A

Report

On

“Embedded System Design Using MSP430.”

Submitted By:

Arpit Shrivastava (312059)

Durgesh Vitore (312070)

Tejas Shinde (312058)

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING,
VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY,
PUNE– 411048
YEAR 2018-2019

Introduction to MSP430:-

The MSP microcontrollers (MCUs) from Texas Instruments (TI) are 16-bit and 32-bit RISC-based, mixed signal processors designed for ultra-low power applications. The MSP430 family of microcontrollers consists of several devices featuring different sets of peripherals targeted for various applications. The MSP430 device features a powerful 16-bit RISC CPU, 16-bit registers, and constant generators that contribute to maximum code efficiency. The architecture, which includes five low-power modes, is optimized to achieve extended battery life in portable measurement applications. The digitally controlled oscillator (DCO) allows wake-up from low-power modes to active mode in less than 1 microsecond.

The MSP430 instruction set consists of 27 core instructions that make it easy to program in assembler or in C, and provide exceptional flexibility and functionality. MSP low-power + performance microcontrollers from TI provide designers with enhanced performance and more peripherals on chip while using less power. MSP ultra-low-power microcontrollers are ideal for applications where the majority of the microcontroller life is spent in standby. Typical applications of these processors include low-cost sensor systems that capture analog signals, convert them to digital values, and then process the data for display or for transmission to a host system.

The architecture of the MSP430 family is based on a memory-to-memory architecture, a common address space for all functional blocks, and a reduced instruction set applicable for all functional blocks. The MSP430G2553 CPU has a 16-bit RISC architecture integrated with 16 registers that provide reduced instruction execution time. All operations, other than program-flow instructions, are performed as register operations. Memory of the MSP430G2553 is divided to program memory (ROM) and data memory (RAM). The program memory is accessed by the 16-bit Memory Data bus. The data memory is accessed by the Memory Address Bus (MAB) and the Memory Data Bus (MDB).

Peripheral modules such as Digital I/O, Analog-to-Digital Converter (ADC), Timers, Universal Asynchronous Receiver Transmitter (UART), Universal Serial Communications Interface (USCI), Comparator A+, etc., are connected to the CPU via MAB, MDB and interrupt service and request lines. The operations of the various peripherals are controlled by the contents of the Special Function Registers (SFRs).

The **clock system** is supported by the basic clock module that includes support for a 32768-Hz watch crystal oscillator, an internal very-low-power, low-frequency oscillator and an internal Digitally Controlled Oscillator (DCO). The brownout circuit is implemented to provide the proper internal reset signal to the device during power on and power off.

The **brownout** circuit is implemented to provide the proper internal reset signal to the device during power on and power off.

The **Watchdog Timer** (WDT+) module restarts the system on occurrence of a software problem or if a selected time interval expires. The WDT can also be used to generate timely interrupts for an application.

So some of the main features of MSP430 includes:

Low supply voltage range: 1.8 V to 3.6 V

Low-power consumption:

220 A / MHz active mode.

0.7 A real-time clock mode.

A RAM retention.

16-bit RISC Architecture:

Basic clock module configurations:

Internal frequencies up to 16 MHz with four calibrated frequency

Serial communication modules:

USART

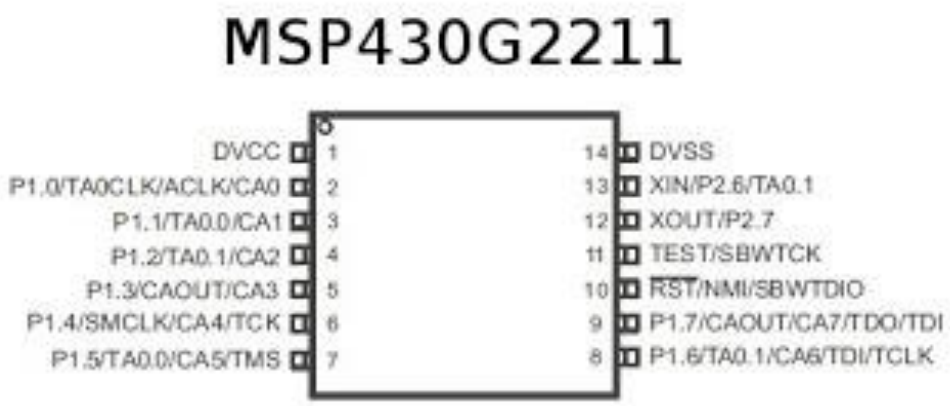
USI

Memory:

Up to 56 KB Flash

Up to 4 KB SRAM

MSP430 Pinout Diagram:



Code Composer Studio (Software):-

Code Composer Studio (CCS) is the integrated development environment for the whole span of TI Microcontrollers, DSPs and application processors. Code Composer Studio includes a suite of tools used to develop and debug embedded applications. For all device families, CCS includes compilers, source code editor, project build environment, debugger, profiler, simulators and many other features. Code Composer Studio v6 is the latest version of the IDE used currently and is based on Eclipse 4.3, whereas the earlier version CCSv5 was based on the Eclipse 3.x stream.

