SVR ENGINEERING COLLEGE

Approved by AICTE & Permanently Affiliated to JNTUA

Ayyaluru metta, Nandyal–518503.Website: www.svrec.ac.in

Department of Electronics and Communication Engineering

(15A04607) MICROPROCESSORS AND MICROCONTROLLERS LABORATORY

B.Tech – III-II Sem

STUDENTNAME	
ROLLNUMBER	
SECTION	

SVR ENGINEERING COLLEGE

Approved by AICTE & Permanently Affiliated to JNTUA

Ayyaluru metta, Nandyal–518503.Website: www.svrec.ac.in

DEPARTMENT OF

ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

ACADEMIC YEAR : 2020-21

This is to certify that the bonafide record work done by

Mr./Ms._____

Bearing H.T.NO. _____ of III B.TECH II Semester in

the MICROPROCESSORS AND MICROCONTROLLERS LABORATORY.

Faculty In-Charge **Head of the Department**

ECE DEPT VISION & MISSION PEOs and PSOs

Vision

To produce highly skilled creative and competitive Electronics and Communication Engineers To meet the emerging need the society.

Mission

- ImpartcoreknowledgeandnecessaryskillsinElectronicsandCommunicationEngineeringthroughinnovative teaching and learning.
- Inculcatecriticalthinking,ethics,lifelonglearningandcreativityneededforindustryandsociety
- Cultivate the students with all-round competencies, for career, higher education and self-employability.

I. PROGRAMME EDUCATIONAL OBJECTIVES (PEOS)

PEO1: Graduates apply their knowledge of mathematics and science to identify, analyze and Solve problems in the field of Electronics and develop sophisticated communication systems.

PEO2: Graduate some body a commitment to professional ethics, diversity and social awareness in their professional career.

PEO3: Graduate exhibit a desire for life Long learning through technical training and professional activities.

II. PROGRAM SPECIFIC OUTCOMES (PSOS)

PSO1: Apply the fundamental concepts of electronics and communication engineering to design a variety of components and systems for applications including signal processing, image processing, communication, networking, embedded systems, and VLSI and control system.

PSO2: Select and apply cutting-edge engineering hardware and software tools to solve complex Electronics and Communication Engineering problems.

III. PROGRAMMEOUTCOMES(PO'S)

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/developmentofsolutions:Designsolutionsforcomplexengineeringproblemsanddesignsystemcomponents or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental Considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multi-disciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance : Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broad section text of technological change.

IV. COURSE OBJECTIVES:

- To Understand the students with basic programming of 8086 microprocessors.
- To Understand concepts of Intel x86 series of processors.
- Program MSP 430 for designing any basic Embedded System.
- Design and implement some specific real time applications Using MSP 430 low power microcontroller.
- To design and implement different Interfacing concept techniques.

V. COURSE OUTCOMES:

After the completion of the course students will be able to

Course Outcomes	Course Outcome statements	BTL
CO1	Understand Assembly Language Programming using MASM/ DOSBOX.	L1
CO2	Execute8086ALPsforarithmeticcalculations,strings,Test and Debug 8086ALPs.	L2
CO3	Understand Programming of MSP430 using Embedded C.	L3
CO4	Observe the performance of Low Power modes and Energy trace++	L4
CO5	Observe the performance of the various interfacing devices.	L5

VI. COURSE MAPPING WITH PO'S AND PEO'S:

CourseTitle	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
MICROPROCESSORS AND MICROCONTROLLERS LABORATORY	3	3	2	3	3	2	2	1	2	2	1	2	3	2

VII. MAPPING OF COURSE OUTCOMES WITH PEO'S AND PO'S:

Course Title	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
CO1	3	3	3	3	3	2	1	1	1	2	1	2	3	2
CO2	2	2	2	2	2	1	2	1	2	2	1	2	2	2
CO3	3	3	3	3	3	3	2		2	2	1		3	2
CO4	2	3	2	3	2	2	2	2	2	1	2	2	2	2
CO5	3	3	2	2	3	2	1	1	1	2		1	3	1

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

B.Tech – III-II Sem **L T P C**

0 0 4 2

15A04607 MICROPROCESSORS AND MICROCONTROLLERS LABORATORY

LIST OF EXPERIMENTS

Part A : 8086 Microprocessor Programs using MASM/8086 microprocessor kit.

1. Introduction to MASM Programming.
2. Programs using arithmetic and logical operations
3. Programs using string operations and Instruction prefix: Move Block, Reverse string, Sorting, String comparison
4. Programs for code conversion
5. Multiplication and Division programs
6. Sorting and multi byte arithmetic
7. Programs using CALL and RET instructions

Part B Embedded C Experiments using MSP430 Microcontroller

1. Interfacing and programming GPIO ports in C using MSP430 (blinking LEDs , push buttons)
2. Usage of Low Power Modes: (Use MSPEXP430FR5969 as hardware platform and demonstrate the low power modes and measure the active mode and standby mode current)
3. Interrupt programming examples through GPIOs
4. PWM generation using Timer on MSP430 GPIO
5. Interfacing potentiometer with MSP430
6. PWM based Speed Control of Motor controlled by potentiometer connected to MSP430 GPIO
7. Using ULP advisor in Code Composer Studio on MSP430
8. Low Power modes and Energy trace++:

a. Enable Energy Trace and Energy Trace ++ modes in CCS

b. Compute Total Energy, and Estimated lifetime of an AA battery.

LABORATORY RULES

General Rules of Conduct in Laboratories:

1. You are expected to arrive on time and not depart before the end of a laboratory.
2. You must not enter a lab unless you have permission from a technician or lecturer.
3. You are expected to comply with instructions, written or oral, that the laboratory Instructor gives you during the laboratory session.
4. You should behave in an orderly fashion always in the lab.
5. You must not stand on the stools or benches in the laboratory.
6. Keep the workbench tidy and do not place coats and bags on the benches.
7. You must ensure that at the end of the laboratory session all equipment used is stored away where you found it.
8. You must put all rubbish such as paper outside in the corridor bins. Broken components should be returned to the lab technician for safe disposal.
9. You must not remove test equipment, test leads or power cables from any lab without permission.
10. Eating, smoking and drinking in the laboratories are forbidden.
11. The use of mobile phones during laboratory sessions is forbidden.
12. The use of email or messaging software for personal communications during laboratory sessions is forbidden.
13. Playing computer games in laboratories is forbidden.

Specific Safety Rules for Laboratories:

1. You must not damage or tamper with the equipment or leads.
2. You should inspect laboratory equipment for visible damage before using it. If there is a problem with a piece of equipment, report it to the technician or lecturer. DONOT return equipment to a storage area
3. You should not work on circuits where the supply voltage exceeds 40 volts without very specific approval from your lab supervisor. If you need to work on such circuits, you should contact your supervisor for approval and instruction on how to do this safely before commencing the work.
4. Always use an appropriate stand for holding your soldering iron.
5. Turn off your soldering iron if it is unlikely to be used for more than 10 minutes.
6. Never leave a hot soldering iron unattended.
7. Never touch a soldering iron element or bit unless the iron has been disconnected from the mains and has had adequate time to cool down.
8. Never strip insulation from a wire with your teeth or a knife, always use an appropriate wire stripping tool.
9. Shield wire with your hands when cutting it with a pliers to prevent bits of wire flying about the bench.

Do's

1. Proper dress code has to be maintained while entering in to the Lab.
2. Students should carry observation notes and record completed in all aspects.
3. Assembly level program and its theoretical result should be there in the observation before coming to the next lab.
4. Student should be aware of next ALPs.
5. Students should be at their concerned desktop/bench, unnecessary moment is restricted.
6. Student should follow the procedure to start executing the ALP they have to get signed by the Lab instructor for theoretical result then with the permission of Lab instructor they need to switch on the desktop and after completing the same they need to switch off and keep the chairs properly.
7. After completing the ALP Students should verify the ALP by the Lab Instructor.
8. The Practical Result should be noted down into their observations and result must be shown to the Lecturer In-Charge for verification.
9. Students must ensure that all switches are in the OFF position, desktop is shut down properly.

Don'ts

1. Don't come late to the Lab.
2. Don't leave the Lab without making proper shut down of desktop and keeping the chairs properly.
3. Don't leave the Lab without verification by Lab instructor. 4. Don't leave the lab without the permission of the Lecturer In-Charge.

NDEX

Mr./Ms._____ Roll Number:_____

S. No	Date	Title of the Experiment	Text Book	Page No.	Marks	Signature of the Staff
PART-A						
1		Introduction MASM Software				
2		Introduction To MSP 430 Microcontroller				
PART – A : LIST OF EXPERIMENTS						
1		Programs using Arithmetic and logical operations: Addition, Subtraction, Logical	T1: 51-60	13		
2		Programs using string operations and instruction prefix: Move Block, Reverse string, String Comparison	T1: 61-62	20		
3		Programs for code conversion: BCD to Seven Segment Code Conversion, Binary to Gray Code conversion, BCD to ASCII	T1: 51-60	27		
4		Multiplication and division programs: Multiplication, Division	T1: 51-60	34		
5		Sorting and MultiByte arithmetic: Ascending order / Descending order, MultiByte addition / subtraction	T1: 51-60	40		
6		Programs using CALL and RET instructions: Factorial of a number using procedure, Finding prime number or not, Fibonacci series generation.	T1: 63-67	46		

7		Interfacing and Programming GPIO Ports	T2: 208	55		
8		Usage of Low Power Modes	T2: 198	61		
9		Interrupt programming through GPIOs	T2: 186	67		
10		PWM generation using Timer on MSP430 GPIO	T2: 330	71		
11		Interfacing Potentiometer with MSP430	T2: 393	75		
12		PWM based Speed Control of Motor by Potentiometer	T2: 393	79		
13		Using ULP Advisor in CCS on MSP430		84		
14		Low Power modes and Energy Trace++	T2: 198	88		
Additional Experiment Beyond the Curriculum						
15		Serial Communication using MSP430	T2: 574	93		
PART C						
16		Lab Based Mini Project		99		

TEXT BOOKS:

T1: A.K.Ray and K.M. Bhurchandi, "Advanced Microprocessors and Peripherals", 3rd Edition, TMH Publications.

T2: MSP430 Microcontroller Basics- John H. Davie

Part – A Module-1

Introduction to MASM 8086Software

There are two ways to execute 8086 programs using a computer. One is using the DOS tool called DEBUG and the second way is using the Assembler MASM. Hence both the ways are to be learned.

DEBUG

The DOS utility called DEBUG allows entering assembly language programs, to execute these programs, to view memory locations and also to trace the program execution.

Starting and Quitting DEBUG

The simple way of starting the DEBUG is to type the command DEBUG at the DOS prompt. After the DEBUG is loaded the DEBUG prompt '-' is displayed. Then the DEBUG is ready to accept any DEBUG commands.

To quit DEBUG the command is Q. This command takes control back to DOS. DEBUG can be started along with a file also. To load DEBUG with a program say test1.com at the DOS prompt type DEBUGtest1.com.

DEBUG Commands

The commands which are required to run / enter the programs are the following.

A	Assemble
D	Display /Dump
E	Enterdata
F	Fillmemory
G	Run theprogram
L	Load
N	Name aprogram
P	Proceed
Q	Quit
R	Register
T	Trace
U	UnAssemble
W	Write

The DEBUG does not detect errors until Enter key is pressed. If there is any syntax mistake then the command line is redisplayed with the word error added at the point at which the error was detected. If a command line contains more than one error, only the first error will be detected and indicated and execution will stop at that point.

Part – A Module-1

Introduction to Assembly LanguageProgramming

LEVELS OF PROGRAMMING:

There are three levels of programming

1. Machine language
2. Assembler language
3. High level language

Machine language programs are programs that the computer can understand and execute directly. Assembly language instructions match machine language instructions, but are written using character strings so that they are more easily understood and High-level language instructions are much closer to the English language and are structured.

