	HPF_OUT	0	0	FS1	FS0

3) W rejestrze konfiguracyjnym CTRL_REG1 (adres wewnętrzny 0x2A) ustawiamy bit ACTIVE=1 (stan aktywny).

Na tym kończy się podstawowe programowanie.

Normalnie powinniśmy wykorzystać przerwania do obsługi danych. Jednak ograniczymy się do prostego odpytywania układu, czy dane są gotowe do odczytu. W tym celu należy odczytać rejestr stanu STATUS (adres wewnętrzny 0x0) i sprawdzić bit ZYXDR.

0x00 STATUS: Data status register (read only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ZYXOW	ZOW	YOW	XOW	ZYXDR	ZDR	YDR	XDR

Jeśli dane są gotowe, to można je odczytać blokowo (wszystkie w jednym cyklu transferu na magistrali I^2C). Wybrany jest mod pracy 14-bitowej, więc wystarczy podać adres pierwszego rejestru danych i wykonać sześć odczytów. Wewnętrzne adresy rejestrów danych są automatycznie zwiększane o 1 po każdym odczycie pojedynczego bajta. Po odczycie ostatniego bajta adres jest automatycznie ustawiany na zero, tak aby znowu wskazywał na rejestr stanu STATUS.

Do odczytu blokowego podaje się adres pierwszego rejestru danych, którym jest OUT_X_MSB (adres wewnętrzny 0x1), a ilość rejestrów do odczytu 6 (patrz funkcje z punktu 4.1.2).

0x01: OUT_X_MSB: X_MSB register (read only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
XD13	XD12	XD11	XD10	XD9	XD8	XD7	XD6

0x02: OUT_X_LSB: X_LSB register (read only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
XD5	XD4	XD3	XD2	XD1	XD0	0	0

0x03: OUT_Y_MSB: Y_MSB register (read only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
YD13	YD12	YD11	YD10	YD9	YD8	YD7	YD6

0x04: OUT_Y_LSB: Y_LSB register (read only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
YD5	YD4	YD3	YD2	YD1	YD0	0	0

0x05: OUT_Z_MSB: Z_MSB register (read only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ZD13	ZD12	ZD11	ZD10	ZD9	ZD8	ZD7	ZD6

0x06: OUT_Z_LSB: Z_LSB register (read only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ZD5	ZD4	ZD3	ZD2	ZD1	ZD0	0	0

Proszę zauważyć, że dana 14-bitowa jest „dosunięta” do lewej strony, w kierunku MSB. Aby uzyskać poprawną daną, po odczycie, starszy bajt należy przesunąć w lewo o 8 pozycji, dodać do młodszego bajta, a następnie całość przesunąć o 2 pozycje w prawo. Uzyskujemy w ten sposób jedną, integralną, 14-bitową wartość, z dwóch 8-bitowych. Wszystkie dane pobrane z czujnika są liczbami ze znakiem, zapisane w kodzie U2. Dlatego najlepiej jest deklarować typ

tych danych jako *int16_t* (liczba całkowita ze znakiem), a w czasie dostosowywania do zakresu (skalowania) rzutować na typ *double*.

Ostatnią czynnością, w celu otrzymania realnej wartości przyspieszenia, jest przeskalowanie danej, zgodnie z wybranym zakresem (czułością). Otrzymaną wcześniej 14-bitową daną należy podzielić przez:

4096 dla zakresu 2g

2048 dla zakresu 4g

1024 dla zakresu 8g.

Wynik jest gotowy do wykorzystania. Jest to wartość wyrażona w jednostkach przyspieszenia ziemskiego „g".

6. Ćwiczenie nr 1 – szukanie urządzeń podpiętych do portu I2C0

Rozpakować zbiór *1_I2C0_Ping.zip*. Uruchomić projekt *I2C0_Ping.uvprojx*. Na ekranie wyświetlacza LCD pojawią się adresy (wartość hex) wszystkich urządzeń, które są podpięte do portu I2C0. Powinny pojawić się dwa adresy: 0x1D (akcelerometr) i 0x3F lub 0x2F (ekspander wyświetlacza LCD).

6.1. Zadanie

- ✓ Przeanalizować działanie programu.

7. Ćwiczenie nr 2 – pomiar przyspieszenia w trzech osiach XYZ

Rozpakować zbiór *2_I2C0_XYZ.zip*. Uruchomić projekt *I2C0_XYZ.uvprojx*. Na wyświetlaczu pojawią się wyniki pomiaru przyspieszenia w trzech osiach: X, Y i Z. Dane są wyrażone w jednostkach przyspieszenia ziemskiego „g".

7.1. Zadanie

- ✓ Przeanalizować działanie programu.

8. Ćwiczenie nr 3 – sterowanie gestami

Rozpakować zbiór *2_I2C0_Gest.zip*. Uruchomić projekt *I2C0_Gest.uvprojx*. W zależności od położenia płytki, zaświecają się odpowiednie diody LED. Dla osi X dioda czerwona, dla osi Y dioda zielona, a dla Z niebieska. Świeci tylko dioda, charakterystyczna dla danej osi.

8.1. Zadanie

- ✓ Przeanalizować działanie programu.
- ✓ Zmodyfikować program tak, aby odpowiednie diody gasły po wykryciu przekroczenia przyspieszenia w danej osi powyżej 1.5g.