The CCS App center, introduced in CCSv6, lists all the relevant packages for the chosen devices during the installation. The Getting Started View, which is open by default when CCS is first launched, provides fast access to many of the common tasks the users would prefer to experiment when starting to use a new environment, such as creating a new project, browsing examples and visiting the App Center. There are also links to the support forums, YouTube videos, training material and the wiki. There is also a video that is prominently displayed in the center of the screen that walks users through the steps to use the CCS environment

The CCS supports all the phases of the development cycle: design, code and build, debug. The CCS includes all the development tools - compilers, assembler, linker, debugger, BIOS and one target - the Simulator

CSS Functional Overview:-

- The C compiler converts C source code into assembly language source code. The assembler translates assembly language source files into machine language object files.
- The standard run-time libraries contain ANSI standard run-time-support functions, compiler-utility functions, floating-point arithmetic functions and I/O functions that are supported by the C compiler.
- The linker combines object files, library files and system or BIOS configuration files into a single executable object module. The .out file is the executable program for the target device. etc. The .gel files initialize the debugger with address of memory, peripherals and other hardware setup details.

Getting Started With Programming:-

Project Creation and Build –

1. Click File New CCS Project
2. Type in your “Project name” and make the selections shown below (your dialog may look slightly different than this one).
3. If you are using the MSP430G2553, make the appropriate choices for that part. Make sure to click Empty Project (with main.c) and then click Finish.
4. If you are writing an Assembly Code, select Empty Assembly-only Project.
5. Code Composer will add the named project to your workspace and display it in the Project Explorer pane. Based on your template selection, it will also add a file called main.c/ main.asm and open it for editing.
6. Change the include header file statement from #include to #include .
7. Type in your program code in the main.c / main.asm file and save it.
8. CCS can automatically save modified source files, build the program, open the debug perspective view, connect and download it to the target (flash device), and then run the program to the beginning of the main function. To do this, click on the "Debug" button.
9. When the Ultra-Low-Power Advisor (ULP Advisor) appears, click the Proceed button.
10. When the download completes, CCS is in the Debug perspective. Notice the Debug tab in the upper right-hand corner indicating that we are now in the "CCS Debug" view.
11. Click and drag the perspective tabs to the left until you can see all of both tabs. You can observe that the program has run through the C-environment initialization routine in the runtime support library and stopped at main() in main.c.

Other Software Development Environment's:-

Texas Instruments provides various hardware experimenter boards that support large (approximately two centimeters square) and small (approximately one millimeter square) MSP430 chips. TI also provides software development tools, both directly, and in conjunction with partners (see the full list of compilers, assemblers, and IDEs). One such toolchain is the IAR C/C++ compiler and *Integrated development environment*, or IDE.

TI also combines a version of its own compiler and tools with its Eclipse-based Code Composer Studio IDE ("CCS"). It sells full-featured versions, and offers a free version for download which has a code size limit of 16 KB. CCS supports in-circuit emulators, and includes a simulator and other tools; it can also work with other processors sold by TI.

For those who are more comfortable with the Arduino, there is also another software Energia, an open source electronics prototyping platform with the goal to bring the Wiring and Arduino framework to the Texas Instruments MSP430 based LaunchPad where Arduino code can be exported for programming MSP430 chips.

The open source community produces a freely available software development toolset based on the GNU toolset.