Ultimately, an assembly language or high level language program must be converted into machine language by programs called translators. If the program being translated is in assembly language, the translator is referred to as an assembler, and if it is in a high level language the translator is referred to as a compiler or interpreter.

ASSEMBLY LANGUAGE PROGRAM DEVELOPMENT TOOLS:

EDITOR: An editor is a program, which allows you to create a file containing the assembly language statements for your program.

ASSEMBLER: An assembler program is used to translate the assembly language Mnemonic instructions to the corresponding binary codes. The second file generated by assembler is called the assembler List file.

LINKER: A Linker is a program used to join several object files in to one large object file. The linkers produce link files with the .EXE extension.

DEBUGGER: If your program requires no external hardware, then you can use a debugger to run and debug your program. A debugger is a program, which allows you to load your object code program into system memory, execute the program, and troubleshoot or “debug” it.

Part – A Module-1

Introduction to 8086 - AssemblerDirectives

The logical errors or other programming errors are not found by the assembler. For completing all these tasks, an assembler needs some hints from the programmer. These types of hints are given to the assembler using some predefined alphabetical strings called assembler directives, which helps the assembler to correctly understand the assembly language program to prepare the codes.

Another type of hint which helps the assembler to assign a particular constant with a label or initialize particular memory locations or labels with constants is an operator.

- DB: Define Byte:The DB directive is used to reserve byte or bytes of memory locations in the available memory.
- DW: Define Word: The DW directive serves the same purposes as the DB directive, but it makes the assembler reserves the number of memory words (16bit) instead of bytes.
- DQ: Define Quad word: This directive is used to direct the assembler to reserve 4words (8bytes) of memory for the specified variable and may initialize it with the specified values.
- DT: Define Ten Bytes: The DT directive directs the assembler to define the specified variable requiring10-bytesforitsstorageandinitializethe10-byteswiththespecifiedvalues.
- ASSUME: Assume Logical Segment Name: The ASSUME directive is used to inform the assembler, the names of the logical segments to be assumed for different segments used in the program.
- END: END Of Program: The END directive marks the end of an assembly languageprogram.
- ENDP: END Of Procedure: The ENDP directive is used to indicate the end of aprocedure.
- ENDS: END of Segment: This directive marks the end of a logical segment. DATASEGMENT
  :
- EVEN: Align On Even Memory Address: The EVEN directive updates the location counter to the next even address, if the current location counter contents are not even, and assigns the following routine or variable or constant to that address. If the content of the location counter is already even, then the procedure will be assigned with the sameaddress.
- EQU: Equate: The directive EQU is used to assign a label with a value or symbol. The use of this directive is just to reduce the recurrence of the numerical values or constants in the program code.

  LABEL EQU 0500H

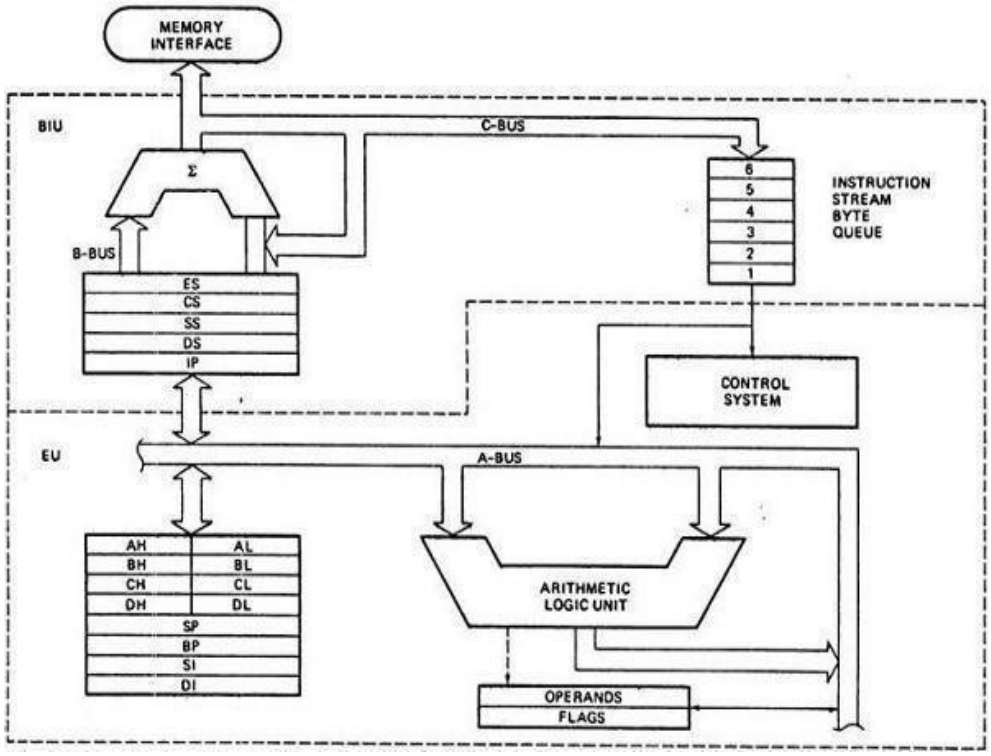
- EXTERN: External and PUBLIC: Public: The directive EXTERN informs the assembler that the names, procedures and labels declared after this directive have already been defined in some other assembly languagemodule.
- GROUP:GrouptheRelatedSegments:Thisdirectiveisused toformlogicalgroupsofsegments with similar purpose ortype.
- PROGRAM GROUP CODE, DATA, STACK //this statement directs the loader/linker to prepare an EXE file such that CODE, DATA, STACK segment must lie within a 64byte memory segment that is named asPROGRAM
- LABEL: The LABEL directive is used to assign a name to the current content of thelocation counter. A LABEL directive can be used to make a FAR jump. The label directive can be used to refer to the data segment along with the data type, byte or word.
- DATA SEGMENT
- DATAS DB 50H DUP(?)
- DATA-LAST LABEL BYTEFAR
- DATAENDS
- Afterreserving50HlocationsforDATAS,thenextlocationwillbeassignedalabelDATA-LAST and its type will be byte andfar.
- LENGTH:ByteLengthofaLabel:Thisdirectiveisusedtorefertothelengthofadataarray ora string. Not available inMASM.
- LOCAL:Thelabel,variables,constantsorproceduresdeclaredLOCALinamoduleare tobeused only by that particularmodule.
- NAME: Logical Name of a Module: The NAME directive is used to assign a name to anassembly language programmodule.
- OFFSET: Offset of a Label: When the assembler comes across the OFFSET operator along with a label,itfirstcomputesthe16-bitdisplacementoftheparticularlabel,andreplacesthestring 'OFFSET LABEL' by the computed displacement.
- ORG:Origin:TheORGdirectivedirectstheassemblertostartthememoryallotmentforthe particular segment, block or code from the declared address in the ORGstatement.
- PROC:Procedure:ThePROCdirectivemarksthestartofanamedprocedureinthestatement. Also the types FAR and NEAR specifies the type of theprocedure.
- PTR: Pointer: The POINTER operator is used to declare the type of a label, variable or memory operand.TheoperatorPTRisprefixedbyeitherBYTE(8-bitquantity)orWORD(16-bitquantity).
- SEGMENT:LogicalSegment:TheSEGMENTdirectivemarksthestartingofalogicalsegment.The started segment is also assigned a name, i.e. label, by thisstatement.
- SHORT:TheSHORToperatorindicatestheassemblerthatonlyonebyteisrequiredtocodethe displacement for a jump. This method of specifying jump address savesmemory.

- TYPE:TheTYPEoperatordirectstheassemblerto decidethedatatypeofthespecifiedlabeland replaces the TYPE label by the decided datatype.
- GLOBAL:Thelabels,variables,constantsorproceduresdeclaredGLOBAL maybeusedbyother modules of theprogram.
- '+ & -'Operators: These operators represent arithmetic addition and subtraction respectively. And are typically used to add or subtract displacements (8 or 16 bit) to base or index registers or stack or basepointers.
- FAR PTR: This directive indicates the assembler that the label following FARPTR is not available within the same segment and the address of the bit is of 32 bits i.e. 2 bytes offset followed by 2 bytes.
- NEAR PTR: This directive indicates that the label following NEAR PTR is in the same segment and need only 16 bit i.e. 2 byte offset to address it. A NEAR PTR label is considered as default if a label is not preceded by NEAR PTR or FARPTR.

Part – A Module-1

8086Architecture

ARCHITECTURE OF 8086



The 8086 CPU is organized as two separate processors, called the Bus Interface Unit (BIU) and the Execution Unit (EU). The BIU handles all transactions of data and addresses on the buses for EU. The BIU performs all bus operations such as instruction fetching, reading and writing operands for memory and calculating the addresses of the memory operands. The instruction bytes are transferred to the instruction queue. EU executes instructions from the instruction system byte queue. Both units operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as Pipelining. This results in efficient use of the system bus and system performance. BIU contains Instruction queue, Segment registers, Instruction pointer, Address adder. EU contains Control circuitry, Instruction decoder, ALU, Pointer and Index register, Flag register. Features of 8086 Microprocessor:

- Intel 8086 was launched in1978.
- It was the first 16-bitmicroprocessor.
- This microprocessor had major improvement over the execution speed of8085.
- It is available as 40-pin Dual-Inline-Package(DIP).

Part – A Module-1

DOS / BIOSINTERRUPTS

1. **DOS Interrupt 21 – Service 01**(Read Character From Standard Input - Keyboard, WithEcho)

Registers used:

AH = 1

AL = character read from keyboard

Ex:

```
MOV AH, 1
INT 21H
```

2. **DOS Interrupt 21 - Service 02** (Write Character To Standard Output -Monitor)

Registers used:

AH = 2

DL = the character to be displayed. Ex:

```
MOV
AH,     2
MOV
DL,'A'
INT 21H
```

3. **DOSInterrupt21 –Service09**(WriteStringToStandardOutput–Monitor&terminatedbya$ to themonitor)

Registers used:

AH = 09

DX = the offset address of the data to be displayed.

Ex:

```
MOV AH, 09
MOV DX, OFFSET MESS1
INT 21H
```

4. **DOS Interrupt 21 - Service 0Ah** (BufferedInput.)

Registers used:

AH = 0AH

DX = the offset address of the input string (Entered by Keyboard). Ex:

```
MOV     AH,
0AH     LEA
DX,     BUFF
INT 21H
```

5. **DOS Interrupt 21 –Service 4Ch** (Terminatesa process,byreturningcontrol toaparent process or toDOS.)

Registers used:

AH = 4CH

Ex:

MOV AH,

4CH INT

21H

Part – A Module-2

Introduction toMSP430

MSP430:

The **MSP430** is a mixed-signal microcontroller family from Texas Instruments. Built around a 16- bit CPU, the MSP430 is designed for low cost and, specifically, low power consumption embedded applications.

Applications

The MSP430 can be used for low powered embedded devices. The current drawn in idle mode can be less than 1 μA. The top CPU speed is 25 MHz. It can be throttled back for lower power consumption. The MSP430 also uses six different low-power modes, which can disable unneeded clocks and CPU. Additionally, the MSP430 is capable of wake-up times below 1 microsecond, allowing the microcontroller to stay in sleep mode longer, minimizing its average current consumption. The device comes in a variety of configurations featuring the usual peripherals: internal oscillator, timer including PWM, watchdog, USART, SPI, I²C, 10/12/14/16/24-bit ADCs, and brownout reset circuitry. Some less usual peripheral options include comparators (that can be used with the timers to do simple ADC), on- chip op-amps for signal conditioning, 12-bit DAC, LCD driver, hardware multiplier, USB, and DMA for ADC results. Apart from some older EPROM (MSP430E3xx) and high volume mask ROM (MSP430Cxxx) versions, all of the devices are in-system programmable via JTAG (full four-wire or Spy-Bi-Wire) or a built in bootstrap loader (BSL) using UART such as RS232, or USB on devices with USBsupport.

