**ВСТУП**

Дана робота виконана на основі завдання до курсової роботи, виданого кафедрою інфомаційно-вимірювальних технологій НТУУ «КПІ ім. Ігоря Сікорського». Курсова робота присвячена розробці системи вимірювання віброприскорення відокремлюваного елементу.

На данний момент дуже нагальним є питання щодо випробувань різних елементів техніки на стійкість до можливих деформацій, ударів, перевантажень тощо. Одна з основних проблем подібних вимірів полягає у власній спроможності вимірювального каналу протистояти руйнівним факторам. Не менш важливий аспект – споживання енергії. Так як подібні системи часто використовуються у ракетобудуванні та суміжних галузях – важливим чинником є низьке споживання електроенергії, адже розмір та вага акумуляторної батареї має бути якомога меншою. Важливу роль грають такі характеристики, як точність вимірювань, та їх швидкодія. Системи екстремальних випробувань (Crash-системи) створюються на базі акселерометра з додаванням потрібних додаткових датчиків, таких як гіроскоп, та інші. Використання таких систем варіюється в доволі широкому спектрі – це може бути авіакосмічна галузь, різні види наземного транспорту: автомобілі, потяги, та будь-які інші системи, які потребують моніторингу руйнівного впливу фізичних чинників.

1. **ОГЛЯД МЕТОДІВ І ВІДОМИХ ТЕХНІЧНИХ РІШЕНЬ**
   1. Особливості мікроконтролера MSP430-G2553

Мікроконтролер MSP430-G2553 - 16-ти розрядний мікроконтролер фірми Texas Instruments (www.ti.com). Американська фірма TI є світовим лідером з виробництва цифрових сигнальних процесорів. Нею також виробляються мікроконтролери індустріального застосування, аналогові і цифрові мікросхеми широкого застосування [1].

Мікроконтролер MSP430-G2553 має 16-розрядний процесорний ядро. Як і AVR мікроконтролери, MSP реалізує RISC (Reduced Instruction Set Computer) ідеологію (всього 27 базових команд). Основними рисами розглянутого мікроконтролера є широкий набір вбудованих периферійних пристроїв, висока продуктивність процесора і вкрай низька потужність споживання, що робить його ідеальними для застосування в портативних вимірювальних індустріальних приладах. На рис. 1.1 наведено функціональну схему MSP430-G2553 [2].

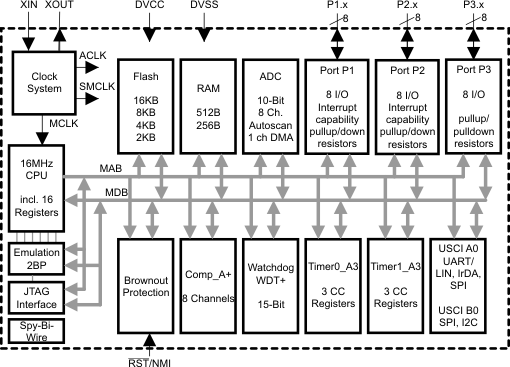


Рисунок 1.1 – Функціональна схема MSP430-G2553

Внутрішня заводська програма підтримує режими внутрішньосхемного послідовної завантаження і налагодження (через UART), а також режим одновиводной емуляції. Дана серія побудована за фон-неймановскою (von Neumann) архітектурі, яка передбачає, що пам'ять програм і пам'ять даних знаходяться в одному адресному просторі.

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**.**

**5 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

5.1 Вибір середовища розробки програмного забезпечення

Середовищем інструментальної розробки для обраного мікроконтролера є Code Composer Studio. На рис.5.1 зображено зовнішній вигляд середовища розробки Code Composer Studio.

