



**Department of Computer Science and Engineering**

Jnanaprabha, Virgo Nagar Post, Bengaluru-560049

**Academic Year: 2023-24**

# **LABORATORY MANUAL**

**SEMESTER : IV**

**SUBJECT : MICROCONTROLLERS LABORATORY**

**SUBCODE : BCSCS402**

**NAME :** \_\_\_\_\_

**USN :** \_\_\_\_\_

**SECTION:** \_\_\_\_\_

## **PROGRAM OUTCOMES**

**PO1: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**PO2: Problem analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**PO3: Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**PO4: 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.

**PO5: Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

**PO6: 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.

**PO7: 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.

**PO8: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO9: Individual and Team Work:** Function effectively as an individual and as a member or leader in diverse teams, and in multi – disciplinary settings.

**PO10: 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.

**PO11: 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 multidisciplinary environments.

**PO12: Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life -long learning in the broadest context of technological change.



**Department of Computer Science and Engineering**

## **INSTITUTE VISION AND MISSION**

### **VISION**

The East Point College of Engineering and Technology aspires to be a globally acclaimed institution, recognized for excellence in engineering education, applied research and nurturing students for holistic development.

### **MISSION**

**M1:** To create engineering graduates through quality education and to nurture innovation, creativity and excellence in teaching, learning and research

**M2:** To serve the technical, scientific, economic and societal developmental needs of our communities

**M3:** To induce integrity, teamwork, critical thinking, personality development and ethics in students and to lay the foundation for lifelong learning



Department of Computer Science and Engineering

### **DEPARTMENT VISION AND MISSION**

#### **VISION**

The department aspires to be a Centre of excellence in Computer Science & Engineering to develop competent professionals through holistic development.

#### **MISSION**

**M1:** To create successful Computer Science Engineering graduates through effective pedagogies, the latest tools and technologies, and excellence in teaching and learning.

**M2:** To augment experiential learning skills to serve technical, scientific, economic, and social developmental needs.

**M3:** To instil integrity, critical thinking, personality development, and ethics in students for a successful career in Industries, Research, and Entrepreneurship.

#### **PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

**PEO 1:** To produce graduates who can perform technical roles to contribute effectively in software industries and R&D Centre

**PEO 2:** To produce graduates having the ability to adapt and contribute in key domains of computer science and engineering to develop competent solutions.

**PEO 3:** To produce graduates who can provide socially and ethically responsible solutions while adapting to new trends in the domain to carve a successful career in the industry

## **PROGRAM SPECIFIC OUTCOMES (PSOs)**

**PSO1:** To conceptualize, model, design, simulate, analyse, develop, test, and validate computing systems and solve technical problems arising in the field of computer science & engineering.

**PSO2:** To specialize in the sub-areas of computer science & engineering systems such as cloud computing, Robotic Process Automation, cyber security, big data analytics, user interface design, and IOT to meet industry requirements.

**PSO3:** To build innovative solutions to meet the demands of the industry using appropriate tools and techniques.

## **COURSE LEARNING OBJECTIVES**

CLO 1: Understand the fundamentals of ARM-based systems and basic architecture of CISC and RISC.

CLO 2: Familiarize with ARM programming modules along with registers, CPSR and Flags.

CLO 3: Develop ALP using various instructions to program the ARM controller.

CLO 4: Understand the Exceptions and Interrupt handling mechanism in Microcontrollers.

CLO 5: Discuss the ARM Firmware packages and Cache memory policies.

## **COURSE OUTCOMES**

At the end of the course the student will be able to:

CO1: Explain the ARM Architectural features and Instructions.

CO2: Develop programs using ARM instruction set for an ARM Microcontroller.

CO3: Explain C-Compiler Optimizations and portability issues in ARM Microcontroller.

CO4: Apply the concepts of Exceptions and Interrupt handling mechanisms in developing applications.

CO5: Demonstrate the role of Cache management and Firmware in Microcontrollers.

MICROCONTROLLERS			
Course Code	BCS402	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	3:0:2:0	SEE Marks	50
Total Hours of Pedagogy	40 T + 8-10 P	Total Marks	100
Credits	04	Exam Hours	03
<b>Course Objectives:</b> CLO 1: Understand the fundamentals of ARM-based systems and basic architecture of CISC and RISC. CLO 2: Familiarize with ARM programming modules along with registers, CPSR and Flags. CLO 3: Develop ALP using various instructions to program the ARM controller. CLO 4: Understand the Exceptions and Interrupt handling mechanism in Microcontrollers. CLO 5: Discuss the ARM Firmware packages and Cache memory policies.			
<b>Teaching-Learning Process</b> These are sample Strategies, which teachers can use to accelerate the attainment of the various course outcomes. <ol style="list-style-type: none"> <li>1. Lecturer method (L) needs not to be only a traditional lecture method, but alternative effective teaching methods could be adopted to attain the outcomes.</li> <li>2. Use of Video/Animation to explain functioning of various concepts.</li> <li>3. Encourage collaborative (Group Learning) Learning in the class.</li> <li>4. Ask at least three HOT (Higher order Thinking) questions in the class, which promotes critical thinking.</li> <li>5. Adopt Problem Based Learning (PBL), which fosters students' Analytical skills, develop design thinking skills such as the ability to design, evaluate, generalize, and analyze information rather than simply recall it.</li> <li>6. Introduce Topics in manifold representations.</li> <li>7. Show the different ways to solve the same problem with different circuits/logic and encourage the students to come up with their own creative ways to solve them.</li> <li>8. Discuss how every concept can be applied to the real world - and when that's possible, it helps improve the students understanding.</li> <li>9. Use any of these methods: Chalk and board, Active Learning, Case Studies.</li> </ol>			
<b>Module-1</b>			
<b>ARM Embedded Systems:</b> The RISC design philosophy, The ARM Design Philosophy, Embedded System Hardware, Embedded System Software.  <b>ARM Processor Fundamentals:</b> Registers, Current Program Status Register, Pipeline, Exceptions, Interrupts, and the Vector Table, Core Extensions  <b>Textbook 1: Chapter 1 - 1.1 to 1.4, Chapter 2 - 2.1 to 2.5</b> <b>RBT: L1, L2, L3</b>			
<b>Module-2</b>			
<b>Introduction to the ARM Instruction Set:</b> Data Processing Instructions, Branch Instructions, Software Interrupt Instructions, Program Status Register Instructions, Coprocessor Instructions, Loading Constants.  <b>Textbook 1: Chapter 3 - 3.1 to 3.6</b> <b>RBT: L1, L2, L3</b>			
<b>Module-3</b>			
<b>C Compilers and Optimization:</b> Basic C Data Types, C Looping Structures, Register Allocation, Function Calls, Pointer Aliasing, Portability Issues.  <b>Textbook 1: Chapter 5.1 to 5.7 and 5.13</b> <b>RBT: L1, L2, L3</b>			
<b>Module-4</b>			

**Exception and Interrupt Handling:** Exception handling, ARM processor exceptions and modes, vector table, exception priorities, link register offsets, interrupts, assigning interrupts, interrupt latency, IRQ and FIQ exceptions, basic interrupt stack design and implementation. Firmware: Firmware and bootloader, ARM firmware suite, Red Hat redboot, Example: sandstone, sandstone directory layout, sandstone code structure.

**Textbook 1: Chapter 9.1 and 9.2, Chapter 10**  
**RBT: L1, L2, L3**

#### Module-5

**CACHES:** The Memory Hierarchy and Cache Memory, Caches and Memory Management Units: CACHE Architecture: Basic Architecture of a Cache Memory, Basic Operation of a Cache Controller, The Relationship between Cache and Main Memory, Set Associativity, Write Buffers, Measuring Cache Efficiency, CACHE POLICY: Write Policy—Writeback or Writethrough, Cache Line Replacement Policies, Allocation Policy on a Cache Miss. Coprocessor 15 and caches.

**Textbook 1: Chapter 12.1 to 12.4**  
**RBT: L1, L2, L3**

Sl.No.	Experiments
<b>Module – 1</b>	
1.	Using Keil software, observe the various Registers, Dump, CPSR, with a simple Assembly Language Programs (ALP).
<b>Module – 2</b>	
2.	Develop and simulate ARM ALP for Data Transfer, Arithmetic and Logical operations (Demonstrate with the help of a suitable program).
3.	Develop an ALP to multiply two 16-bit binary numbers.
4.	Develop an ALP to find the sum of first 10 integer numbers.
5.	Develop an ALP to find the largest/smallest number in an array of 32 numbers.
6.	Develop an ALP to count the number of ones and zeros in two consecutive memory locations.
<b>Module – 3</b>	
7.	Simulate a program in C for ARM microcontroller using KEIL to sort the numbers in ascending/descending order using bubble sort.
8.	Simulate a program in C for ARM microcontroller to find factorial of a number.
9.	Simulate a program in C for ARM microcontroller to demonstrate case conversion of characters from upper to lowercase and lower to uppercase.
<b>Module – 4 and 5</b>	
10.	Demonstrate enabling and disabling of Interrupts in ARM.
11.	Demonstrate the handling of divide by zero, Invalid Operation and Overflow exceptions in ARM.

#### Course outcome (Course Skill Set)

At the end of the course, the student will be able to:

- CO 1. Explain C-Compilers and optimization
- CO 2. Describe the ARM microcontroller's architectural features and program module.
- CO 3. Apply the knowledge gained from programming on ARM to different applications.
- CO 4. Program the basic hardware components and their application selection method.
- CO 5. Demonstrate the need for a real-time operating system for embedded system applications.