There are, however, limitations that preclude its use in more complex embedded systems. The MSP430 does not have an external memory bus, so it is limited to on-chip memory (up to 512 KB flash memory and 66 KB RAM) which may be too small for applications that require large buffers or data tables. Also, although it has a DMA controller, it is very difficult to use it to move data off the chip due to a lack of a DMA output strobe.

MSP430 Nomenclature

An **MSP430** part number such as "**MSP430F2618ATZQWT-EP**" consists of the following pieces:

- **MSP430**: Standardprefix.
- **F**: Indicates a memory type or specialized application. "**F**" indicating flash memory is by far the most popular. Other options for memory type include "**C**" for masked ROM, "**FR**" for FRAM, "**G**" for Flash Value Line, and "**L**" as in the MSP430L09x series, which indicates a RAM-only part; it must remain continuously powered to retain its programming. A second letter (except for "FR") indicates a specialized application for the part. For example, "**G**" is an optional specialization letter indicating hardware support for a specialized use. "**E**" indicates special electricity meter functions, "**G**" devices are designed for medical instrumentation, and "**W**" devices include a special "scan interface" designed for flow meters. An exception is the MSP430FG2xx devices, which are considered a separate generation.

- **2**: The generation of device. There can be significant changes to core peripherals (clock generators, UARTs, etc.) in different generations. These are not in chronological order, but rather higher values roughly indicate greater size, complexity and cost. For example, generations **3** and **4** include LCD controllers which the others donot.
- **6**: The model within the generation. This indicates the mixture of on-board peripheral devices and number of pins.
- **18**: One or two digits indicating the amount of memory on the device. The numbering is (mostly) consistent throughout the MSP430 series. Not all suffixes are valid with all models; most models are available in 3–6 memory sizes, chosen to match the other capabilities of the device. Larger numbers indicate increasing amounts of memory, but sometimes one type of memory (RAM or ROM) is sacrificed to fit more of theother.

MSP430 Memory configurations

Suffix	**RAM**	**ROM**	**Suffix**	**RAM**	**ROM**
0	128	1 K	10	5 K	32 K
1	128	2 K	11	10 K	48 K
2	256	4 K	12	5 K	55/56 K
3	256	8 K	13		
4	512	12 K	14		
5	512	16 K	15		
6	1 K	24 K	16	4 K	92 K
7	1 K	32 K	17	8 K	92 K
8	2 K	48K	18	8 K	116 K
9	2 K	60K	19	4 K	120 K

- An optional suffix digit indicating a variant device, adding or deleting some analog peripherals. For example, a "**1**" suffix may indicate the addition of a comparator or deletion of an ADC. If the memory size is "**1**", this suffix can be confused with part of the memory size, but no single model is available in both "1" and "10" (or greater) memorysizes.
- An optional "**A**" suffix indicating an upward-compatible revised version. The MSP430F11x1A has an additional 256 bytes of data flash not present in the plain'F11x1.

Trailing suffix letters indicate options not visible to software:

- **T**: Indicates a temperature range of −40 °C to +105°C.
- **ZQW**: Indicates the package the part is kept in. "**ZQW**" is a TI-specific name for a ball gridarray.
- **T**: Indicates that the parts are shipped in small reel (7-inch)packaging.
- **-EP**: Indicates an additional feature. "**-Q1**" specifies that the part is automotive qualified. "**-EP**" and "**-HT**" indicate extended temperature products. Enhanced products, "**-EP**", have a temperature range from -40 °C to 125 °C, and extreme temperature parts, "**-HT**", have a temperature range from -56 °C to 150°C.

MSP430 generations

There are six general generations of MSP430 processors. In order of development, they were the '3xx generation, the '1xx generation, the '4xx generation, the '2xx generation, the '5xx generation, and the '6xx generation. The digit after the generation identifies the model (generally higher model numbers are larger and more capable), the third digit identifies the amount of memory on board, and the fourth, if present, identifies a minor model variant. The most common variation is a different on-chip analog-to- digital converter.

The 3xx and 1xx generations were limited to a 16-bit address space. In the later generations this was expanded to include '430X' instructions that allow a 20-bit address space. As happened with other processor architectures (e.g. the processor of the PDP-11), extending the addressing range beyond the 16-bit word size introduced some peculiarities and inefficiencies for programs larger than 64 Kbytes.

In the following list, it helps to think of the typical 200 mA Hr capacity of a CR2032 lithium coin cell as 200,000 µAHr, or 22.8 µAyear. Thus, considering only the CPU draw, such a battery could supply a 0.7 µA current draw for 32 years. (In reality, battery self-discharge would reduce thisnumber.)

The significance of the 'RAM retention' vs the 'real-time clock mode' is that in real time clock mode the CPU can go to sleep with a clock running which will wake it up at a specific future time. In RAM retention mode, some external signal is required to wake it, e.g. I/O pin signal or SPI slave receive interrupt.

MSP430G2xx series

The MSP430G2xx Value Series features flash-based Ultra-Low Power MCUs up to 16 MIPS with 1.8–3.6 V operation. Includes the Very-Low power Oscillator (VLO), internal pull-up/pull-down resistors, and low- pin count options, at lower prices than the MSP430F2xx series.

- Ultra-Low Power, as low as (@2.2V):
  - 0.1 µARAM retention
  - 0.4 µA Standby mode (VLO)
  - 0.7 µA real-time clockmode
  - 220 µA / MIPSactive
  - Ultra-Fast Wake-Up From Standby Mode in <1µs
- Device parameters
  - Flash options: 0.5–56KB
  - RAM options: 128 B–4KB
  - GPIO options: 10, 16, 24, 32pins
  - ADC options: Slope, 10-bitSAR
  - Other integrated peripherals: Capacitive Touch I/O, up to 3 16-bit timers, watchdog timer, brown-out reset, USI module (I^2C, SPI), USCI module, Comparator_A+, Tempsensor

Experiment No

1

Programs using Arithmetic and logical operations

Regd. No:_____ **Date:**_____

OBJECTIVES

1) To Perform addition of n-numbers using ALP
2) To evaluate an expression W=X+Y-Z
3) To perform conditional addition using logical operations

APPARATUS

Computer System Installed with EMULATOR 8086 Software.

PRE LAB QUESTIONS

1) Differentiate ADD and ADC instructions?

2) List different arithmetic instructions in8086?

3) List various logical instructions in8086?

4) What is an assembler directive and mention some assembler directives in8086?

5) Explain about CMP instruction?

PROGRAM

A) ADDITION OF N-BITNUMBERS

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV BX, OFFSET ARRAY	
			MOV AL, 0	
			MOV CL, LEN	
			L1: ADD AL,[BX]	
			INC BX	
			LOOP L1	
			MOV ANS, AL	
			INT 3	
			ARRAY DB 34H, 12H, 16H,15H, 29H,11H, 09H, 19H	
			LEN EQU 08	
			ANS DB ?	
			END	

OBSERVATIONS

INPUT: Array= 34H, 12H, 16H, 15H, 29H, 11H, 09H,19H

OUTPUT: ANS (ADDRESS)=

THEORITICAL CALCULATIONS

B) EVALUATION OF ANEXPRESSION

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV AL,0	
			ADD AL, X	
			ADD AL, Y	
			SUB AL, Z	
			MOV W,AL	
			INT 3	
			X DB25H	
			Y DB33H	
			Z DB11H	
			W DB ?	
			END	

OBSERVATIONS

INPUT:X=25H Y=33H Z=11H

OUTPUT: W (ADDRESS) =

THEORITICAL CALCULATIONS

C) PROGRAM TO ADD N NUMBERS WITH A CONDITION OF "BIT NO 3 AND 6 MUST BE1".

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV BX, OFFSET ARRAY	
			MOV AL,0	
			MOV CL, LEN	
			L1: MOVAH,[BX]	
			AND AH, 01001000B	
			CMP AH,01001000B	
			JNZ NEXT	
			ADD AL,[BX]	
			NEXT: INCBX	
			LOOP L1	
			MOV ANS, AL	
			INT 3	
			ARRAY DB 34H, 12H, 59H,15H, 49H,11H, 09H,19H	
			LENEQU 08	
			ANS DB ?	
			END	

OBSERVATIONS

INPUT: ARRAY= 34H, 12H, 59H,15H, 49H,11H, 09H,19H

OUTPUT: ANS (ADDRESS) =

THEORITICAL CALCULATIONS

POST LAB QUESTIONS

1) What do you understand the importance of OFFSER assembler directive in aboveprogram?

2) What happens if LOOP instruction is executed in aboveprogram?

3) Which addressing mode is used for addition and subtraction in aboveprogram?

4) Identify the branch instructions in aboveprograms?

_____

_____

_____

5) What are the major two flags that effects arithmetic and logicaloperations?

_____

_____

_____

OUT COMES

Upon completion of the experiment, the student will be able to

1. Analyze & understand the different Arithmetic and logicalOperations.
2. Verified theoretical values with practicalvalues

GRADING

DATE OF SUBMISSION						
MARKSAWARDED (Max. Marks -10M	**Pre-lab Questions (2M)**	**Observations (2M)**	**Calculations (2M)**	**Post-Lab Questions (2M)**	**Viva (2M)**	**Total (10M)**
Remarks						
Signature of the Evaluator with Date						

NOTES

Experiment No

2

Move Block, Reverse string, StringComparison

Regd.No:_____ **Date:**_____

OBJECTIVES

1) To move a block of data from source to destination usingALP
2) To reverse a given string usingALP
3) To compare two strings usingALP

APPARATUS

Computer System Installed with EMULATOR 8086 Software.

PRE LAB QUESTIONS

1) What is the importance of SI and DI registers instrings?

2) List different string instructions in8086?

3) What is the importance of REP in stringinstructions?

4) Explain about STD andCLD?

5) List various DOSinterrupts?

PROGRAM

A) MOVE BLOCK

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV SI, OFFSET SOURCE	
			MOV DI, OFFSET DEST	
			MOV CX, LEN	
			REP MOVSB	
			INT 3	
			SOURCE DB “RCEW”	
			LEN EQU 04	
			DEST DB 04 DUP (?)	
			END	

OBSERVATIONS

INPUT: SOURCE=“RCEW”

OUTPUT: DEST (ADDRESS)=

B) REVERSESTRING

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV DI, OFFSET SOURCE	
			MOV AL,'$'	
			REPNE SCASB	
			MOV SI,DI	
			DEC SI	
			DEC SI	
			MOV DI, OFFSET DEST	
			MOV CX, LEN	
			STD	
			L1: MOVSB	
			ADD DI,02	
			LOOP L1	
			INT 3	
			SOURCEDB "RCEW$"	
			LENGTH EQU04	
			DEST DB 04DUP(?)	
			END	

OBSERVATIONS

INPUT: SOURCE="RCEW"

OUTPUT: DEST (ADDRESS)=

C) STRING COMPARISON

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV SI, OFFSET S1	
			MOV DI, OFFSET S2	
			MOV CX, LEN	
			CLD	
			REPZ CMPSB	
			JZ MSG1	
			MOV DX, OFFSET S4	
			JMP MSG2	
			MSG1: MOV DX, OFFSET S3	
			MSG2: MOV AH,09	
			INT 21H	
			INT 3	
			S1 DB "THISIS FIRST STRING$"	
			S2 DB "THISIS SECOND STRING$"	
			LENGTH EQU 25	
			S3 DB "STRINGS E EQUAL$"	
			S4 DB "STRINGS ARE NOT EQUAL$"	
			END	

OBSERVATIONS

INPUT: S1= "THIS IS FIRST STRING" S2="THIS IS SECONDSTRING"

OUTPUT:

POST LAB QUESTIONS

1) What happens if we execute STD before MOVSBinstruction?

2) What is the need of ADD SI, 02 IN reverse stringprogram?

3) Which addressing mode is used for comparison of a string in aboveprogram?

4) What is the importance of EQU assembler directive in aboveprogram?