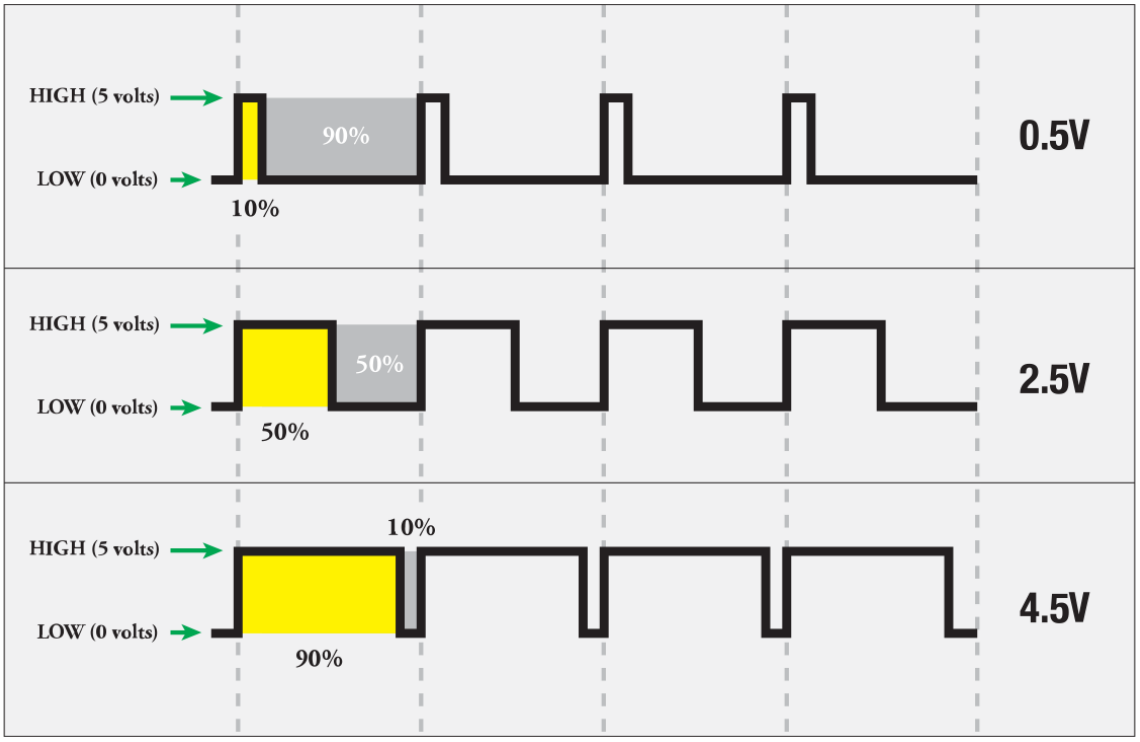
TI consulted with RedHat to provide official support for the MSP430 architecture to the GNU Compiler Collection C/C++ compiler. This msp430-elf-gcc compiler is supported by TI's Code Composer Studio version 6.0 and higher.

There is a very early llvm-msp430 project, which may eventually provide better support for MSP430 in LLVM.

Other commercial development tool sets, which include editor, compiler, linker, assembler, debugger and in some cases code wizards, are available. VisSim, a block diagram language for model based development, generates efficient fixed point C-Code directly from the diagram.[6] VisSim generated code for a closed loop ADC+PWM based PID control on the F2013 compiles to less than 1 KB flash and 100 bytes RAM.[7] VisSim has on-chip peripheral blocks for thentire MSP430 family I²C, ADC, SD16, PWM.

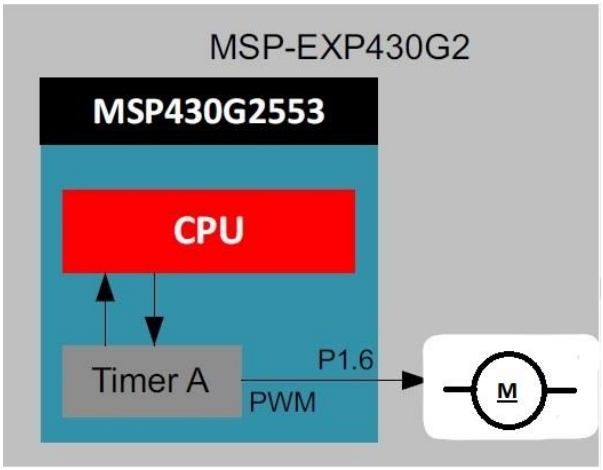
PWM Motor Control using MSP430:

Pulse Width Modulation (PWM) is a method of digitally encoding analog signal levels. High-resolution digital counters are used to generate a square wave of a given frequency, and the duty cycle of the square wave is modulated to encode the analog signal. The duty cycle determines the time during which the signal pulse is HIGH.



As from above figure it is clear that average Analog voltage increases with increase in Duty Cycle. More is the voltage across Motor more will be the speed.