**Assessment Details (both CIE and SEE)**

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures not less than 35% (18 Marks out of 50) in the semester-end examination (SEE), and a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together

**Continuous Internal Evaluation:**

Three Unit Tests each of **20 Marks (duration 01 hour)**

1. First test at the end of 5<sup>th</sup> week of the semester
2. Second test at the end of the 10<sup>th</sup> week of the semester
3. Third test at the end of the 15<sup>th</sup> week of the semester

Two assignments each of **10 Marks**

4. First assignment at the end of 4<sup>th</sup> week of the semester
5. Second assignment at the end of 9<sup>th</sup> week of the semester

Practical Sessions need to be assessed by appropriate rubrics and viva-voce method. This will contribute to **20 marks**.

- Rubrics for each Experiment taken average for all Lab components – 15 Marks.
- Viva-Voce– 5 Marks (more emphasized on demonstration topics)

The sum of three tests, two assignments, and practical sessions will be out of 100 marks and will be **scaled down to 50 marks**

(to have a less stressed CIE, the portion of the syllabus should not be common /repeated for any of the methods of the CIE. Each method of CIE should have a different syllabus portion of the course).

**CIE methods /question paper has to be designed to attain the different levels of Bloom's taxonomy as per the outcome defined for the course.**

**Semester End Examination:**

Theory SEE will be conducted by University as per the scheduled timetable, with common questionpapers for the subject (**duration 03 hours**)

1. The question paper will have ten questions. Each question is set for 20 marks. Marks scored shall be proportionally reduced to 50 marks
2. There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), **should have a mix of topics** under that module.

The students have to answer 5 full questions, selecting one full question from each module

**Suggested Learning Resources:****Textbooks**

1. Andrew N Sloss, Dominic Symes and Chris Wright, ARM system developers guide, Elsevier, Morgan Kaufman publishers, 2008.
2. Shibu K V, "Introduction to Embedded Systems", Tata McGraw Hill Education, Private Limited, 2<sup>nd</sup> Edition.

**Reference Books**

1. Raghunandan. G.H, Microcontroller (ARM) and Embedded System, Cengage learning Publication, 2019
2. The Insider's Guide to the ARM7 Based Microcontrollers, Hitex Ltd., 1st edition, 2005.
3. Steve Furber, ARM System-on-Chip Architecture, Second Edition, Pearson, 2015.
4. Raj Kamal, Embedded System, Tata McGraw-Hill Publishers, 2nd Edition, 2008.

**Weblinks and Video Lectures (e-Resources):****Activity Based Learning (Suggested Activities in Class)/ Practical Based learning**



Index					
Sl. No.	Program List	CO	PO, PSO	RBT	Pg. No
	<b>PART A</b>				
1	Using Keil software, observe the various Registers, Dump, CPSR, with a simple Assembly Language Programs (ALP).	CO1	PO1, PO2, PO3, PO4, PO5, PSO1,2,3	L3	1
2	Develop and simulate ARM ALP for Data Transfer, Arithmetic and Logical operations (Demonstrate with the help of a suitable program).	CO2	PO1, PO2, PO3, PO4, PO5, PSO1,2,3	L3	2
3	Develop an ALP to multiply two 16-bit binary numbers.	CO2	PO1, PO2, PO3, PO4, PO5, PSO1,2,3	L3	3
4	Develop an ALP to find the sum of first 10 integer number	CO2	PO1, PO2, PO3, PO4, PO5, PSO1,2,3	L3	4
5	Develop an ALP to find the largest/smallest number in an array of 32 number	CO2	PO1, PO2, PO3, PO4, PO5, PSO1,2,3	L3	5
6	Develop an ALP to count the number of ones and zeros in two consecutive memory locations	CO2	PO1, PO2, PO3, PO4, PO5, PSO1,2,3	L3	6
7	Simulate a program in C for ARM microcontroller using KEIL to sort the numbers in ascending/descending order using bubble sort.	CO3	PO1, PO2, PO3, PO4, PO5, PSO1,2,3	L2	7
8	Simulate a program in C for ARM microcontroller to find factorial of a number.	CO3	PO1, PO2, PO3, PO4, PO5, PSO1,2,3	L2	8
9	Simulate a program in C for ARM microcontroller to demonstrate case conversion of characters from upper to lowercase and lower to uppercase.	CO3	PO1, PO2, PO3, PO4, PO5, PSO1,2,3	L2	9
10	Demonstrate enabling and disabling of Interrupts in ARM	CO4, CO5	PO1, PO2, PO3, PO4, PO5, PSO1,2,3	L3	10

11	Demonstrate the handling of divide by zero, Invalid Operation and Overflow exceptions in ARM	CO5	PO1, PO2, PO3, PO4, PO5, PSO1,2,3	L3	10
Viva Question & Answers					25

### Course Articulation Matrix

COs	POs												PSOs		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3	2	1	3	-	-	-	-	-	-	-	3	2	1
CO2	3	3	3	1	3	-	-	-	-	-	-	-	3	2	1
CO3	3	3	3	1	3	-	-	-	-	-	-	-	3	2	1
CO4	3	3	3	1	3	-	-	-	-	-	-	-	3	2	1
CO5	3	3	3	1	3	-	-	-	-	-	-	-	3	2	1

**3 - High Correlation**

**2 - Medium Correlation**

**1 – Low Correlation**

# 1. Using Keil software, observe the various Registers, Dump, CPSR, with a simple Assembly Language Programs (ALP).

```

AREA Multiply, CODE, READONLY ;Define a logical area named PRG6
ENTRY                          ; Entry point where the code starts
MOV R1, #2                     ; Data transfer R1=2
MOV R3, #8                     ; Data transfer R1=8
END

```

Register	Value
<b>Current</b>	
R0	0x00000000
R1	0x00000007
R2	0x00000000
R3	0x00000008
R4	0x0000000F
R5	0x00000001
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00080004
R15 (PC)	0x0000000C

<b>CPSR</b>	0x000000D7
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x17
<b>SPSR</b>	0x000000D7
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x17

## 2. Develop and simulate ARM ALP for Data Transfer, Arithmetic and Logical operations (Demonstrate with the help of a suitable program).

```

AREA PRG6, CODE, READONLY ; Define a logical area named PRG6
ENTRY                      ; Entry point where the code starts
LDR R0, =5                 ; Data transfer – R0=5
LDR R1, =3                 ; R1=3
ADD R2, R0, R1             ; Arithmetic: R2 = 8 (5 + 3)
SUB R3, R0, R1             ; SUB: R3 = 2 (5 - 3)
MUL R4, R0, R1             ; MUL: R4 = F (5 * 3 = 15 = F in hexadecimal)
AND R5, R0, R1             ; Logical AND: R5 = 1 (5 && 3)
ORR R6, R0, R1             ; Logical OR: R6 = 7 (5 || 3)
EOR R7, R0, R1             ; Logical XOR: R7 = 6 (5 ^ 3)
END                         ; End of the program

```

Register	Value
<b>Current</b>	
R0	0x00000005
R1	0x00000003
R2	0x00000008
R3	0x00000002
R4	0x0000000F
R5	0x00000001
R6	0x00000007
R7	0x00000006
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00080004
<b>R15 (PC)</b>	<b>0x00000010</b>
+ CPSR	0x000000D7
+ SPSR	0x000000D3
+ User/System	
+ Fast Interrupt	
+ Interrupt	
+ Supervisor	

### 3. Develop an ALP to multiply two 16-bit binary numbers.

AREA Multiply, CODE, READONLY

ENTRY

```

LDR      R0, =NUM          ; load address of multiplicand
LDRH     R1, [R0]           ; load First number
LDRH     R2, [R0,#2]        ; load Second number
MUL      R3, R1, R2         ; R3 = R1 x R2
STOP B STOP                ; all done
NUM DCW  0X1222,0X1133     ; Declaration of no's to be multiply

```

END

### OUTPUT :

Register	Value
<b>Current</b>	
R0	0x00000010
R1	0x00001222
R2	0x00001133
R3	0x0137DEC6
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00080004
R15 (PC)	0x0000000C
CPSR	0x000000D7
<b>SPSR</b>	<b>0x000000D7</b>
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
<b>Abort</b>	
Undefined	

#### 4. Write a program to find the sum of first 10 integer numbers.

```

        AREA ADD1TO10, CODE, READONLY
        ENTRY
        MOV R1, #10                ;length of array
        LDR R2, =ARRAY             ;Load the starting address of the array
        MOV R4, #0                 ;Initial sum
NEXT    LDR R3, [R2], #4           ;Load first integer of the array in R3
        ADD R4,R4,R3               ;R4=sum of integers
        SUBS R1,R1,#1
        BNE NEXT                  ;repeat until R1=0
        STOP B STOP

        ARRAY DCD 1,2,3,4,5,6,7,8,9,10
        END

```

### OUTPUT:

Register	Value
<b>Current</b>	
R0	0x00000000
R1	0x00000000
R2	0x0000004C
R3	0x0000000A
R4	0x00000037
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000020
CPSR	0x600000D3
SPSR	0x00000000
+	User/System
+	Fast Interrupt
+	Interrupt
+	<b>Supervisor</b>
+	Abort
+	Undefined

### 5. Write a program to find the largest/smallest number in an array of 32 numbers.

```

AREA LARGE, CODE, READONLY
ENTRY
MOV R5, #5                ;R5 = length of array - 1
LDR R1, =ARRAY            ;load starting addressing of array
LDR R2, [R1], #4          ;load 1st element of array
LOOP LDR R4, [R1], #4      ;load next element of array
CMP R2, R4                ;compare 1st and 2nd element
BHI NEXT                  ;R2=largest value
MOV R2, R4                ;R2=largest value
NEXT SUBS R5, R5, #1       ;decrement the counter after every comparison
BNE LOOP                  ;repeat until R5=0
STOP B STOP