5) What are the two flags that effects stringoperations?

OUT COMES

Upon completion of the experiment, the student will be able to

1. Analyze the stringinstructions
2. Analyze the DOSinterrupts

GRADING

DATE OF SUBMISSION					
MARKSAWARDED (Max. Marks -10M	**Pre-lab Questions (2M)**	**Observations (3M)**	**Post-Lab Questions (2M)**	**Viva (3M)**	**Total (10M)**
Remarks					
Signature of the Evaluator with Date					

NOTES

Experiment No

3

Programs for code conversion

Regd.No:_____ **Date:**_____

OBJECTIVES

1) To convert BCD to seven segment code conversion usingALP
2) To convert Binary to gray code conversion usingALP
3) To convert BCD to ASCII code conversion usingALP

APPARATUS

Computer System Installed with EMULATOR 8086 Software.

PRE LAB QUESTIONS

1) Give an example for BCD to seven segment codeconversion?

2) GiveanexampleforBinarytograycodeconversion?

3) Give an example for BCD to ASCII codeconversion?

4) Explain about ROR and XLAT instruction in8086?

5) Differentiate SAL andSHL?

PROGRAM

A) BCD TO SEVEN SEGMENT CODECONVERSION

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV BX, OFFSET S	
			MOV AL,X	
			XLAT	
			MOV ANS, AL	
			INT 3	
			S DB 3FH,06H,5BH,4FH, 66H,6DH,7DH,07H,7FH,6FH	
			X DB 05	
			ANS DB ?	
			END	

OBSERVATIONS

INPUT: X= 05H

OUTPUT: ANS (ADDRESS)=

THEORITICAL CALCULATIONS

B) BINARY TO GRAY CODECONVERSION

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV AL,X	
			MOV BL,AL	
			RCR AL,1	
			XOR AL,BL	
			MOV ANS,AL	
			INT 3	
			X DB 05	
			ANS DB ?	
			END	

OBSERVATIONS

INPUT: X=05H

OUTPUT: ANS (ADDRESS)=

THEORITICAL CALCULATIONS

C) BCD TO ASCII CODECONVERSION

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV AL,X	
			AND AL,0FH	
			ADD AL, 30H	
			MOV AH,X	
			AND AH,F0H	
			ROR AH,04	
			ADD AH, 30H	
			MOV ANS,AX	
			INT 3	
			X DB 45H	
			ANS DW ?	
			END	

OBSERVATIONS

INPUT: X=45H

OUTPUT: ANS (ADDRESS) =

THEORITICAL CALCULATIONS

POST LAB QUESTIONS

1) WhathappensifXLATinstructionisexecutedinBCDtosevensegmentcodeconversion program?

2) WhathappensifthisinstructionANDAL,80isexecutedinabovebinarytograycodeconversion program?

3) WhathappensifthisinstructionRORAH,04isexecutedinaboveBCDtoASCIIcodeconversion program?

4) What is the value present in AH after executing the SAL AH,1 andAH=05h?

_____

_____

_____

5) What is the addressing mode of OR AL, AHinstruction?

_____

_____

_____

OUT COMES

Upon completion of the experiment, the student will be able to

1. Analyze & understand the different codeconversions.
2. Verified theoretical values with practicalvalues

GRADING

DATE OF SUBMISSION						
MARKSAWARDED (Max. Marks -10M	**Pre-lab Questions (2M)**	**Observations (2M)**	**Calculations (2M)**	**Post-Lab Questions (2M)**	**Viva (2M)**	**Total (10M)**
Remarks						
Signature of the Evaluator with Date						

NOTES

Experiment No

4

Programs for Multiplication and Division

Regd.No:_____ **Date:**_____

OBJECTIVES

1) To perform multiplication operation usingALP
2) To perform division operation usingALP

APPARATUS

Computer System Installed with EMULATOR 8086 Software.

PRE LAB QUESTIONS

1) In DIV instruction the numerator and denominator values are stored in whichregister?

2) Explain about BYTE PTR in8086?

3) Give an example for 8-bit multiplicationoperation?

4) Which flag is affected when we execute IMULinstruction?

5) Differentiate MUL andIMUL?

_____

_____

_____

PROGRAM

A) PROGRAM TO FIND ∑ XY

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV SI, OFFSET X	
			MOV BX, OFFSET Y	
			MOV CX,09	
			MOV DX,0	
			L1: MOVAL,[SI]	
			MUL BYTE PTR [BX]	
			ADD DX,AX	
			INC SI	
			INC BX	
			LOOP L1	
			MOV AX,DX	
			INT 3	
			X DB 1,2,4,5,6,7,8,9	
			Y DB 1,1,1,1,1,1,1,1	
			ANS DB ?	
			END	

OBSERVATIONS

INPUT: X = 1,2,4,5,6,7,8,9
Y = 1, 1,1,1, 1,1,1,1,1

OUTPUT: ANS (ADDRESS)=

THEORITICAL CALCULATIONS

B) PROGRAM TO FIND AVERAGE OF ANARRAY

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV BX, OFFSET X	
			MOV CX,09	
			MOV AX,0	
			L1: ADDAL,[BX]	
			INC BX	
			LOOP L1	
			MOV BL,09	
			DIV BL	
			MOV ANS, AL	
			INT 3	
			X DB 1, 2,3,4,5,6,7,8,9	
			ANS DB ?	
			END	

OBSERVATIONS

INPUT: X = 1,2, 3, 4, 5,6,7,8,9

OUTPUT: ANS (ADDRESS)=

THEORITICAL CALCULATIONS

POST LAB QUESTIONS

1) What happens if we execute LOOP L1 instruction in aboveprogram?

2) What is the status of the flag after unsignedaddition?

3) Which addressing mode is used for DIV instruction in aboveprogram?

4) Is immediate addressing mode is allowed or not in MUL instruction inn aboveprogram?

5) What is the status of flags after execution of DIV BL instruction in aboveprogram?

OUT COMES

Upon completion of the experiment, the student will be able to

1. Understand and analyze the multiplication and divisionoperations
2. Verify the theoretical and practicalvalues

GRADING

DATE OF SUBMISSION						
MARKSAWARDED (Max. Marks -10M	**Pre-lab Questions (2M)**	**Observations (2M)**	**Calculations (2M)**	**Post-Lab Questions (2M)**	**Viva (2M)**	**Total (10M)**
Remarks						
Signature of the Evaluator with Date						

NOTES

Experiment No

5

Programs for Sorting and MultiByteArithmetic

Regd.No:_____ **Date:**_____

OBJECTIVES

1) To perform ascending order/ descending order operations usingALP
2) To perform MultiByte addition/subtraction operation usingALP

APPARATUS

Computer System Installed with EMULATOR 8086 Software.

PRE LAB QUESTIONS

1) Explain about XCHG instruction in8086?

2) Give an example for ADC instruction in8086?

3) Give an example for INC and DEC instructions in8086?

4) What are the address storage registers in8086?

5) Differentiate DB andDW?

_____

_____

_____

PROGRAM

A) SORTING PROGRAM

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV BX, OFFSET X	
			MOV CL, COUNT	
			DEC CL	
			L2: MOVDI,CX	
			L1: MOVAL,[BX]	
			CMP AL,[BX+1]	
			JNC NEXT	
			XCHG AL,[BX+1]	
			MOV [BX],AL	
			NEXT: INC BX	
			LOOP L1	
			MOV BX, OFFSET X	
			MOV CX,DI	
			LOOP L2	
			INT 3	
			X DB 1,9,8,4,5,3,2,7,6	
			COUNTEQU 09	
			END	

OBSERVATIONS

INPUT: X =1, 9, 8,4,5,3,2,7,6

OUTPUT: BX (ADDRESS) =

THEORITICAL CALCULATIONS

B) MULTIBYTE ADDITION

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV BX, OFFSET X	
			MOV SI, OFFSET Y	
			MOV DI, OFFSET ANS	
			MOV CX, COUNT	
			L1: MOVAL,[BX]	
			ADC AL,[SI]	
			MOV [DI],AL	
			INC BX	
			INC SI	
			INC DI	
			LOOP L1	
			INT 3	

			X DB 35H,67H,45H,23H,11H	
			Y DB 11H,22H,65H,44H,37H	
			ANS DB 5 DUP(?)	
			COUNTEQU 5	
			END	

OBSERVATIONS

INPUT: X = 35H,67H,45H,23H,11H
Y = 11H,22H,65H,44H,37H

OUTPUT: ANS (ADDRESS)=

THEORITICAL CALCULATIONS

POST LAB QUESTIONS

1) Whatisthestatusofthe flagafterexecutionofADCAL,[SI]instructioninabovemultibyte additionprogram?

2) What is the result stored in BX if MOV BX, OFFSET X instruction isexecuted?

3) Explain about SI and DI instruction in aboveprograms?

_____

_____

_____

4) Which register is important for executing LOOPinstruction?

_____

_____

_____

OUT COMES

Upon completion of the experiment, the student will be able to

1. Understand and analyze the sorting and Multibyte addition operations usingALP
2. Verify the theoretical and practicalvalues

GRADING

DATE OF SUBMISSION						
MARKSAWARDED (Max. Marks -10M	**Pre-lab Questions (2M)**	**Observations (2M)**	**Calculations (2M)**	**Post-Lab Questions (2M)**	**Viva (2M)**	**Total (10M)**
Remarks						
Signature of the Evaluator with Date						

NOTES

Experiment No

6

Programs using CALL and RET instructions

Regd.No:_____ **Date:**_____

OBJECTIVES

1) To find factorial of a number usingALP
2) To find whether the given number is prime or not usingALP
3) To perform Fibonacci series UsingALP

APPARATUS

Computer System Installed with EMULATOR 8086 Software.

PRE LAB QUESTIONS

1) Differentiate procedure and macro arithmetic instructions in8086?

2) Explain about RET instruction in8086?

3) What is addressing mode used for MUL instruction?

4) Explain about JE and JNE instructions in8086?

_____

_____

_____

5) What is the difference between SUB and CMP instructions in8086?

_____

_____

_____

PROGRAM

A) FACTORIAL OF A NUMBER USINGPROCEDURE

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV CL, X	
			MOV CH,0	
			CALL FACT	
			MOV ANS,AL	
			INT 3	
			FACT PROC NEAR	
			MOV AL,01	
			L1: MULCL	
			LOOP L1	
			RET	
			ENDP	
			END	
			X DB 5	
			ANS DB ?	

OBSERVATIONS

INPUT: X =5

OUTPUT: ANS (ADDRESS)=

THEORITICAL CALCULATIONS

B) PROGRAM TO FIND PRIME OR NOT

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV AL,NUM	
			MOV BL,02H	
			MOV DX,0000H	
			MOV AH,00H	
			L1: DIV BL	
			CMP AH,00H	
			JNE NEXT	
			INC BH	
			NEXT: CMP BH,02H	
			JE FALSE	
			INC BL	
			MOV DX,0000H	
			MOV AH,00H	
			MOV AL,NUM	
			CMP BL,NUM	

			JNE L1	
			TRUE: LEA DX,MSG1	
			MOV AH,09H	
			INT 21H	
			JMP EXIT	
			FALSE: LEA DX,MSG2	
			MOV AH,09H	
			INT 21H	
			JMP EXIT	
			EXIT: INT 3	
			MSG1 DB "GIVEN NO IS PRIME"	
			MSG2 DB "GIVEN NO IS NOT PRIME"	
			NUM DB 71H	
			END	

OBSERVATIONS

INPUT: NUM =71H

OUTPUT:

THEORITICAL CALCULATIONS

C) FIBONACII SERIESGENERATION

S.NO	ADDRESS	OPCODE	MNEMONIC/ OPERANDS	COMMENTS
			MOV BX, OFFSET SERIES	
			CALL FIBONACI	
			INT 3	
			FIBONACCI PROC NEAR	
			MOV AH, 00	
			MOV AL, 01	
			MOV [BX], AH	
			INC BX	
			MOV [BX], AL	
			MOV CX, 08	
			L1: ADD AL, AH	
			MOV AH, [BX]	
			INC BX	
			MOV [BX], AL	
			LOOP L1	
			RET	
			ENDP	
			END	
			SERIES DB 10 DUP(?)	