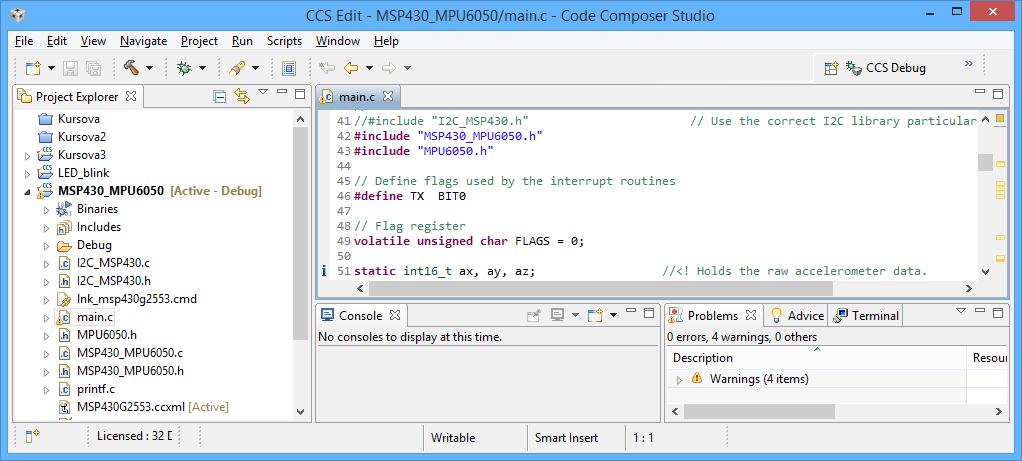


Рисунок 5.1 – Зовнішній вигляд середовища розробки Code Composer Studio

Інструментальні засоби розробки Code Composer Studio включають C і EC (Embedded C) компілятори, асемблери, відладчики та симулятори і інтегровані середовища розроблення. Code Composer Studio розробляє і виробляє засоби розробки для наступних промислових стандартів:

* MSP430 усіх сімейств;
* Infineon C16x/XC16x і ST-Micro ST10/Super10;
* ARM7 і ARM7TDMI;
* 8051 Classic і Extended;
* Philips LPC;
* 251 Atmel, Intel і Sanyo.

Code Composer Studio підтримує усі різновиди вищевказаних мікроконтролерів і усі стадії розробки програми: створення вихідного файла на C/EC або асемблері, трансляцію, виправлення помилок, лінкування об'єктних файлів, тестування програми (рис. 5.2).

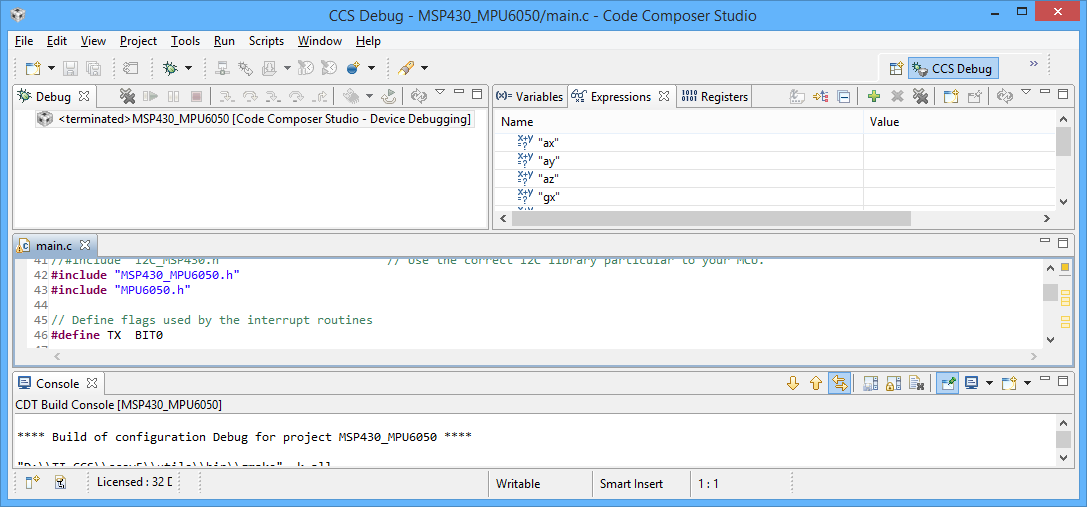


Рисунок 5.2 – Зовнішній вигляд режиму дебагера у Code Composer Studio

Code Composer Studio підтримує всі стадії розробки програми: створення вихідного файлу на С або Асемблері, трансляцію, виправлення помилок, лінкування об'єктних файлів, тестування програми.

5.2 Розробка програмного забезпечення для MSP430G2553

Дане програмне забезпечення буде розроблене на мові програмування EC (Embedded C). Для створення програмного забезпечення, для мікроконролера MSP430G2553 у середовищі Code Composer Studio, треба до проекту підключити відповідну бібліотеку:

#include "msp430g2553.h" ;

Для ініціалізації, та підключення датчика прискорення до мікроконтролера необхідно включити бібліотеки "MSP430\_MPU6050.h" та "MPU6050.h":

#include "MSP430\_MPU6050.h";

#include "MPU6050.h";

За стандартом мови програмування С, кожна з відокремлених функцій повинна мати прототип цієї функції:

void sendByte(unsigned char);

void printf(char \*, ...);

void initTimer(void);

void initUART(void);

void sendString(char\*);

Функція void sendByte(unsigned char) призначена для надсилання одного байту за допомогою UART:

void sendByte(unsigned char byte )

{

while (!(IFG2&UCA0TXIFG));

UCA0TXBUF = byte;

}

Функція void printf(char \*, ...) надсилає рядок за допомогою UART:

void printf(char \*format, ...)