```

```

ARRAY DCD 0X23, 0X45, 0X65, 0X76, 0X12, 0X99

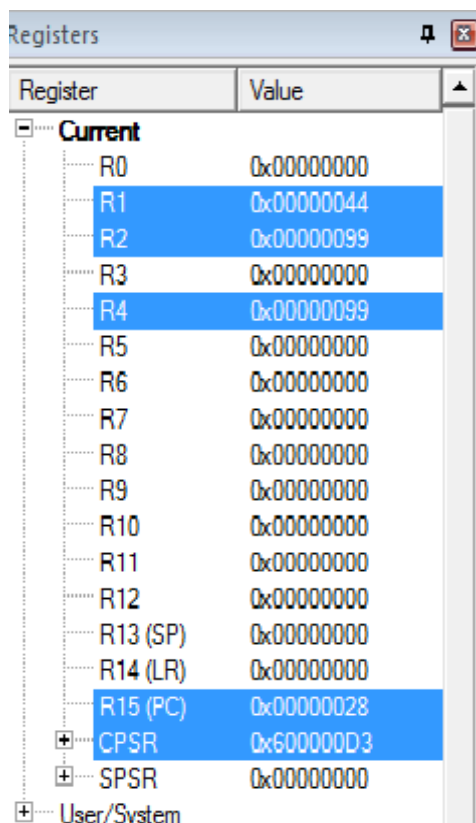
```

```

END

```

### OUTPUT :



Register	Value
<b>Current</b>	
R0	0x00000000
R1	0x00000044
R2	0x00000099
R3	0x00000000
R4	0x00000099
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000028
CPSR	0x600000D3
SPSR	0x00000000
User/System	

*Note: Use BLS instead of BHI for finding the smallest number.*



**6. Write a program to count the number of ones and zeros in two consecutive memory locations.**

```

AREA ONEZERO, CODE, READONLY ENTRY
MOV R2,#0           ;Counter for ones
MOV R3,#0           ;Counter for zeros
MOV R7,#2           ;Counter of 2 numbers
LDR R6,=LOOKUP      ;Load starting address of numbers LOOP
MOV R1,#32          ;Number of bits in each number
LDR R0,[R6]          ;Load 1st number to r0 NEXTBIT
MOVS R0,R0,ROR #1    ;Check the bit is one or zero
BHI ONES            ; IF CF=1 increment r2 else increment r3 ZEROS
ADD R3,R3,#1         ;R3 stores count of 0s
B REPEAT
ONES                ;R2 stores count of 1s
REPEAT              ; Decrement the bit counter by 1 till 0 BNE
NEXTBIT             ; Repeat until r1=0
ADD R6,R6,#4         ; Load r6=address of next number
SUBS R7,R7,#1        ; Decrement the number counter by 1 till 0 BNE
LOOP
STOP                B STOP
LOOKUP DCD 0X5,0X7    ; Memory address of lookup table END

```

## OUTPUT :

Register	Value
<b>Current</b>	
R0	0x00000007
R1	0x00000000
R2	0x00000005
R3	0x0000003B
R4	0x00000000
R5	0x00000000
R6	0x0000004C
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000040
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
<b>Supervisor</b>	

**7. Simulate a program in C for ARM microcontroller using KEIL to sort the numbers in ascending/descending order using bubble sort.**

```
#include <stdio.h>

void swap(int* arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

void bubbleSort(int arr[], int n) {
    int i, j;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                swap(arr, j, j + 1);
            }
        }
    }
}

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int arr[] = {5, 1, 4, 2, 8}; // Your input array
    int N = sizeof(arr) / sizeof(arr[0]);

    bubbleSort(arr, N);

    printf("Sorted array: ");
    printArray(arr, N);

    return 0;
}
```

**8. Simulate a program in C for ARM microcontroller to find factorial of a number.**

```
#include <stdio.h>

unsigned long long factorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    } else {
        unsigned long long result = 1;
        for (int i = 2; i <= n; ++i) {
            result *= i;
        }
        return result;
    }
}

int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);

    if (num < 0) {
        printf("Error! Factorial of a negative number doesn't exist.\n");
    } else {
        unsigned long long fact = factorial(num);
        printf("Factorial of %d = %llu\n", num, fact);
    }

    return 0;
}
```

---

**9. Simulate a program in C for ARM microcontroller to demonstrate case conversion of characters from upper to lowercase and lower to uppercase.**

```
#include <stdio.h>

// Convert uppercase to lowercase (and vice versa) using XOR
char convertCase(char c) {
    if (c >= 'A' && c <= 'Z') {
        // Convert uppercase to lowercase
        return c ^ 0x20;
    } else if (c >= 'a' && c <= 'z') {
        // Convert lowercase to uppercase
        return c ^ 0x20;
    }
    // Return unchanged for non-alphabetic characters
    return c;
}

int main() {
    char input[] = "Hello, World!"; // Your input string
    int len = sizeof(input) - 1; // Exclude the null terminator

    for (int i = 0; i < len; ++i) {
        input[i] = convertCase(input[i]);
    }

    printf("Converted string: %s\n", input);

    return 0;
}
```





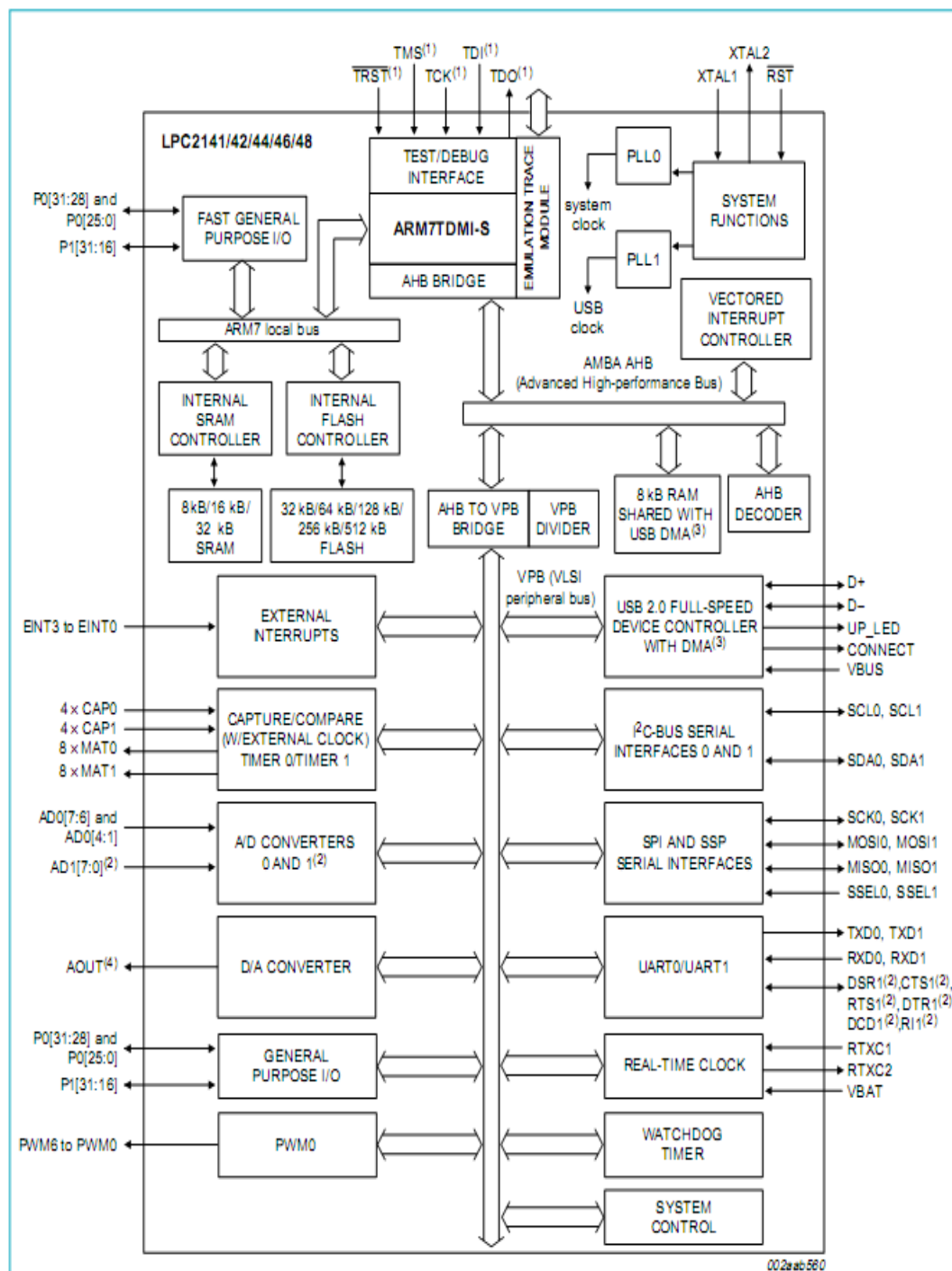
**INTRODUCTION OF THE ARM LPC 2148:**

The LPC2148 microcontrollers are based on a 32 bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combines the microcontroller with embedded high speed flash memory of 512 kB FEATURES 32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package. 8 to 40 kB of on-chip static RAM and 32 to 512 kB of on-chip flash program memory. 128 bit wide interface/accelerator enables high speed 60 MHz operation. In-System/In-Application Programming (ISP/IAP) via on-chip boot-loader software. Single flash sector or full chip erase in 400 ms and programming of 256 bytes in 1 ms. Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip Real Monitor software and high speed tracing of instruction execution. USB 2.0 Full Speed compliant Device Controller with 2 kB of endpoint RAM. In addition, the LPC2146/8 provide 8 kB of on-chip RAM accessible to USB by DMA.