OBSERVATIONS

INPUT: SERIES =10

THEORITICAL CALCULATIONS

OUTPUT:

POST LAB QUESTIONS

1) How many times the loop executes if CX=08h instruction is executed in above program?

2) What is the importance of CALL instruction in above program?

3) Explain about FIBONACCI PROC NEAR instructions in above program?

4) Explain about LEA instruction in above program?

5) Differentiate END and ENDP in above program?

OUTCOMES

Upon completion of the experiment, the student will be able to

1. Understand and analyze the CALL and RET instructions
2. Verify the theoretical and practicalvalues

GRADING

DATE OF SUBMISSION						
MARKSAWARDED (Max. Marks -10M	**Pre-lab Questions (2M)**	**Observations (2M)**	**Calculations (2M)**	**Post-Lab Questions (2M)**	**Viva (2M)**	**Total (10M)**
Remarks						
Signature of the Evaluator with Date						

NOTES

PART-B

EMBEDDED C EXPERIMENTSUSING MSP430 MICROCONTROLLER

Department of Electronics and Communication
Engineering SVR Engineering College: Nandyal
Approved by AICTE, New Delhi & Affiliated to JNTUA,
Anantapuramu (An ISO 9001:2007 Certified Institution)
Near Ayyalur(v), Allagadda Rd, Nandyal, Andhra Pradesh 518503

Experiment No.

7 Interfacing and Programming GPIO Ports

Regd.No:_____ Date:_____

Objectives:

(A)To blink the RED LED using C language.

(B)To blink the GREEN LED using C language.

(C)To blink the RED & GREEN LED using Clanguage.

(D)To control the on-board GREEN LED by taking the input fromswitch

Apparatus:

1. Code ComposerStudio
2. MSP430G2553 Launch Pad plugged into your PC viaUSB

Pre Lab Questions:

1. Define EmbeddedSystem.

_____

_____

_____

2. What are the features ofMSP430?

_____

_____

_____

3. Define WatchdogTimer.

_____

_____

_____

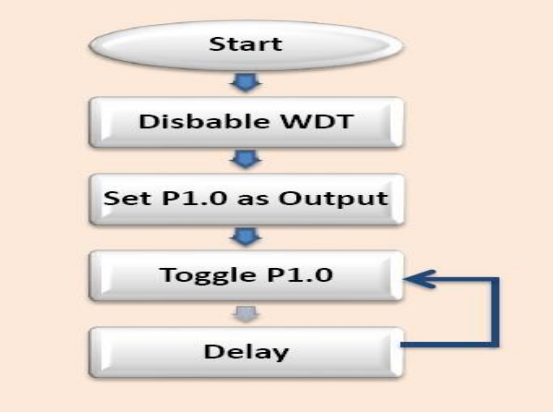
Procedure:

1. Open code composerstudio
2. Select new CCSproject
3. Write embedded Cprogram
4. Build theprogram
5. Load and run thecode
6. Observe the outputs on MSP430 launchpad

(A) Program:

```
#include<msp430.h>
intmain(void) {
WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
P1DIR |= 0x01; // Set P1.0 to output direction
while(1) {
volatile unsigned long i; // Volatile to prevent
//optimization
P1OUT ^= 0x01; // Toggle P1.0 using XOR
i = 50000; // SW Delay
doi--;
while(i != 0);
}
}
```

Flow Chart:



(B) Program:

```
#include<msp430.h>
intmain(void) {
WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
P1DIR |= 0x40; // Set P6.0 to output direction
while(1) {
volatile unsigned long i; // Volatile to prevent
//optimization
P1OUT ^= 0x40; // Toggle P6.0 using XOR
i = 50000; // SW Delay
doi--;
while(i != 0);
}
}
```

Flow Chart:

(C) Program:

```
#include<msp430.h>
intmain(void)
 {
WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
P1DIR |= 0x41; // Set P6.0 to output direction
while(1)
 {

volatile unsigned long i; // Volatile to prevent
//optimization
P1OUT ^= 0x41; // Toggle P6.0 using XOR
i = 50000; // SW Delay
doi--;
while(i != 0);
}
}
```

(D)Program:

```
#include<msp430.h>
intmain(void)
{
WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
```

```
P1DIR |= 0x40; // Set P1.6 to output direction
P1REN |= 0x08;
P1OUT  |=0X08;
while(1)
{
if((P1IN & BIT3)) { // If button is open(P1.3 HIGH)
P1OUT = P1OUT & ~BIT6; // ... else turn it off.
} // or P1OUT |= BIT0;
Else
{
P1OUT = P1OUT | BIT6; // ... turn on LED
// or P1OUT &= ~BIT0
}
}
}
```

Post lab Questions:

1. What is the need of Pull up and Pull down Resistors for GPIOlines?

_____

_____

_____

2. What are I/O operations inMSP430?

.

_____

_____

_____

3. List the registers available inMSP430?

_____

_____

_____

Outcomes:

Upon completion of the experiment, the student will be able to

1. Develop an algorithm, the flow diagram, source code and perform the compilation.
2. Generate the required binary file which can be dumped into the controller and obtain the respective output control on the connectedperipheral.
3. Verify the logic with the necessaryhardware.

Grading:

Date of Submission					
Marks Awarded (Max.Marks-10M)	Pre-Lab Questions (2M)	Observations (3M)	Post-Lab Questions (2M)	Viva (3M)	Total (10M)
Remarks					
Signature of the Evaluator with Date					

NOTES

Experiment No.

8 Usage of Low PowerModes

Regd.No:_____ Date:_____

Objectives:

To measure active mode and standby mode current using MSPEXP430FR5969 as hardware platform.

Apparatus:

1. Code Composer Studio
2. MSPEXP430FR5969 plugged into your PC viaUSB
3. DigitalMultimeter.

Pre Lab Questions:

1. What is ActiveMode?

_____

_____

_____

2. List the Low PowerModes?

_____

_____

_____

3. What is LPM2 &LPM3?

_____

_____

_____

Procedure:

Measurement of Active Current:

1. Copy the code **main_active.c**in the CCS newproject.
2. Build, load, and run the code. The green LED will blink once every three or fourseconds.
3. When done, halt the code and click the Terminate button to return to the “CCSEdit”.
4. Remove all five jumpers on headerJ3.
5. The red lead of the multimeter should connect to the top (emulation side) Vcc pin on header J3 and the black lead of the multimeter should connect to the bottom (target side) Vcc pin on headerJ3.
6. Press the Reset button on the Launchpadboard.
7. Measure the current drawn by theMSP430.

Measurement of Standby Current:

1. Copy the code **main_standby.c**in the CCS newproject.
2. Build,load,andrunthecode.ThegreenLEDwillblinkonceeverythreeorfourseconds. 3.When done, halt the code and click the Terminate button to return to the “CCSEdit”.
4. Remove all five jumpers on headerJ3.
5. The red lead of the multimeter should connect to the top (emulation side) Vcc pin on header J3 and the black lead of the multimeter should connect to the bottom (target side) Vcc pin on headerJ3.
6. Press the Reset button on the Launchpadboard.
7. Measure the current drawn by theMSP430.

Program:

Main_active.c#inclu
de<msp430.h>**intma**
in(**void**){
WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
P1DIR |= 0x01; // Set P1.0 to output direction
while(1) {
volatile unsigned long i; // Volatile to prevent
//optimization
P1OUT ^= 0x01; // Toggle P1.0 using XOR
i = 50000; // SW Delay
doi--;
while(i != 0);
}
}

Main_standby.c:

```
#include <msp430g2553.h>
#ifndefTIMER0_A1_VECTOR
#define TIMER0_A1_VECTORTIMERA1_VECTOR
#define TIMER0_A0_VECTORTIMERA0_VECTOR
#endif
volatile long tempRaw;
//volatile unsigned int i;
voidFaultRoutine(void);
voidConfigWDT(void);
voidConfigClocks(void);
voidConfigPins(void);
void ConfigADC10(void);
void ConfigTimerA2(void);
void main(void)
{
ConfigWDT();
ConfigClocks(); ConfigPins();
ConfigADC10();
ConfigTimerA2();
// _BIS_SR(GIE);
while(1)
{
_bis_SR_register(LPM3_bits + GIE); // Enter LPM3 with interrupts
}
} void ConfigWDT(void)
{
WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer
}
voidConfigClocks(void)
{
if(CALBC1_1MHZ ==0xFF || CALDCO_1MHZ == 0xFF)
FaultRoutine(); // If calibration data is erased
// run FaultRoutine()
BCSCTL1 = CALBC1_1MHZ; // Set range
DCOCTL = CALDCO_1MHZ; // Set DCO step + modulation
BCSCTL3 |= LFXT1S_2; // LFXT1 = VLO
IFG1 &= ~OFIFG; // Clear OSCFault flag
BCSCTL2 |= SELM_0 + DIVM_3 + DIVS_3; // MCLK = DCO/8, SMCLK = DCO/8
```

```
} void FaultRoutine(void)
{
P1OUT = BIT0; // P1.0 on (red LED)
while(1); // TRAP
} void ConfigPins(void)
{
P1DIR = ~BIT3; // P1.6 and P1.0 outputs
P1OUT = 0;
P2SEL = ~(BIT6 + BIT7);
P2DIR |= BIT6 +BIT7;
P2OUT = 0;// LEDsoff
} void ConfigADC10(void)
{
ADC10CTL1 = INCH_10 + ADC10DIV_0; // Temp Sensor ADC10CLK
} void ConfigTimerA2(void)
{
CCTL0 = CCIE;
CCR0 = 36000;
TACTL = TASSEL_1 + MC_2;
}
#pragma vector=TIMER0_A0_VECTOR
interruptvoid Timer_A(void)
{
ADC10CTL0 = SREF_1 + ADC10SHT_3 + REFON + ADC10ON;
_delay_cycles(4); // Wait for ADC Ref to settle
ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
P1OUT |= BIT6; // P1.6 on (green LED)
_delay_cycles(100);
ADC10CTL0 &= ~ENC; // Disable ADC conversion
ADC10CTL0 &= ~(REFON + ADC10ON); // Ref and ADC10 off
tempRaw = ADC10MEM; // Read conversion value
P1OUT &= ~BIT6; // green LED off
CCR0 +=36000; // add 1 second to the timer
_bic_SR_register_on_exit(LPM3_bits); // Clr LPM3 bits from SR on exit
}
```

Observations:

The current consumption in both the active and standby modes is measured while running the same application code. The reading for both the cases for MSP430G2553 are tabulated in below Table

Table: Current Consumption for MSP430G2553

Platform	Active Mode	Standby Mode
MSP430G2553		

Post lab Questions:

1. Write the Contents of StatusRegister.

2. What is StandbyMode?

3. Write the contents of Status register when MSP430 is operated inLPM4?

Outcomes

Upon completion of the experiment, the student will be able to

1. Develop an algorithm, the flow diagram, source code and perform thecompilation.
2. Generate the required binary file which can be dumped into the controller and obtain the respective output control on the connectedperipheral.
3. Verify the logic with the necessaryhardware.

Grading:

Date of Submission					
Marks Awarded (Max.Marks-10M)	Pre-Lab Questions (2M)	Observations (3M)	Post-Lab Questions (2M)	Viva (3M)	Total (10M)
Remarks					
Signature of the Evaluator with Date					

NOTES

Experiment No.

9 Interrupt programming throughGPIOs

Regd.No:_____ Date:_____

Objectives:

To configure interrupts through GPIOs.