{

char c;

int i;

long n;

va\_list a;

va\_start(a, format);

while(c = \*format++) {

if(c == '%') {

switch(c = \*format++) {

case 's': // String

puts(va\_arg(a, char\*));

break;

case 'c': // Char

putc(va\_arg(a, char));

break;

case 'i': // 16 bit Integer

case 'u': // 16 bit Unsigned

i = va\_arg(a, int);

if(c == 'i' && i < 0) i = -i, putc('-');

xtoa((unsigned)i, dv + 5);

break;

case 'l': // 32 bit Long

case 'n': // 32 bit uNsigned loNg

n = va\_arg(a, long);

if(c == 'l' && n < 0) n = -n, putc('-');

xtoa((unsigned long)n, dv);

break;

case 'x': // 16 bit heXadecimal

i = va\_arg(a, int);

puth(i >> 12);

puth(i >> 8);

puth(i >> 4);

puth(i);

break;

case 0: return;

default: goto bad\_fmt;

}

} else

bad\_fmt: putc(c);

}

va\_end(a);

}

Функція void initTimer(void) ініціалізує таймер для виклику переривання кожну секунду:

void initTimer(void) {

P1DIR |= BIT0; // set P1.0 (LED1) as output

P1OUT |= BIT0; // P1.0 low

CCTL0 = CCIE; // CCR0 interrupt enabled

CCR0 = 4096; // 32kHz/8/4096 -> 1 sec

TACTL = TASSEL\_1 + ID\_3 + MC\_1; // ACLK, /8, upmode

}

Функція void initUART(void) ініціалізує UART для подальшої роботи з ним:

void initUART(void) {

P1SEL = BIT1 + BIT2 ; // P1.1 = RXD, P1.2=TXD

P1SEL2 = BIT1 + BIT2 ; // P1.1 = RXD, P1.2=TXD

UCA0CTL1 |= UCSSEL\_1; // CLK = ACLK

UCA0BR0 = 0x03; // 32kHz/9600 = 3.41

UCA0BR1 = 0x00;

UCA0MCTL = UCBRS1 + UCBRS0; // Modulation UCBRSx = 3

UCA0CTL1 &= ~UCSWRST; // \*\*Initialize USCI state machine\*\*

IE2 |= UCA0RXIE; // Enable USCI\_A0 RX interrupt

}

Функція void sendString(char\*) надсилає посимвольно отриманий рядок до буферу UART:

void sendString(char \* text) {

int i = 0;

for (i = 0; i < strlen(text); i++) {

while (!(IFG2 & UCA0TXIFG))

; // Если буфер для отправки готов

UCA0TXBUF = text[i]; // Отправляем очередной символ из строки

}

}

Вектори переривання:

#pragma vector=USCIAB0RX\_VECTOR

\_\_interrupt void USCI0RX\_ISR(void)

{

char r = UCA0RXBUF; // Get the received character

if (r == 't') // 'u' received?

{

FLAGS |= TX; // Set flag to transmit data

\_\_bic\_SR\_register\_on\_exit(LPM3\_bits); // Wake-up CPU

}

}

#pragma vector=TIMER0\_A0\_VECTOR

\_\_interrupt void Timer\_A (void)

{

P1OUT ^= BIT0; // Toggle LED

}

Функції перетворення отриманого рядка, перед надсиланням по UART:

static const unsigned long dv[] = {

// 4294967296 // 32 bit unsigned max

1000000000, // +0

100000000, // +1

10000000, // +2

1000000, // +3

100000, // +4

// 65535 // 16 bit unsigned max

10000, // +5

1000, // +6

100, // +7

10, // +8

1, // +9

};

static void xtoa(unsigned long x, const unsigned long \*dp)

{

char c;

unsigned long d;

if(x) {

while(x < \*dp) ++dp;

do {

d = \*dp++;

c = '0';

while(x >= d) ++c, x -= d;

putc(c);

} while(!(d & 1));

} else

putc('0');

}

static void puth(unsigned n)