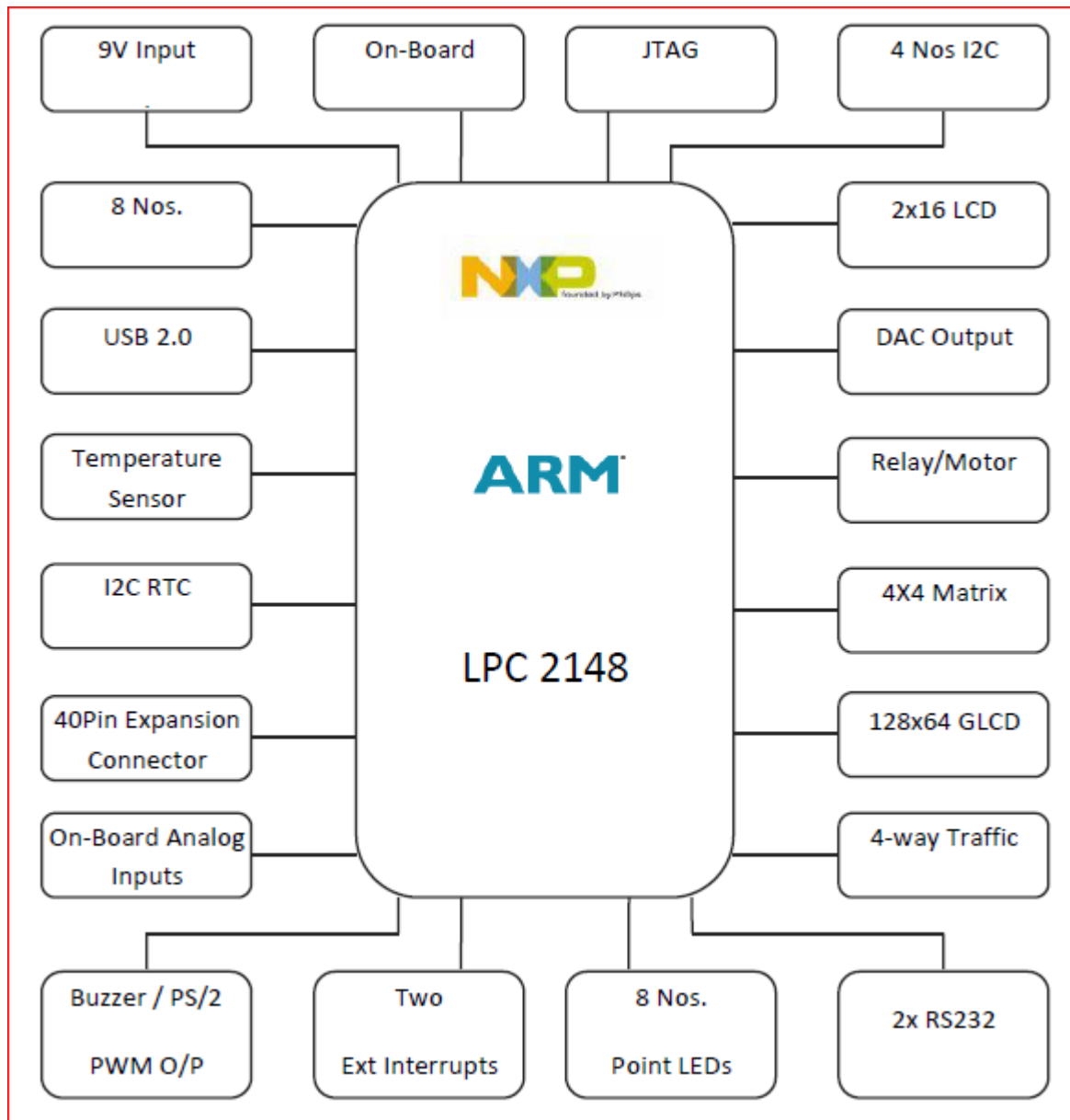
One or two (LPC2141/2 vs. LPC2144/6/8) 10-bit A/D converters provide a total of 6/14 analog inputs, with conversion times as low as 2.44 ms per channel. Single 10-bit D/A converter provides variable analog output.

- Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
- Low power real-time clock with independent power and dedicated 32 kHz clock input.
- Multiple serial interfaces including two UARTs (16C550), two Fast I2C-bus (400 kbit/s), SPI and SSP with buffering and variable data length capabilities.
- Vectored interrupt controller with configurable priorities and vector addresses.
- Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package.
- Up to nine edge or level sensitive external interrupt pins available.

## BLOCK DIAGRAM OF LPC 2148 ARM MICRO CONTROLLER





**GENERAL BLOCK DIAGRAM**

**LPC 2148 ARM Micro Controller Development Board**

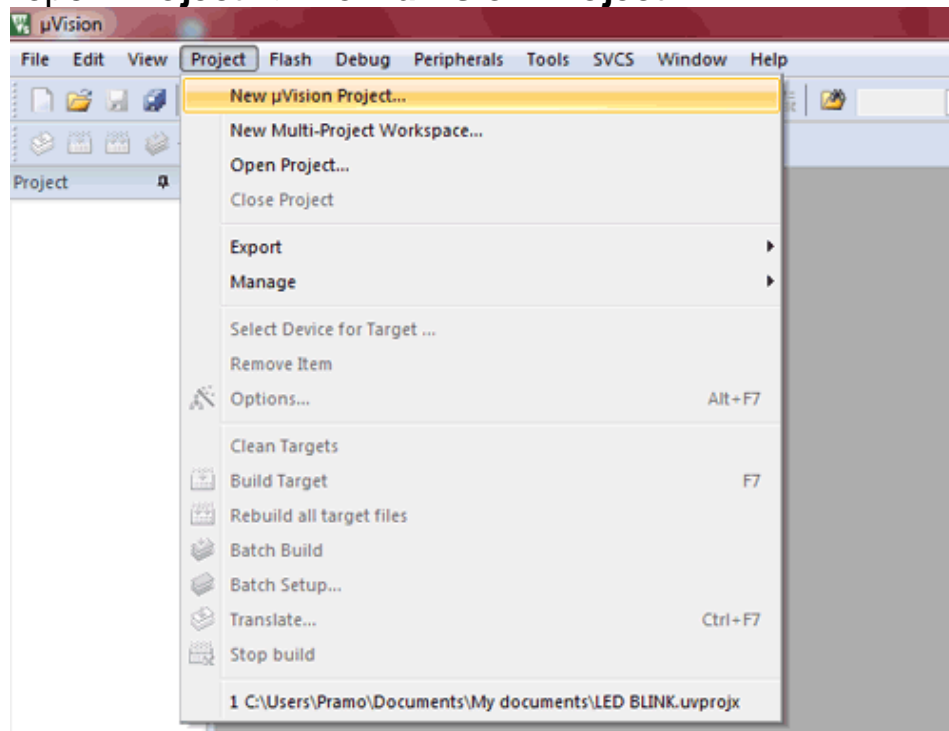
Steps involved to Create a New Project in Keiluvision 5 for ARM7 LPC2148