Apparatus:

1. Code ComposerStudio
2. 2MSP430G2553 Launchpad plugged into your PC viaUSB

Pre Lab Questions:

1. What are the various criteria to choosemicrocontrollers?

_____

_____

_____

2. What is Interrupt andISR?

_____

_____

_____

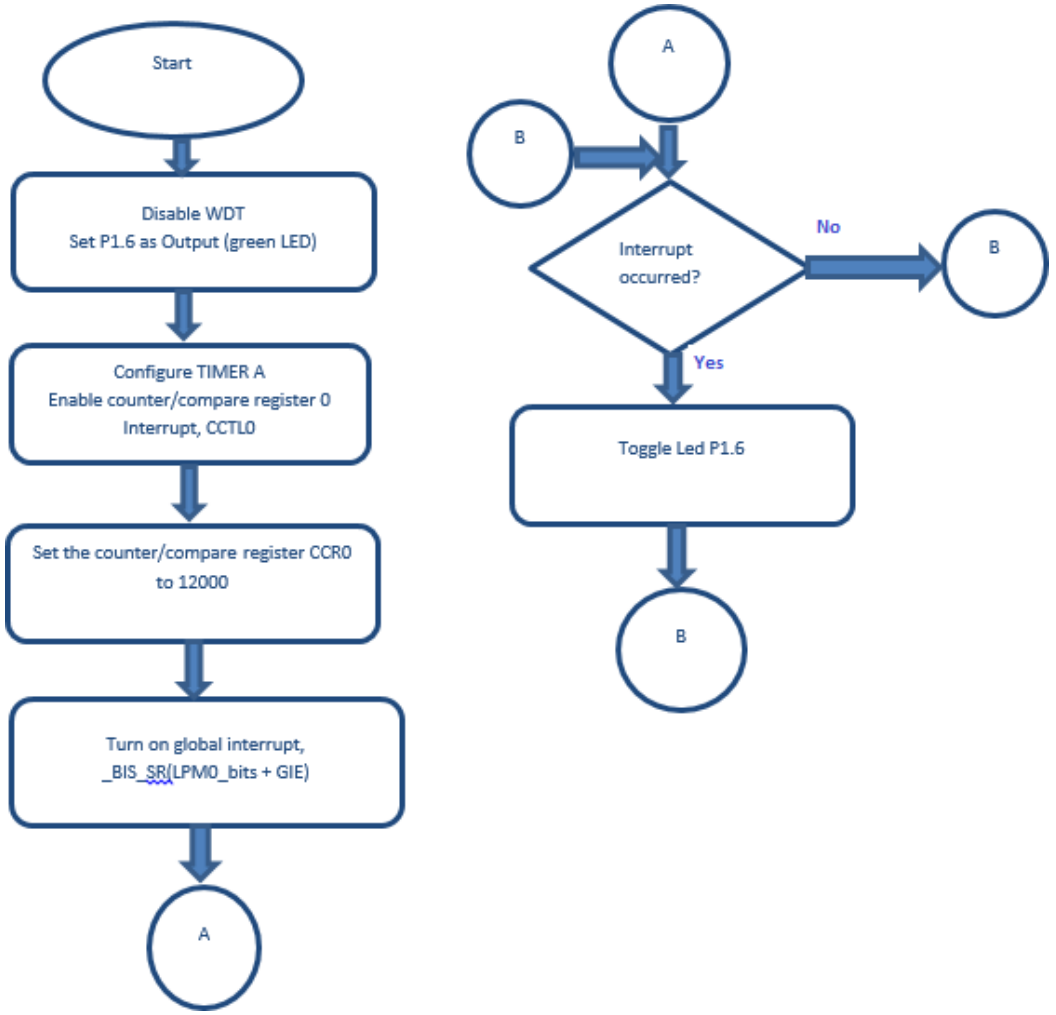
3. What is Interrupt Latency? How can you reduceit?

_____

_____

_____

Flow Chart:



Program:

```
#include <msp430.h>
int main(void)
{
WDTCTL = WDTPW + WDTHOLD; // Stop WDT
P1DIR |= BIT6; // Set P1.6 to output direction P1REN |=
BIT3; // Enable P1.3 internal resistance P1OUT |= BIT3;
// Set P1.3 as pull up resistance P1IES |= BIT3; // P1.3
High/Low Edge
P1IFG &= ~BIT3; // P1.3 IFG Cleared
P1IE |= BIT3; // P1.3 Interrupt Enabled
```

```
_bis_SR_register(LPM4_bits + GIE); // Enter LPM4 w/ interrupt
_no_operation(); // For debugger
}
#pragma vector=PORT1_VECTOR
interrupt void Port_1 (void)
{
P1OUT ^= BIT6; // Toggle P1.6
P1IFG&=~BIT3;//P1.3IFGCleared
}
```

Post lab Questions:

1. What isP1IFG?

_____

_____

_____

2. What isP1IE?

_____

_____

_____

3. What isP1IES?

_____

_____

_____

Outcomes:

Upon completion of the experiment, the student will be able to

1. Develop an algorithm, the flow diagram, source code and perform the compilation.
2. Generate the requiredbinaryfilewhichcanbedumpedintothecontroller and obtain the respective output control on the connectedperipheral.
3. Verify the logic with the necessaryhardware.

Grading:

Date of Submission					
Marks Awarded (Max.Marks-10M)	Pre-Lab Questions (2M)	Observations (3M)	Post-Lab Questions (2M)	Viva (3M)	Total (10M)
Remarks					
Signature of the Evaluator with Date					

NOTES

Experiment No.

10 PWMgenerationusingTimeronMSP430GPIO

Regd.No:_____ Date:_____

Objectives:

To generate PWM using Timer on MSP430 GPIO.

Apparatus:

1. Code ComposerStudio
2. MSP430G2553 Launchpad plugged into your PC viaUSB

Pre Lab Questions:

1. DefinePWM.

_____

_____

_____

2. Define Duty Cycle and how can it becalculated.

_____

_____

_____

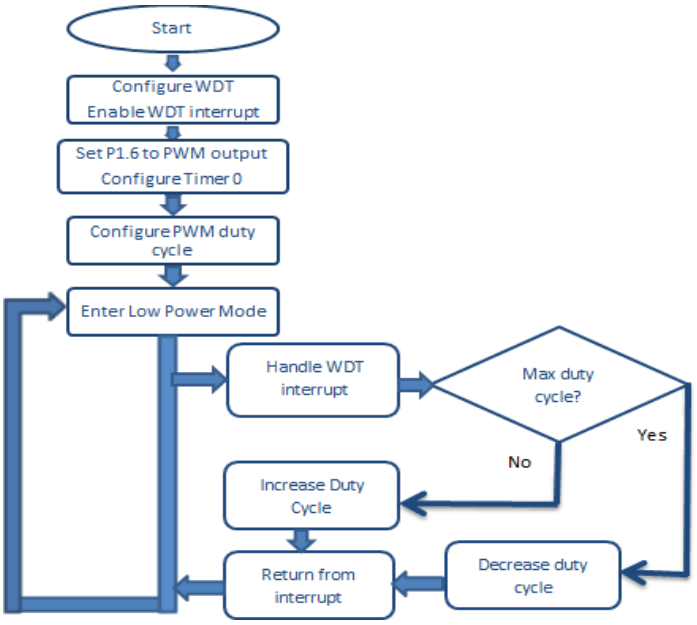
3. What are the concerned registers used for achieving LEDBrightness?

_____

_____

_____

Flow Chart:



Program:

```
#include <msp430g2553.h>
intpwmDirection = 1;
void main(void){
WDTCTL = WDT_MDLY_32; // Watchdog timer 32ms
IE1 |= WDTIE; // enable Watchdog timer interrupts
P1DIR |= BIT6; // Green LED for output
P1SEL |= BIT6; // Green LED Pulse width modulation
TA0CCR0 = 1000; // PWM period
TA0CCR1 = 1; // PWM duty cycle, on 1/1000 initially
TA0CCTL1 = OUTMOD_7; // TA0CCR1 reset/set-high voltage
// below count,low voltage when past
TA0CTL = TASSEL_2 + MC_1; // Timer A control set to SMCLK, 1MHz
        // and count up mode MC_1
_BIS_SR(LPM0_bits+GIE);//EnterLowpowermode0
}
#pragma vector=WDT_VECTOR // Watchdog TimerISR
interrupt void watchdog_timer(void)
```

```
{
TA0CCR1 += pwmDirection*20; // Increase duty cycle, on vs. off time
if( TA0CCR1 > 980 ) // Pulse brighter (increasing TA0CCR1)
TA0CCR1 = 1;
}
```

Post lab Questions:

1. What isTA0CTL?

_____

_____

_____

2. What isTA0CCR0?

_____

_____

_____

3. What isTA0CCTL1?

_____

_____

_____

Outcomes:

Upon completion of the experiment, the student will be able to

1. Develop an algorithm, the flow diagram, source code and perform thecompilation.
2. Generate the required binary file which can be dumped into the controller and obtain the respective output control on the connectedperipheral.
3. Verify the logic with the necessaryhardware.

Grading:

Date of Submission					
Marks Awarded (Max.Marks-10M)	Pre-Lab Questions (2M)	Observations (3M)	Post-Lab Questions (2M)	Viva (3M)	Total (10M)
Remarks					
Signature of the Evaluator with Date					

NOTES

Experiment No.

11 Interfacing Potentiometer withMSP430

Regd.No:_____ Date:_____

Objectives:

To control the on-board, Red LED by the analog input from a potentiometer

Apparatus:

1. Code Composer Studio
2. MSP430G2553 Launchpad plugged into your PC viaUSB
3. 10kΩPotentiometer

Pre Lab Questions:

4. What isADC?

_____

_____

_____

5. What is Resolution inADC?

_____

_____

_____

6. What is Voltage Reference?

_____

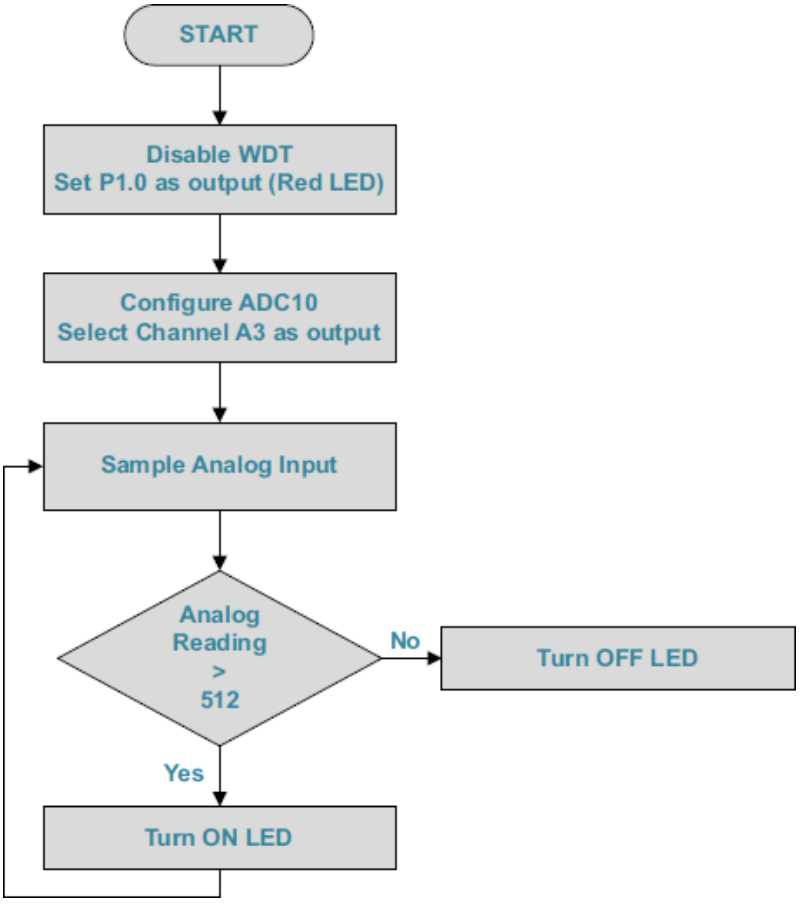
_____

_____

Procedure:

1. Connect the MSP-EXP430G2 LaunchPadto the PC using the USB cablesupplied.
2. Connect the positive lead of the potentiometer to the Vcc pin on the MSP-EXP430G2 Launch-
Pad and the negative lead of the potentiometer to the GND pin.
3. Connect the output lead of the potentiometer to pin P1.3 or Analog ChannelA3.
4. Connect the jumpers on the MSP-EXP430G2 LaunchPad for RXD and TXDhorizontally.
5. Build, program and debug the code into the LaunchPad usingCCS.
6. Vary the potentiometer and observe the on board redLED.