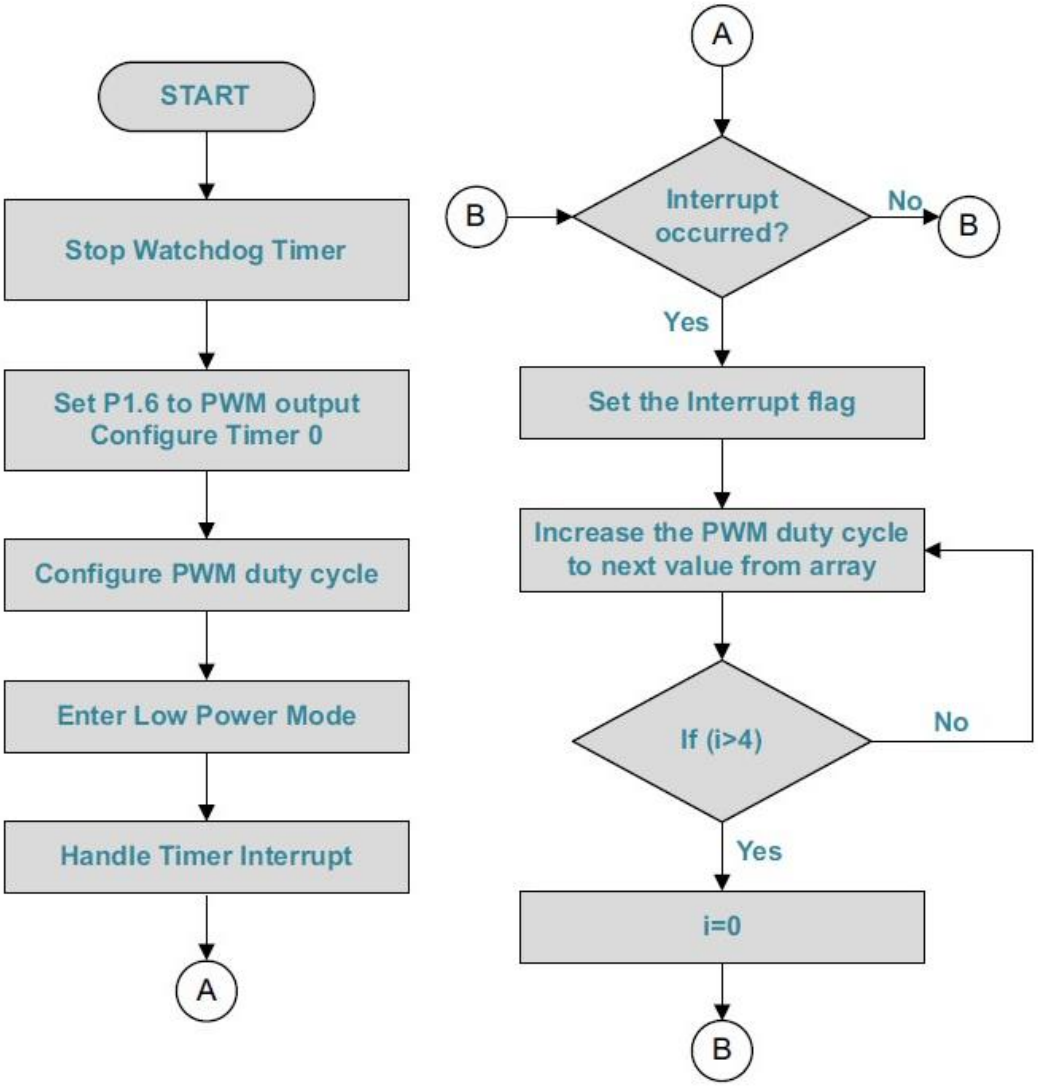
Interfacing Diagram:-



Hardware and Software requirements:-

MSP430G2 LaunchPad
Code Composer Studio.

Flowchart:-



C Program Code:

```
#include <msp430.h>
int a[5] = {0,32,64,128,255};
int i = 0;
void main(void){
WDTCTL = WDTPW|WDTHOLD;       // Stop WDT
//IE1 |= WDTIE;                 // enable Watchdog timer interrupts
P1DIR |= BIT6;                  // Motor for output
P1SEL |= BIT6;                  // Motor Pulse width modulation
```

```
TA0CCR0 = 512;                          // PWM period
TA0CCR1 = a[0];                         // PWM duty cycle
TA0CCTL1 = OUTMOD_7;                    // TA0CCR1 reset/set-high voltage
                                        // below count, low voltage when past
TA0CTL = TASSEL_2 + MC_1 + TAIE +ID_3;
                                        // Timer A control set to SMCLK, 1MHz
                                        // and count up mode MC_1
_bis_SR_register(LPM0_bits + GIE); // Enter Low power mode 0
while (1);
}
#pragma vector= TIMER0_A1_VECTOR        // Watchdog Timer ISR
__interrupt void timer(void) {
TA0CTL &= ~TAIFG;
TA0CCR1 = a[++i];
if (i>4)
{
i=0;
}
 }
```

Calculations for Duty Cycle:

Sl.No	TA0CCR1	TA0CCR0	Duty Cycle = (TA0CCR1/TA0CCR0) * 100
1.	0x00C9	0x03E8	20%
2.	0x02E5	0x03E8	74%
3.	0x03C1	0x03E8	96%

Conclusion:

The MSP430 Value Line controllers are cheaper than the higher-end ATMega controllers used in the Arduino platform. We don't need an external crystal, because the MSP430 controllers can run at the full 16 Mhz from their internal clock source. With the MSP430 Launchpad, TI offers a development board which includes:

A programmer for the MSP430 controllers.
Comes with two microcontrollers.
The USB connection also allows you to talk to the microcontroller via a serial connection at 9600 baud,