Step 1: Open Keiluvision 5

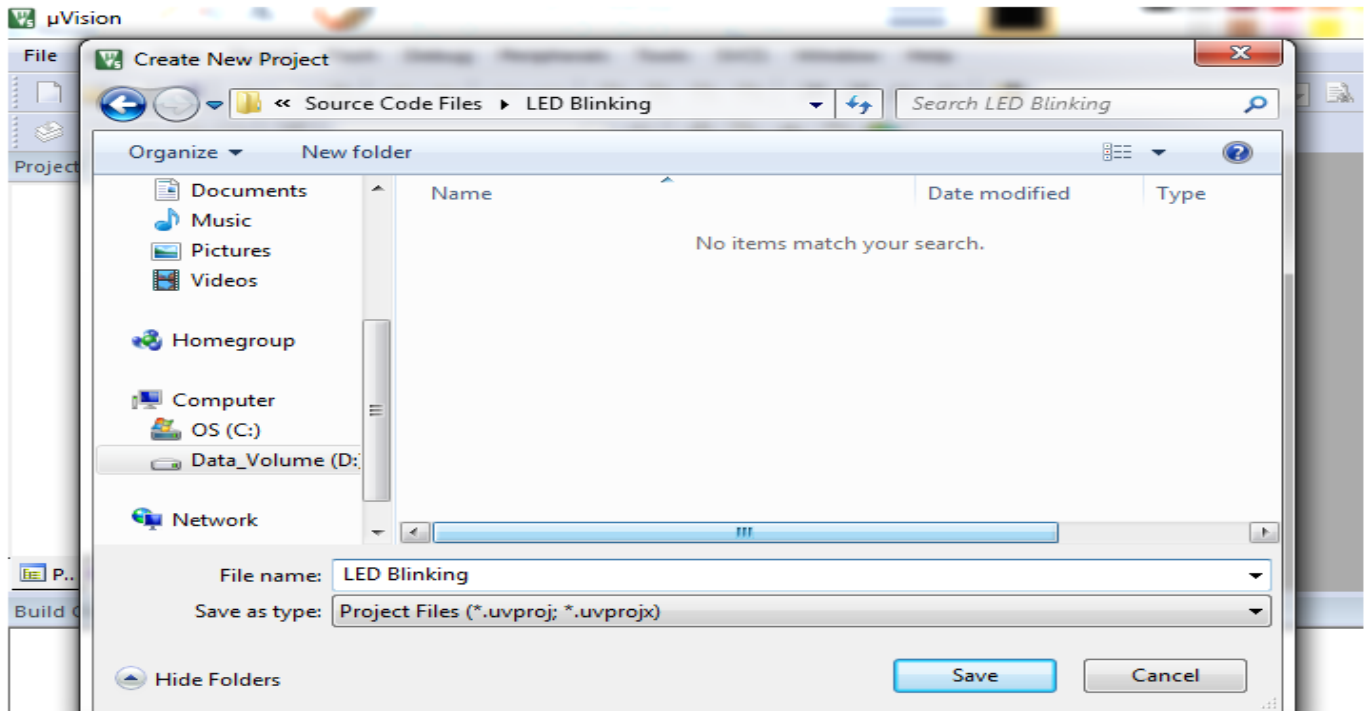


5

**Step 2: Now open Project → New uVision Project**

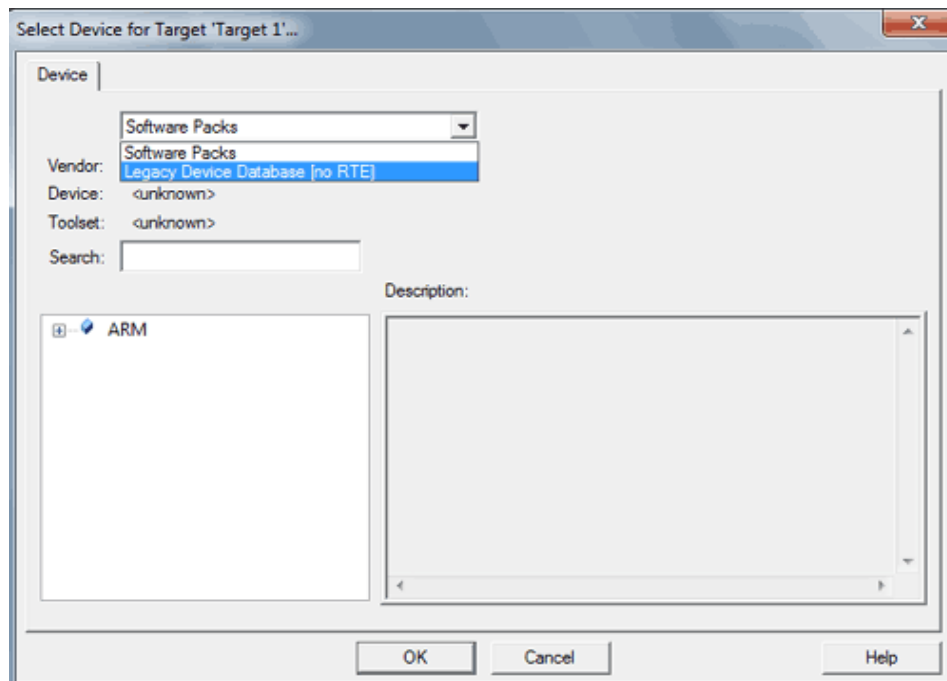


**Step 3: Give Name to Project e.g. "LED blinking" and save it.**

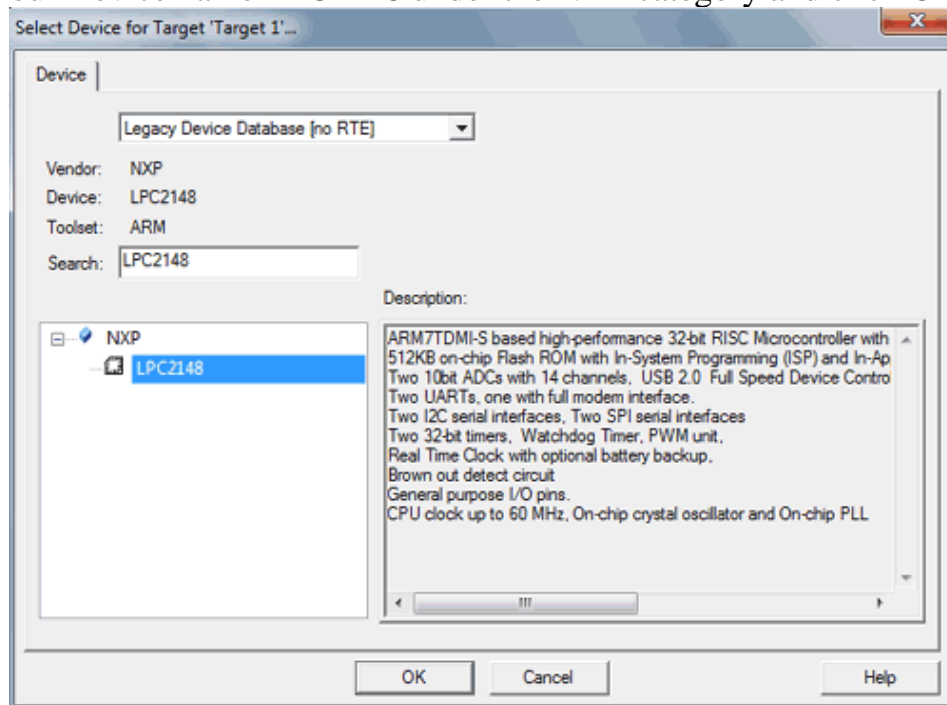


**Step 4:** then appears the popup box **Select Device for Target “Target1”**. Click the drop down menu where you need to select **Legacy Device Database [no RTE]**.

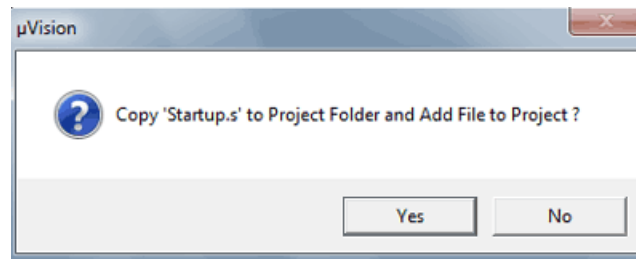
NOTE: If you need to install ARM7 packages this Legacy device database packages.



Step 5: Select our Device name LPC2148 under the NXP category and click OK

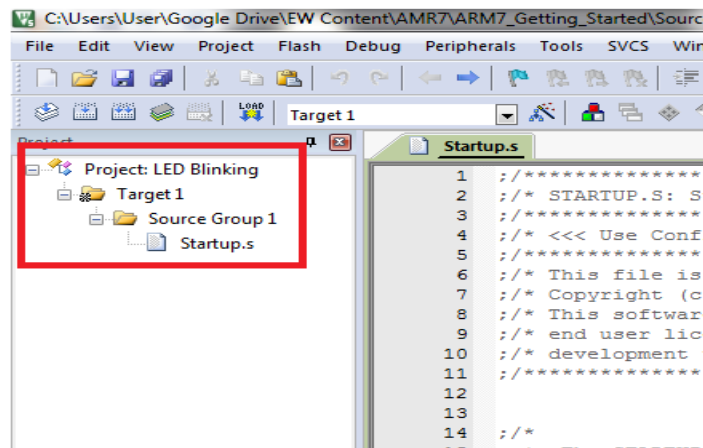


Step 6: A dialogue box appears to copy Startup.s to project folder, just click yes