{

static const char hex[16] = { '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};

putc(hex[n & 15]);

}

Функція puts() використовується функцією printf() для виведення рядку:

void puts(char \*s) {

char c;

// Loops through each character in string 's'

while (c = \*s++) {

sendByte(c);

}

}

Функція putс() використовується функцією printf()для виведення символу:

void putc(unsigned b) {

sendByte(b);

}

Функція main(void) є головною функцією програми. Саме з неї починається виконання всієї програми.

int main(void)

{

WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer

BCSCTL1 = CALBC1\_1MHZ; // 16MHZ operation

DCOCTL = CALDCO\_1MHZ;

//Placed 10k or 4.7k ohm pullup resistors on SDA and SCL lines

//P3DIR |= 0x0F;

//P3SEL |= 0x06; // Assign I2C pins to USCI\_B0

P1DIR |= BIT0; // P1.0 output

P1SEL |= BIT6 + BIT7; // Assign I2C pins to USCI\_B0

P1SEL2|= BIT6 + BIT7;

initializeIMU(); // Initialize IMU

msDelay(30); // Temporary wait. Can be shortened

printf("\r\n %s \r\n", "Motion");

for (;;)

{

getMotion6( &ax, &ay, &az, &gx, &gy, &gz);

// ax, ay, az, gx, gy, gz have RAW values from

// accelerometer and gyroscope

printf("%i %i %i %i %i %i\r\n", ax, ay, az, gx, gy, gz);

printf("Print working..?");

msDelay(100);

char buf[60]; //Буфер для символов

sprintf(buf, "Hello from UART!");

sendString(buf); //отправляем буфер по UART

}

}

У наведеному лістингу відбувається ініціалізація датчика MPU-6050, налаштування параметрів обміну даними за протоколом передачі даних І2С та протоколом передачі даних UART, ініціалізація мікроконтролера MSP430G2553, зчитування даних з датчика прискорення за допомогою інтерфейсу передачі даних І2С та подальше їх надсилання за протоколом передачі даних UART. Для цього викристовується бібліотека printf.c, що спрощує обмін даними за допомогою UART.

**.**

**.**

**.**

**ВИСНОВКИ**

У результаті виконання курсової роботи було розроблено блок вимірювання віброприскорення відокремлюваного елементу. Наведено огляд пристроїв подібного класу і призначення, вибрано сучасні датчики для вимірювального каналу.

У курсовій роботі представлений та обґрунтований вибір структурної схеми пристрою, на основі якого розроблена функціональна схема. Відповідно до функціональної схеми було розроблено принципову схему модуля. Також проаналізовано та розраховано похибки блоку вимірювання віброприскорення відокремлюваного елементу.

**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. Хоровиц П., Хилл У. Искусство схемотехники: в 3-х томах. Пер. з англ. Вид. 4-е. Перероблене і доповнене. М:. Мир, 1993, 413с.

2. Аш Ж. з співавторами Датчики измерительных систем: в 2-х томах. Том 1. Пер. з франц. – М:. Мир, 1992, 480с.

3. Магда Ю. С. Микроконтроллеры серии 8051: практический подход. — М.: ДМК Пресс, 2008. — 228 с.

4. Николайчук О. х51-совместимые микроконтроллеры — М.: ООО «ИД СКИМЕН», 2002 — 472с.

5. ADuC8xx SAR EvalGuide – Analog Devices, Inc., 2003, 10c.

6. Debug ADUC with Different Keil – Analog Devices, Inc., 2010, 9c.

7. ADuC841/ADuC842/ADuC843 Datasheet.– Analog Devices, Inc., 2003, 88c.

8. ADM1485 Datasheet.– Analog Devices, Inc., 2012, 11c.

9. ADuC8xx Evaluation Kit Getting Started User Guide.– Analog Devices, Inc., 2004, 16c.

10 H11L1 Datasheet.– Motorola, Inc., 1995, 7c.

11. SFH5711 Datasheet.– Opto Semiconductors, Inc., 2013, 13c.

12. Дж. Пейтон «Аналогова електроніка на операційних підсилювачах» - М. Біном 1994, 352с.

13. Методичні вказівки до дипломного проектування для студентів за напрямком 6.051.001"Метрологія та інформаційно-вимірювальні технології"/ Укладачі: Є.Т. Володарський, Є.В.Грещенко, В.І. Губар, В.В. Літвіх – Київ : НТУУ (КПІ ), - 40с.- Укр. мовою.