Flow Chart:



Program:

```
int main(void)
{
  WDTCTL = WDTPW+WDTHOLD;          // Stop WDT
  ADC10CTL0 = SREF_0 + ADC10SHT_2 +ADC10ON;
  ADC10CTL1=INCH_3;                 // input A3
  ADC10AE0|=0x08;                   // PA.3 ADC optionselect
  P1DIR|=0x01;                      // Set P1.0 to output direction
  while(1)
  {
    ADC10CTL0 |= ENC+ADC10SC;       // Sampling and conversion start
    if (ADC10MEM<512)               // ADC10MEM = A3 >512?
    P1OUT&=~0x01;                  // Clear P1.0 LED off
    else
      P1OUT|=0x01;                  // Set P1.0 LED on
  }
}
```

Post lab Questions:

1. What isADC10?

_____

_____

_____

2. What is ADC10CTL0?

_____

_____

_____

4. What is ADC10AE &ADC10MEM?

_____

_____

_____

Outcomes:

Upon completion of the experiment, the student will be able to

1. Develop an algorithm, the flow diagram, source code and perform thecompilation.
2. Generate the required binary file which can be dumped into the controller and obtain the respective output control on the connectedperipheral.
3. Verify the logic with the necessaryhardware.

Grading:

Date of Submission					
Marks Awarded (Max.Marks-10M)	Pre-Lab Questions (2M)	Observations (3M)	Post-Lab Questions (2M)	Viva (3M)	Total (10M)
Remarks					
Signature of the Evaluator with Date					

NOTES

Experiment No.

12 PWM based Speed Control of Motorby Potentiometer

Regd.No:_____ Date:_____

Objectives:

To control the speed of a DC Motor using the potentiometer.

Apparatus:

1. Code Composer Studio
2. MSP430G2553 Launchpad plugged into your PC viaUSB
3. ULN2003IC
4. DCMotor

Pre Lab Questions:

1. What isPWM?

_____

_____

_____

2. What isPotentiometer?

_____

_____

_____

3. What isULN2003?

_____

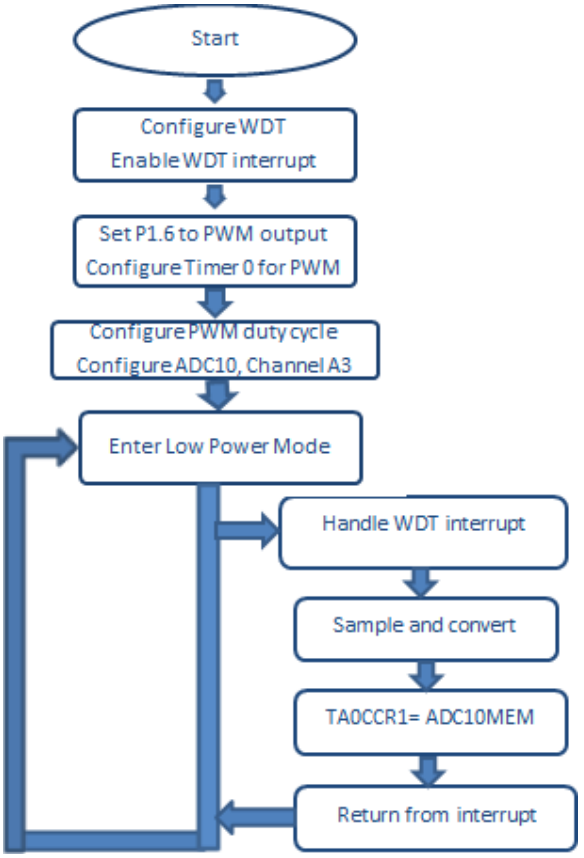
_____

_____

Procedure:

1. Position DIP IC ULN2003AN on a Breadboard.
2. Connect one terminal of the DCmotor to the Common pin (ICPin9)of ULN2003.
3. Connect the junction of the above 2 terminals 5V Power from USB.
4. Connect the other terminal of the DC motor to the Drive Pin (ICPin16)ofULN2003.
5. Connect the GND of the LaunchPad to the Ground Pin (IC Pin 8) ofULN2003.
6. Connect the PWM Output P1.6 to Input Signal (IC Pin 1) ofULN2003.
7. Connect one lead of the Potentiometer to VCC(J1connector,Pin1) on the LaunchPad.
8. Connect other lead of the Potentiometer to the GND Pin of ULN2003 (IC Pin8).
9. Connect the centerleadofthePotentiometer(variableanalogoutput)toP1.3whichis the analog Input 0 of ADC10 module ofMSP43G2553.
10. Build, program and debug the code into the LaunchPad usingCCS.
11. Vary the potentiometer and observe the speed of the DCmotor.
12. Also observe the corresponding digital output on the CCSwindow.

Flow Chart:



Program:

```
#include
<msp430g2553.h>intpwmD
irection = 1;
void main(void){
   WDTCTL=WDT_MDLY_32;              // Watchdog timer32ms
   IE1|=WDTIE;                      // enable Watchdog timer interrupts
   ADC10CTL0 = SREF_0 + ADC10SHT_2 +ADC10ON;
   ADC10CTL1=INCH_3;                // inputA3
   ADC10AE0|=0x08;                  // PA.3 ADC option select
   P1DIR|=BIT6;                     // Green LED foroutput
   P1SEL|=BIT6;                     // Green LED Pulse widthmodulation
   TA0CCR0=1024;                    // PWM period
   TA0CCR1=1;                       // PWM duty cycle,on 1/1000 initially
   TA0CCTL1=OUTMOD_7;               // TA0CCR1 reset/set-highvoltage
                                    // below count,low voltage when past
   TA0CTL = TASSEL_2+MC_1;          // Timer A control set toSMCLK,1MHz
                                    // and count up mode MC_1
   _BIS_SR(LPM0_bits+GIE);          // Enter Low power mode0
}

#pragmavector=WDT_VECTOR                    // Watchdog TimerISR
 interrupt void watchdog_timer(void) {
     ADC10CTL0 |= ENC + ADC10SC;
     TA0CCR1 = ADC10MEM;
}
```

Post lab Questions:

1. What is DCMotor?

_____

_____

_____

2. What is LPM0 &GIE?

_____

_____

_____

3. What isOUTMOD_7?

_____

_____

_____

Outcomes:

Upon completion of the experiment, the student will be able to

1. Develop an algorithm, the flow diagram, source code and perform the compilation.
2. Generate the required binary file which can be dumped into the controller and obtain the respective output control on the connectedperipheral.
3. Verify the logic with the necessaryhardware.

Grading

Date of Submission					
Marks Awarded (Max.Marks-10M)	Pre-Lab Questions (2M)	Observations (3M)	Post-Lab Questions (2M)	Viva (3M)	Total (10M)
Remarks					
Signature of the Evaluator with Date					

NOTES

Experiment No.

13 Using ULP Advisor in CCS onMSP430

Regd.No:_____ Date:_____

Objectives:

To optimize the power efficient application on MSP430 launch pad using ULP Advisor in CCStudio.

Apparatus:

1. Code Composer Studio
2. MSP430G2553 Launchpad plugged into your PC viaUSB

Pre Lab Questions:

4. Mention the capabilities of ULPadvisor.

_____

_____

_____

5. How does combination off unctionsreduce memory requirement in embedded systems?

_____

_____

_____

6. What are the programming languages used in embedded systems?

_____

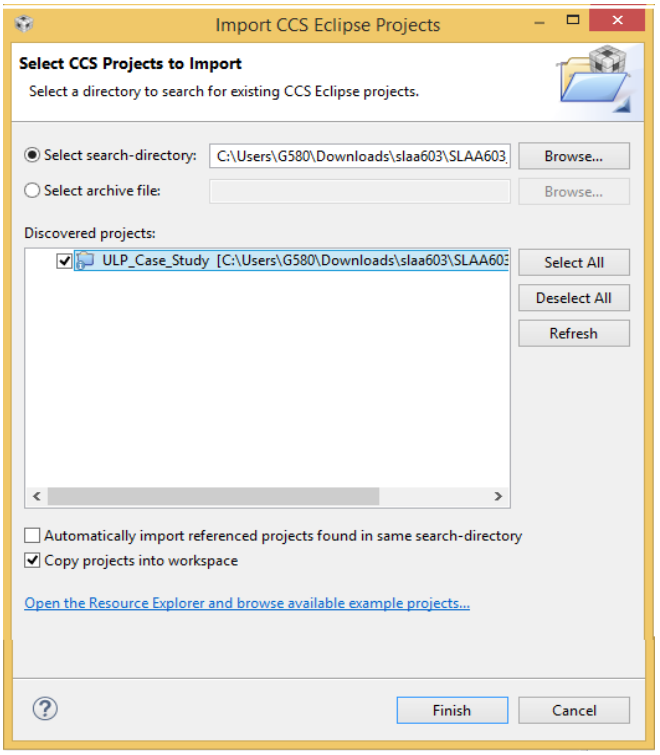
_____

_____

Procedure:

Import the project into your CCS workspace:

1. Click Project --Import Existing CCS Eclipse Project.
2. Browse for the directory where downloaded the file.
3. Checktheboxnexttothe'ULP_Case_Study'projectintheDiscoveredprojects window.
4. Un-check the box next to Copy projects in to workspace.
5. ClickFinish



6. Set as active project by left-clicking on the project name. The active build configuration should be Inefficient. If not, right click on the project name in the Project Explorer, and then click Build Configurations → Set Active →Inefficient.
7. Build the project.
8. The Advice window shows suggestions from the ULP Advisor under the Power (ULP) Advice heading. Click View →Advice
9. Click the link #1532-D, which corresponds to the first ULP violation (for using sprintf()), to open the ULP Advisor wiki page in a second tab titled Advice. All of the information for this particular rule violation is provided.
10. Scroll down to the Remedy section. The first suggestion is to avoid using sprintf() altogether and work with the raw data.

Post lab Questions:

1. How does the ULP advisor software help in designing power optimized code?

_____

_____

_____

2. Which ULP rule violation helps us to detect looping counting violation?

_____

_____

_____

4. Mention how I/O devices are classified for embedded system.

_____

_____

_____

Outcomes:

Upon completion of the experiment, the student will be able to

1. Develop an algorithm, the flow diagram, source code and perform the compilation.
2. Generate the required binary file which can be dumped into the controller and obtain the respective output control on the connected peripheral.
3. Verify the logic with the necessary hardware.

Grading

Date of Submission					
Marks Awarded (Max.Marks-10M)	Pre-Lab Questions (2M)	Observations (3M)	Post-Lab Questions (2M)	Viva (3M)	Total (10M)
Remarks					
Signature of the Evaluator with Date					

NOTES

Experiment No.

14 Low Power modes and EnergyTrace++

Regd.No:_____ Date:_____

Objectives:

(A) To Enable Energy Trace and Energy Trace++ modes in CCS.