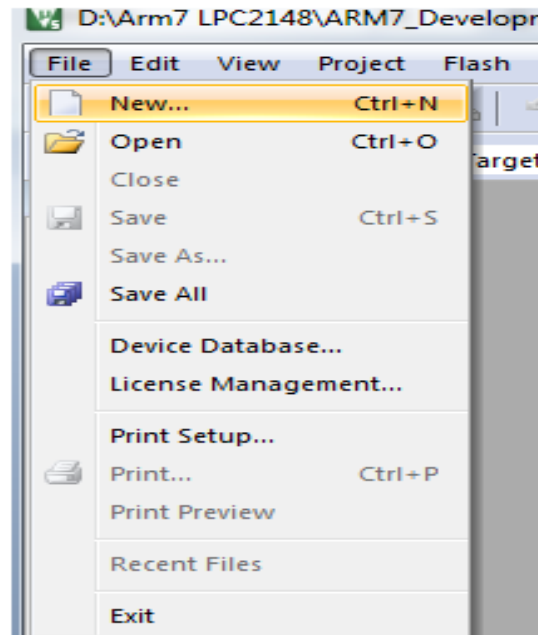


**Step 7:** Now it appears like the below image.

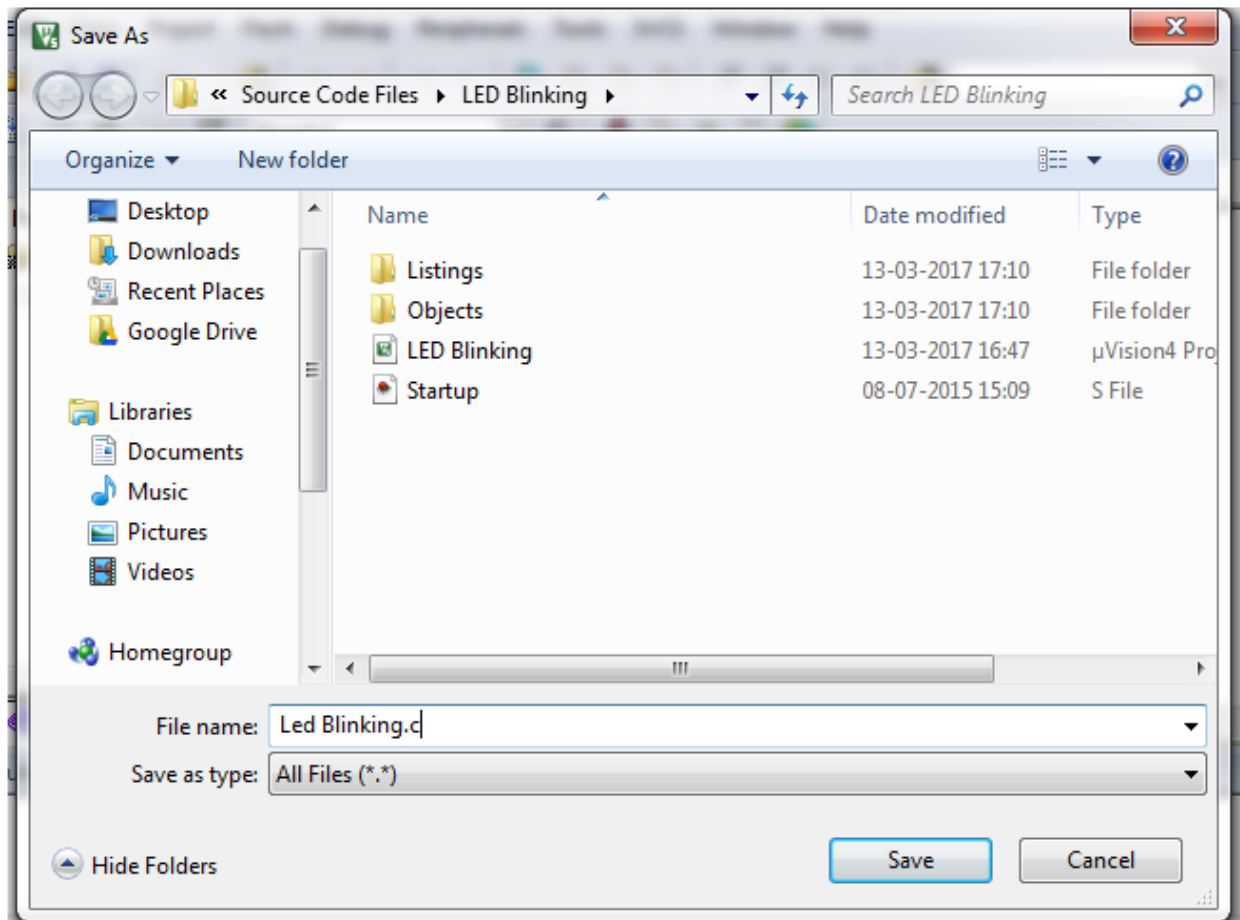
The project name and its folders can be seen on the left side in the project window.



1. Now go to File tab and add **New** file from the menu.

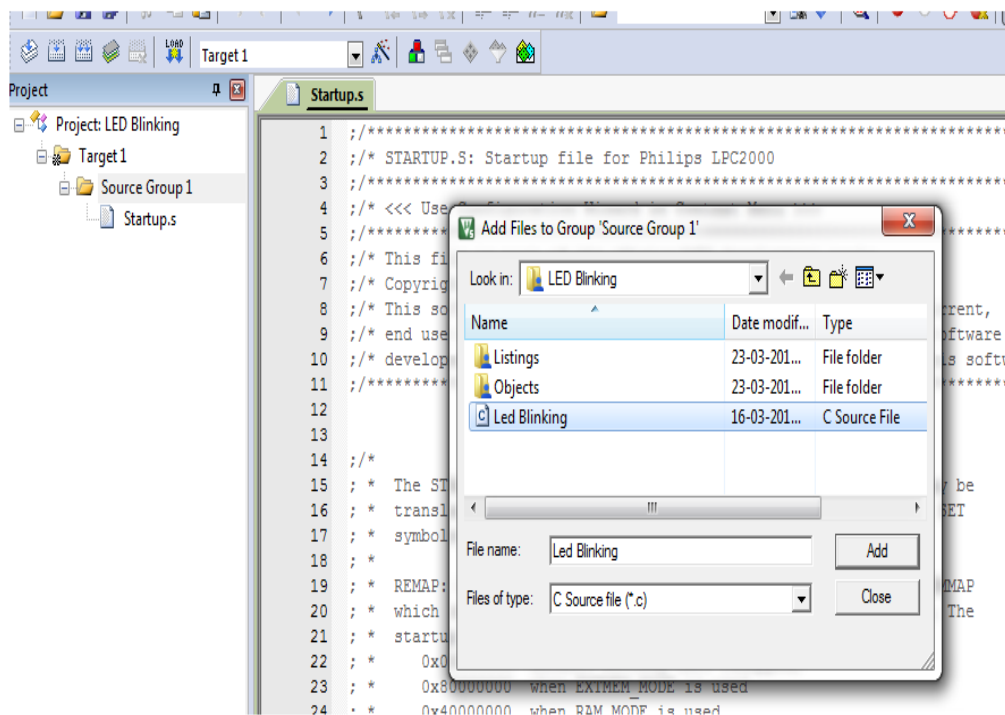
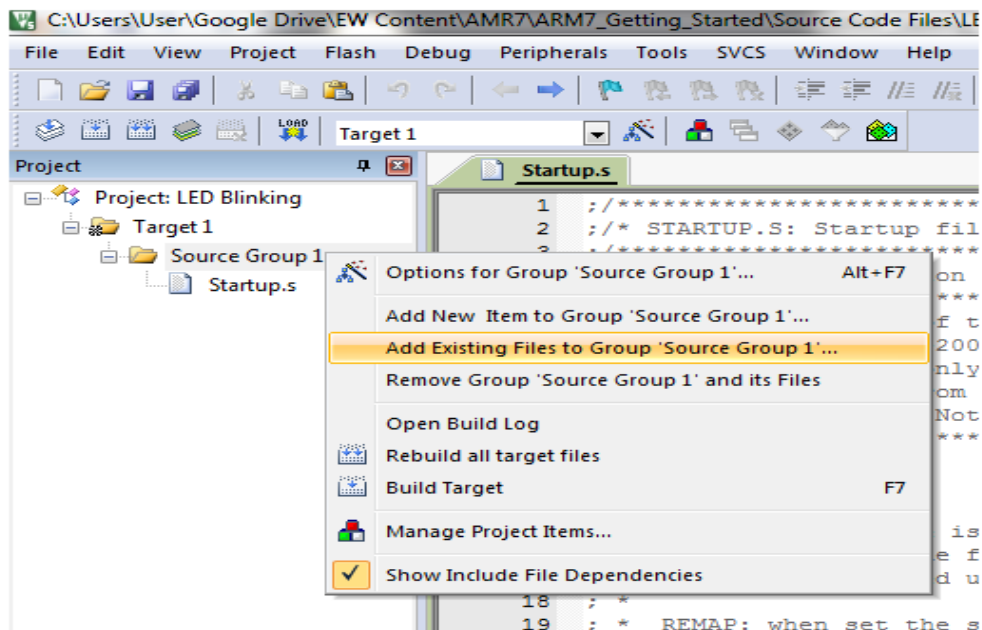


2. Save the file from the previous step with a specific name. Add .c extension to the file name.Eg.(ledblinking.c)



3. Add this .c extensionfile to Source Group folder in the project window by right clicking on Source Group1 folder and selecting **Add Existing Files to Group 'Source Group1'...**

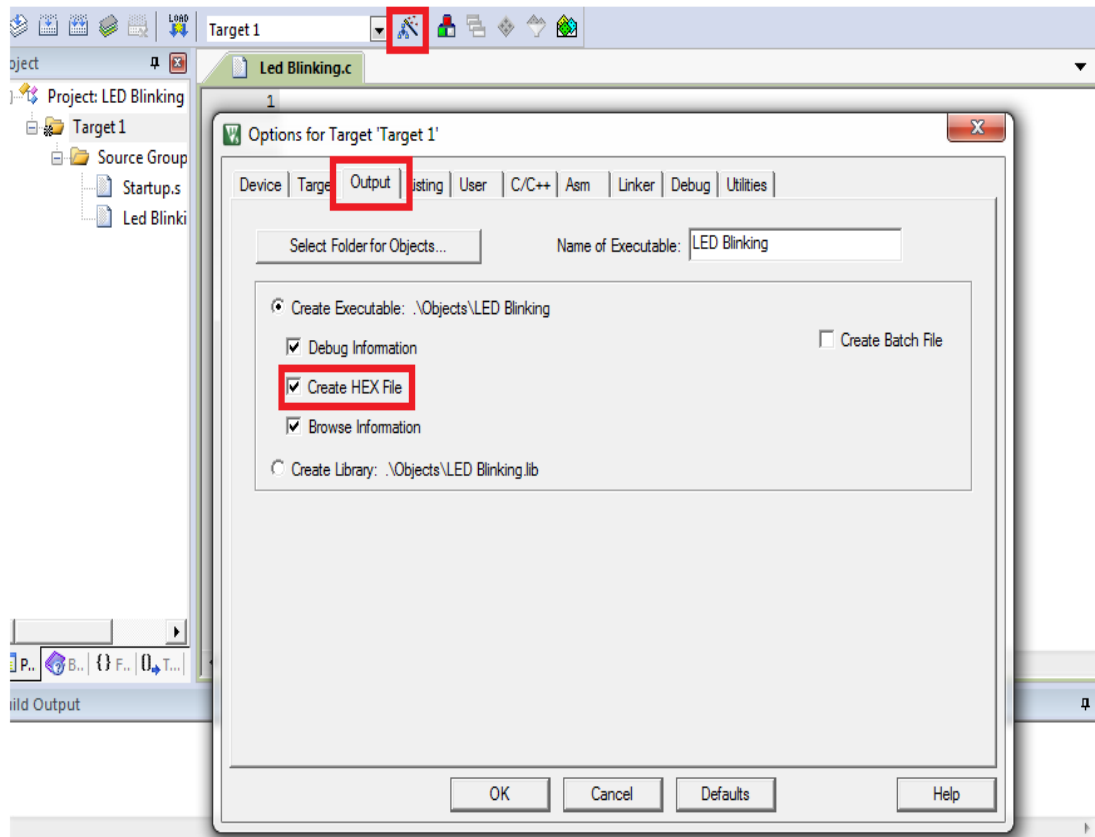




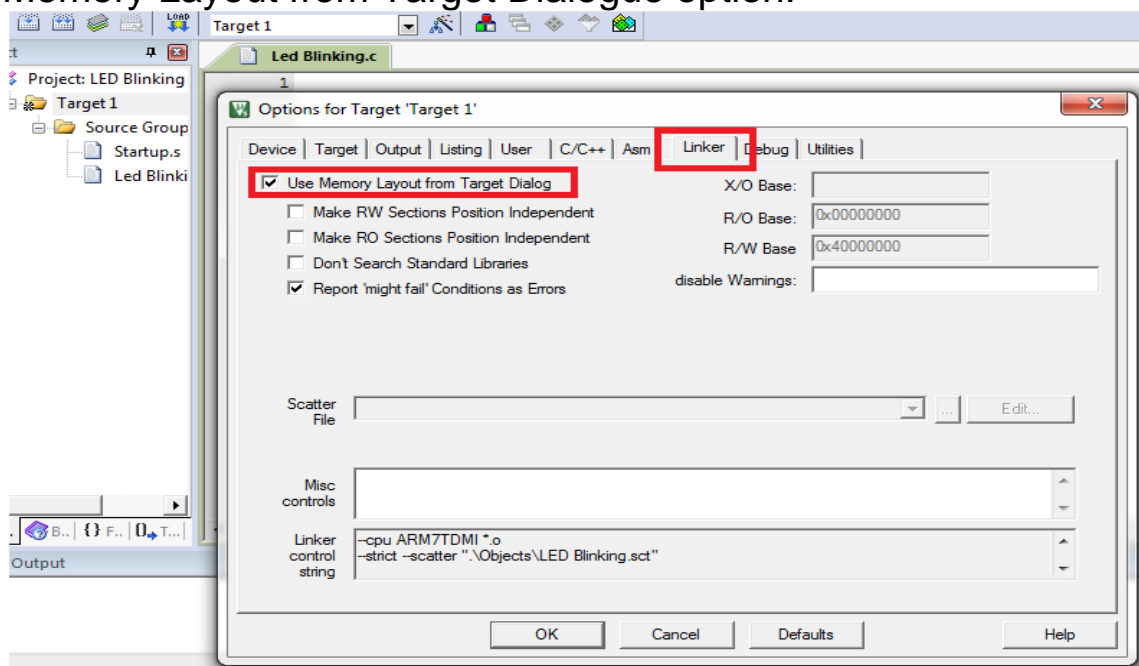
Select the previously saved file from the window that pops up and add it to the Source Group1. In our case, LED Blinking.c

4. Now click on the **Options for Target 'Target1'...** symbol shown in red box in the image below or press **Alt+F7** or right click on Target1 and click on **Options for Target 'Target1'...**. Options for target window will open. Go to the Output tab in that window. Tick 'v' Create HEX File option. Which will be burn into the microcontroller.

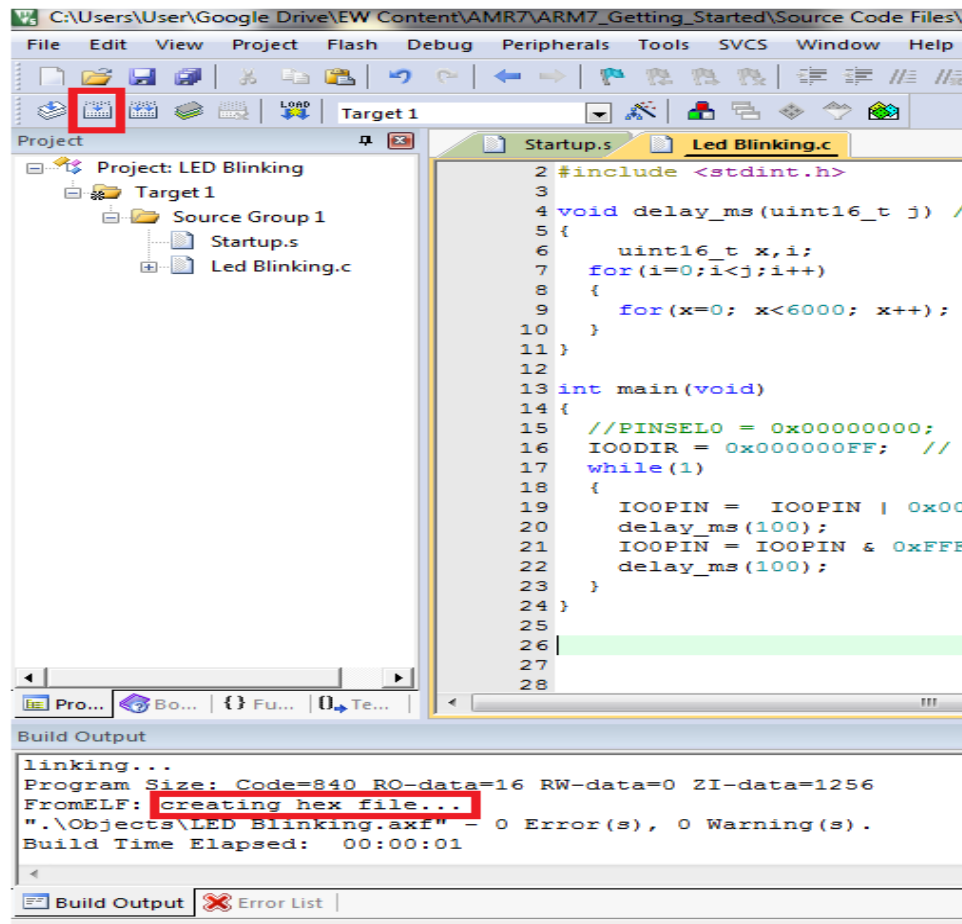




In the options for target window, go to the Linker tab. Select the Use Memory Layout from Target Dialogue option.

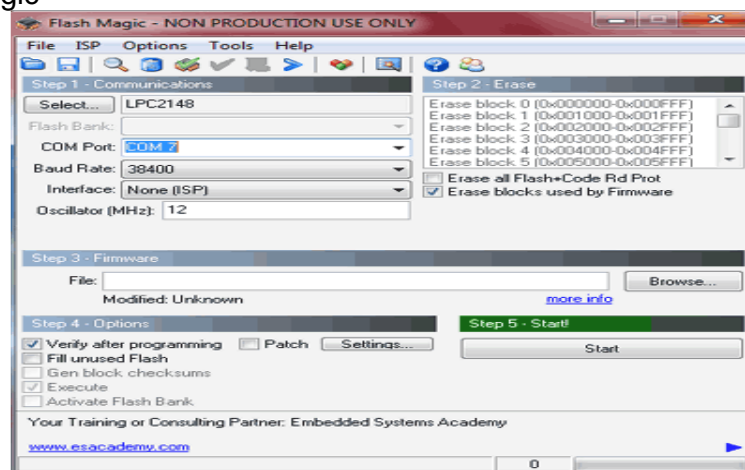


Then click on OK. 12. Now Type the source code for LED Blinking 13. Once the code is typed, Build the code by clicking on the button shown in red in the image below. You can also build the project from the Build Target option in the Project tab or by pressing F7 on the keyboard.



You can see creating hex file ... in the Build Output window as shown in the image.

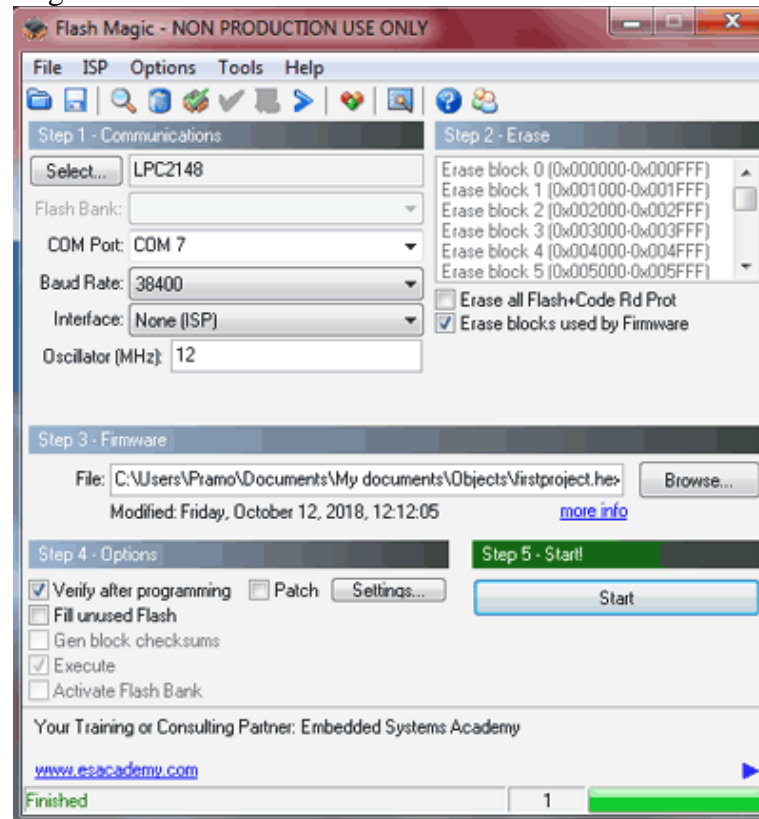
14. Once the project is built, a hex file is created in the Objects folder inside the folder of your project. Use **Flash Magic** software to burn this hex file in your microcontroller. So open Flash Magic



The Flash magic tool appears as above.