(B) To Compute Total Energy, and Estimated lifetime of an AA battery

Apparatus:

1. Code Composer Studio
2. MSP430G2 LaunchPad
3. MSP430FR5969 Launchpad plugged into your PC via the USB cable
5. Jumper Wires

Pre Lab Questions:

1. What are the modes in Energy Capture?

_____

_____

_____

2. What is Energy Trace?

_____

_____

_____

3. What is Energy Trace++?

_____

_____

_____

Procedure:

Hardware Connection

1. Remove the RST, TST, V+, and GND jumpers from J13 on the MSP-EXP430FR5969 Launch-Pad.

2. Remove the RST, TEST, and VCC jumpers from J3 on the MSP-EXP430G2LaunchPad.

3. Use jumper wires to connect the signals on the emulation side of the MSP-EXP430FR5969 board to the corresponding signal on the device side of the MSP-EXP430G2 LaunchPad as in belowTable.

Table: Connection of Signals

MSP-EXP430FR5969 Emulation	MSP-EXP430G2XL Device
RST	RST
TST	TEST
V+	VCC
GND	GND

4. Target power must be supplied through the MSP-EXP430FR5969 LaunchPad that has the Energy Trace technology included in the on-board emulation. Plug-in the micro-USB to the MSPEXP430FR5969.

5. Initialize the debug session for the MSP430G2553 to enable the Energy Trace mode.

Software Execution

1. Build the example code of Experiment 7: PWM Based Speed Control of Motor by Potentiometer that you want to analyze using Energy Trace technology.
2. Enable Energy Trace technology
   a. ClickWindow→Preferences→CodeComposerStudio→AdvancedTools→EnergyTrace Technology
   b. Make sure that the box next to Enable is checked
   c. Select the Energy Trace++mode
   d. Click Apply andOK.
3. Debug the example code, the Energy Trace window will open automatically.
   If the window does not open automatically, click View→Others→MSP430 Energy Trace →Energy Trace Technology
4. Run the code. The Energy Trace Technology window will update the real time data.

(A) To Enable Energy Trace and Energy Trace++ modes inCCS.

Observation:

Complete analysis of Energy Trace Technology for Experiment 7: PWM Based Speed Control of Motor by Potentiometer in terms of Energy, Power, Current, Voltage are given in below Table

Table: Analysis of Energy Trace Technology

Example Code	Energy in mJ	Mean Power in mW	Mean Current in mA	Mean Voltage in V

(B) To Compute Total Energy, and Estimated lifetime of an AAbattery

Observation:

The Energy trace profile for active and stand by code is analyzed using Energy Trace technology. The Energy Trace profile window of the application in active mode provides us the estimated lifetime ofa batteryof_____days. If the same application runs in standby mode, the estimated lifetime of a batteryexceedsto_____days.

Table: Energy measurement and Estimated lifetime of a battery

Mode	Energy in mJ for the 30sec time period	Estimated life time of a battery in days.
Active Mode		
Standby Mode		

Post lab Questions:

1. What is digital signal controller?

2. List the features ofMSP430.

3. Mention the difference between firmware and software.

_____

_____

_____

Outcomes:

Upon completion of the experiment, the student will be able to

1. Develop an algorithm, the flow diagram, source code and perform the compilation.
2. Generate the required binary file which can be dumped into the controller and obtain the respective output control on the connected peripheral.
3. Verify the logic with the necessary hardware.

Grading:

Date of Submission					
Marks Awarded (Max.Marks-10M)	Pre-Lab Questions (2M)	Observations (3M)	Post-Lab Questions (2M)	Viva (3M)	Total (10M)
Remarks					
Signature of the Evaluator with Date					

NOTES

Experiment No.

15

Serial Communication

Regd.No:_____ Date:_____

Objectives:

To use UART of the MSP430G2553 to communicate with the computer.

Apparatus:

1. Code Composer Studio
2. MSP430G2553 Launchpad plugged into your PC viaUSB

Pre Lab Questions:

7. What is Synchronous serial communication?

_____

_____

_____

8. What is Asynchronous serial communication?

_____

_____

_____

9. What is Baud rate?

_____

_____

_____

UART Terminal Setup:

1. In CCS window select **View→Other**and from the options given select**Terminal.**

2. When the terminal window opens, click on the **Settings** icon and set the values as shown in belowFigure

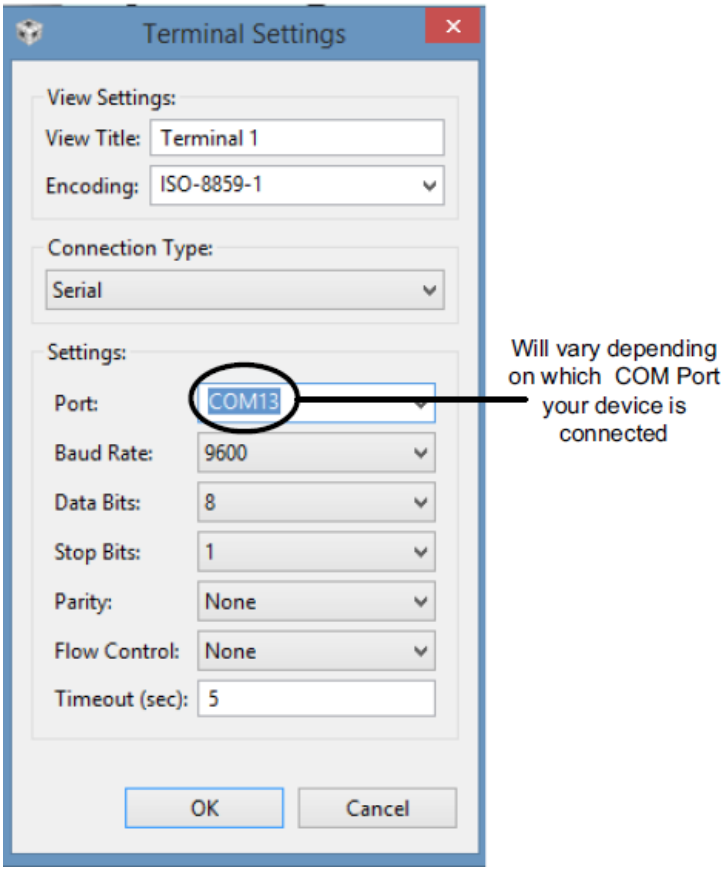


Figure: Terminal Window Settings

This Terminal will allow you to view the serial information received by the computer and will also allow you to type in the information that you wish to transmit to the MSP430G2553 via the UART.

Procedure:

1. Connect the MSP-EXP430G2 LaunchPad to the PC using the USB cable supplied.
2. Build, program and debug the code into the LaunchPad using CCS.

Flow Chart:

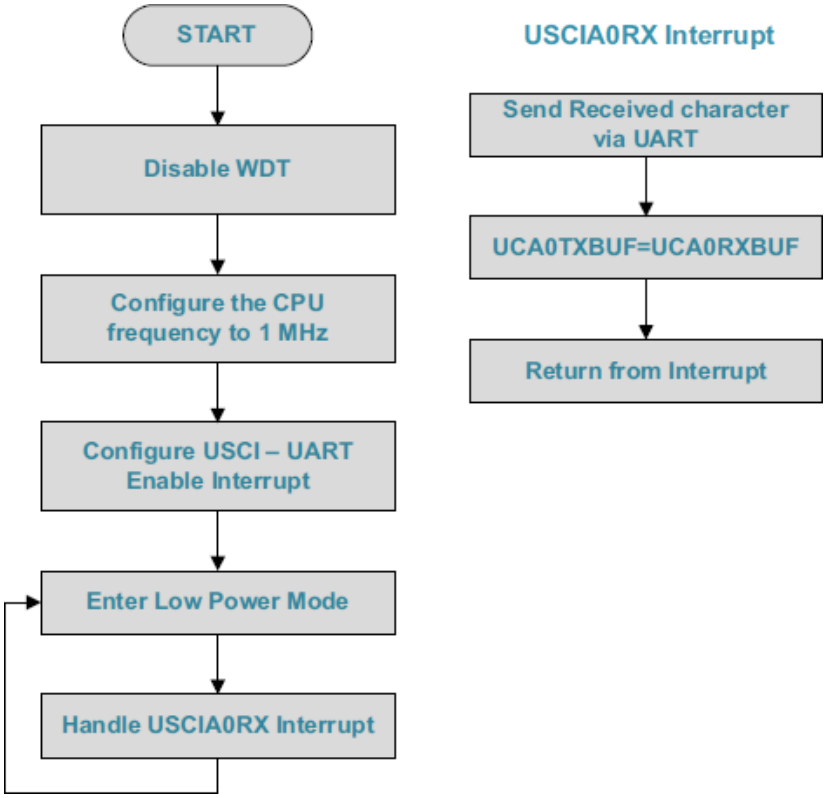


Figure: Flowchart for Serial Communication Using UART

Program:

```
#include
<msp430.h>intmain(
void)
{
    WDTCTL = WDTPW|WDTHOLD;        // Stop watchdog timer
    P1DIR |=BIT0;
   if(CALBC1_1MHZ==0xFF)           // If calibration constanterased
   {
    P1OUT |=BIT0;                  // Red LEDIndication
    while(1);                      // do not load, trapCPU!!
   }
   /* Use Calibration values for 1MHz Clock DCO*/
   DCOCTL= 0;                                  // Select lowest DCOx and MODx
    settings
   BCSCTL1=CALBC1_1MHZ;                        // SetDCO
```

```
  DCOCTL = CALDCO_1MHZ;
  /* Configure Pin P1.1 = RXD and P1.2 = TXD */
  P1SEL = BIT1 + BIT2;
  P1SEL2 = BIT1 + BIT2;
  /* Place UCA0 in Reset to be configured */
  UCA0CTL1 = UCSWRST;
  /* Configure */
  UCA0CTL0=0x00;                          // No parity, LSB first, 8-bit data,1
                                          //stop bit, UART, Asynchronous
  UCA0CTL1|=UCSSEL_2;                     // CLK =SMCLK
  UCA0BR0=104;                            // 1MHz/9600 = 104.166
  UCA0BR1 =0x00;
  UCA0MCTL=UCBRS0;                        // Modulation UCBRSx =1
  /* Take UCA0 out of reset */
  UCA0CTL1&=~UCSWRST;                     // InitializeUSCI_UART
  IE2|=UCA0RXIE;                          // Enable USCI_A0 RXinterrupt
  bis_SR_register(LPM3_bits+GIE);         // Enter LPM0 withinterrupt
}
/*Echo back RXed character, confirm TX buffer is ready first*/
#pragma vector=USCIAB0RX_VECTOR
interrupt void USCI0RX_ISR(void)
{
   while (!(IFG2&UCA0TXIFG));             // USCI_A0 TX bufferready?
   UCA0TXBUF=UCA0RXBUF;                   // TX RXedcharacter
}
```

Post lab Questions:

1. What is UCA0CTL0?

_____

_____

_____

2. What isUCA0CTL1?

_____

_____

_____

3. What is UCA0BR0 &UCA0BR1?

_____

_____

_____

Outcomes:

Upon completion of the experiment, the student will be able to

1. Develop an algorithm, the flow diagram, source code and perform the compilation.
2. Generate the required binary file which can be dumped into the controller and obtain the respective output control on the connected peripheral.
3. Verify the logic with the necessary hardware.

Grading

Date of Submission					
Marks Awarded (Max.Marks-10M)	Pre-Lab Questions (2M)	Observations (3M)	Post-Lab Questions (2M)	Viva (3M)	Total (10M)
Remarks					
Signature of the Evaluator with Date					

NOTES

MICROPROCESSORS & MICROCONTROLLERS LABORATORY

Department of Electronics and Communication
Engineering SVR Engineering College: Nandyal
Approved by AICTE, New Delhi & Affiliated to JNTUA,
Anantapuramu (An ISO 9001:2007 Certified Institution)
Near Ayyalur(v), Allagadda Rd, Nandyal, Andhra Pradesh 518503

PART - C

LAB BASED MINI PROJECT

LAB BASED MINI PROJECT
(Format to be submitted with Mini Project Report)

Title of the Project :

Abstract :

Introduction :

Objectives :

Project Description :

Outcomes :

Conclusions :

References :