Below are the steps for flashing the ARM LPC2148:

- ☐ Select the LPC2148
- ☐ Give the COM port number according to Device Manager (COM1)
- ☐ Give baud rate as 9600 ☐ Oscillator as 12.000Mhz
- ☐ Tick mark the “Erase blocks used by firmware”
- ☐ Now select the hex file path
- ☐ Tick verify after programming checkbox.
- ☐ And click START After successfully burned the code , Finished (In green Colour) appears at the bottom as shown in image below



Now you can see the that LED starts blinking on the development board

**VIVA QUESTIONS:****1.Explain what is embedded system in a computer system?**

An embedded system is a special purpose computer system which is completely encapsulated by device it controls. It is a programmed hardware device in which the hardware chip is programmed with specific function. It is a combination of hardware and software.

**2.Mention what are the essential components of embedded system?**

Essential components of embedded system includes

- ☐ Hardware
- ☐ Processor
- ☐ Memory
- ☐ Timers
- ☐ I/O circuits
- ☐ System application specific circuits
- ☐ Software
- ☐ It ensures the availability of System Memory
- ☐ It checks the Processor Speed availability
- ☐ The need to limit power lost when running the system continuously
- ☐ Real Time Operating System
- ☐ It runs a process as per scheduling and do the switching from one process to another

**3. What are the examples of embedded system?**

An example of embedded system includes ATMs, cell phones, printers, thermostats, calculators, and videogame consoles. Handheld computers or PDAs are also considered embedded devices because of the nature of their hardware design, even though they are more expandable in software terms.

**4.What is embedded c?**

The C standard doesn't care about embedded, but vendors of embedded systems usually provide standalone implementations with whatever amount of libraries they're willing to provide. C is a widely used general purpose high level programming language mainly intended for system programming.

**5, What is main difference between C and embedded C?**

As, embedded C is generally an extension of the C language, they are more or less similar. However, some differences do exist, such as: C is generally used for desktop computers, while embedded C is for microcontroller based applications. C can use the resources of a desktop PC like memory, OS, etc.

**6.What is a Watchdog Timer?**

A watchdog timer is an electronic device or electronic cards that execute specific operation after certain time period if something goes wrong with an electronic system.

**7.What is the need for an infinite loop in embedded systems?**

Embedded systems require infinite loops for repeatedly processing or monitoring the state of the program. For instance, the case of a program state continuously being verified for any exceptional errors that might just happen during run-time such as memory outage or divide by zero, etc.

**8.What is semaphore?**

A semaphore is an abstract data type or variable that is used for controlling access, by multiple processes to a common resource in a concurrent system such as multiprogramming operating system. Semaphores are commonly used for two purposes

- ☐ To share a common memory space
- ☐ To share access to files

**9.What is the ARM7TDMI?**

The ARM7TDMI is a member of the Advanced RISC Machines (ARM) family of general purpose 32-bit microprocessors, which offer high performance for very low power consumption and price.

**10.Mention the features of ARM**

32-bit RISC-processor core (32-bit instructions)

- ,• 37 pieces of 32-bit integer registers (16 available)
- ,• Thumb instruction set,
- Pipelined (ARM7: 3 stages),
- Cached (depending on the implementation),
- Von Neuman-type bus structure (ARM7), Harvard (ARM9),
- Debug Interface,• Embedded ICE macrocell ,
- Jazelle DBX(Direct Bytecode eXecution).

**11.What is meant by Pipelining**

Pipelining is a technique that implements a form of parallelism called instruction-level parallelism within a single processor.

**12.List out the features in LPC2148 ARM microcontrollers (K1,CO2)**

32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package. 8 to 40 kB of on-chip static RAM and 32 to 512 kB of on-chip flash program memory. In-System/In-Application Programming (ISP/IAP) via on-chip boot-loader software. Single flash sector or full chip erase in 400 ms and programming of 256 bytes in 1 ms. Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip Real Monitor software and high speed tracing of instruction execution. USB 2.0 Full Speed compliant Device Controller with 2 kB of endpoint RAM. In addition, the LPC2146/8 provide 8 kB of on-chip RAM accessible to USB by DMA. One or two (LPC2141/2 vs. LPC2144/6/8) 10-bit A/D converters provide a total of 6/14 analog inputs, with conversion times as low as 2.44 ms per channel.

**13.Write the Application of GPIO in LPC2148 ARM Microcontrollers**

It is used to interface the digital input device like limit switch, digital sensors, keypad etc.,  
It is used to interface the digital output devices like LCD, buzzer and etc.,

**14.What is RTC**

The Real Time Clock (RTC) is a set of counters for measuring time when system power is on, and optionally when it is off.

**15.What is the Purpose of RTC**

Measures the passage of time to maintain a calendar and clock. Provides Seconds, Minutes, Hours, Day of Month, Month, Year, Day of Week, and Day of Year.

**16.What is the UART Communication?**

A universal asynchronous receiver/transmitter (UART) is a microchip that performs serial-to-parallel conversion of data received from peripheral devices and parallel-to-serial conversion of data coming from the CPU for transmission to peripheral devices

**17. Mention the feature of UART in LPC2148 ARM microcontrollers**

- ▶ 16 byte Receive and Transmit FIFOs
- ☐ Register locations conform to '550 industry standard.
- ☐ Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.
- ☐ Built-in fractional baud rate generator with auto bauding capabilities.

- ▶ Mechanism that enables software and hardware flow control implementation.

**18.Mention the feature of ADC in LPC2148 ARM microcontrollers (K1,CO3)**

10 bit successive approximation analog to digital converter Measurement range 0 V to VREF (typically 3 V; not to exceed VDDA voltage level).10 bit conversion time 2.44  $\mu$ s. Burst conversion mode for single or multiple inputs.

**19.Mention the feature of SPI interface**

- ☐ Synchronous, Serial, Full Duplex communication.
- ☐ Combined SPI master and slave.
- ☐ Maximum data bit rate of one eighth of the input clock rate.
- ☐ 8 to 16 bits per transfer

**20.Write the advantages of I2c bus in LPC2148 ARM microcontrollers (K1,CO3).**

- The I2C (Inter-IC) bus is a bi-directional two-wire serial bus that provides a communication link between integrated circuits (ICs).
- ☐ Standard I2C compliant bus interfaces that may be configured as Master, Slave, or Master/Slave.
- ☐ Arbitration between simultaneously transmitting masters without corruption of serial data on the bus.
- ☐ Programmable clock to allow adjustment of I2C transfer rates.
- ☐ Bidirectional data transfer between masters and slaves.

**21.What is the RTOS?**

A real time operating system (RTOS) is an operating system that guarantees a certain capability within a specified time constraint.

**22.What is a Thread? What are the differences between process and thread?**

A thread is a single sequence stream within in a process. Because threads have some of the properties of processes, they are sometimes called lightweight processes. Threads are popular way to improve application through parallelism. For example, in a browser, multiple tabs can be different threads. MS word uses multiple threads, one thread to format the text, other thread to process inputs, etc. A thread has its own program counter (PC), a register set, and a stack space. Threads are not independent of one other like

processes as a result threads shares with other threads their code section, data section and OS resources like open files and signals.

**23.What are the different scheduling algorithms**

First-Come, First-Served (FCFS) Scheduling. Shortest-Job-Next (SJN) Scheduling. Priority Scheduling. Shortest Remaining Time. Round Robin (RR) Scheduling. Multiple-Level Queues Scheduling.

**24.What is deadlock?**

Deadlock is a situation when two or more processes wait for each other to finish and none of them ever finish. Consider an example when two trains are coming toward each other on same track and there is only one track, none of the trains can move once they are in front of each other. Similar situation occurs in operating systems when there are two or more processes hold some resources and wait for resources held by other(s).

**25.List out some popular Real Time Operating Systems(RTOS)**

μC/OS – II, POSIX, VxWorks, OSOpen, OS-9, pSOSystem, RTEMS, Linux/RT-Linux, Virtuoso, Windows CE, PalmOS, QNX Neutrino,

**26.What is meant by kernel?**

A kernel, executive or nucleus is the smallest portion of the operating system that provides for task scheduling, dispatching, and inter task communication.

**27. What is meant by task?**

A task is an independent thread of execution that can compete with other concurrent tasks for processor execution time.

**28.What is multitasking and its types?**

Multitasking to share CPU time between two or more tasks.

- ☐ Pre-emptive Multitasking
- ☐ Non-preemptive Multitasking

**29.What is meant by Context switching?**

Context switching is the process of saving and restoring sufficient information for a real-time task so that it can be resumed after being interrupted.



**30. List out the task states of RTOS**

Running , Ready, Block .

**31. What is Rate monotonic Algorithm?**

Given a set of periodic tasks and preemptive priority scheduling, then assigning priorities such that the tasks with shorter periods have higher priorities (rate-monotonic), yields an optimal scheduling algorithm.

**32. Mention the features of  $\mu$ c/os-II**

The Real-Time Kernel is a portable, ROMable, scalable, preemptive real-time, multitasking kernel for microprocessors and microcontrollers.

**33. List out the task management function in  $\mu$ c/os-II.**

The functions to create task, suspend and resume, and time setting and time retrieving functions

- OSTaskCreate()
- OSTaskSuspend()
- OSTaskResume()
- OSTimeSet()
- OSTimeGet()

**34. List out the time management function in  $\mu$ c/os-II**

OSTimeDly()

OSTimeDlyHMSM()

OSTimeDlyResume()