ST. ANNE'S COLLEGE OF ENGINEERING AND TECHNOLOGY Accredited by NAAC

(Approved by AICTE, New Delhi. Affiliated to Anna University, Chennai) ANGUCHETTYPALAYAM, PANRUTI – 607 106.



LABORATORY MANUAL

EC3491 EMBEDDED SYSTEMS AND IOT LAB MANUAL

OBSERVATION NOTE

(FOR III B.E ELECTRONICS AND COMMUNICATION ENGINEERING STUDENTS)

NAME :

REGISTER NO :

YEAR / SEMESTER :

PERIOD :

AS PER ANNA UNIVERSITY (CHENNAI) SYLLABUS

2021 REGULATION

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PREPARED BY

B.ARUN KUMAR AP/ECE

EC3491 EMBEDDED SYSTEMS AND IOT LAB

1

SYLLABUS

PRACTICAL EXERCISES:

Experiments using 8051

- 1. Programming Arithmetic and Logical Operations in 8051.
- 2. Generation of Square waveform using 8051.
- 3. Programming using on Chip ports in 8051.
- 4. Programming using Serial Ports in 8051.
- 5. Design of a Digital Clock using Timers/Counters in 8051.

Experiments using ARM

- 1. Interfacing ADC and DAC
- 2. Blinking of LEDs and LCD
- 3. Interfacing keyboard and Stepper Motor.

Miniprojects for IoT

- 1. Garbage Segregator and Bin Level Indicator
- 2. Colour based Product Sorting
- 3. Image Processing based Fire Detection
- 4. Vehicle Number Plate Detection
- 5. Smart Lock System

COURSE OUTCOMES:

- CO1: Explain the architecture and features of 8051.
- CO2: Develop a model of an embedded system.
- CO3: List the concepts of real time operating systems.
- CO4: Learn the architecture and protocols of IoT.
- CO5: Design an IoT based system for any application.

30 PERIODS

EXP NO:	BASIC ARITHMETIC AND LOGICAL OPERATIONS USING 8051
DATE:	A. 8 BIT ADDITION

AIM:

To write a program to add two 8-bit numbers using 8051 microcontrollers.

ALGORITHM:

- 1. Clear Program Status Word.
- 2. Select Register bank by giving proper values to RS1 & RS0 of PSW.
- 3. Load accumulator A with any desired 8-bit data.
- 4. Load the register R 0 with the second 8- bit data.
- 5. Add these two 8-bit numbers.
- 6. Store the result.
- 7. Stop the program.





PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	4100	CLR	С	C3	Clear CY Flag
	4101	MOV	A,#0A	74 0A	Get the data1 in Accumulator
	4103	ADDC	A,#10	34 10	Add the data1 with data 2
	4105	MOV	DPTR,#4500	90 45 00	Initialize the memory location
	4108	MOVX	@DPTR,A	F0	Store the result in memory location
L1	4109	SJMP	L1	80 FE	Stop the program

Address	Output
4500	1A(LSB)
4501	00(MSB)

RESULT:

Thus the 8051 Assembly Language Program for addition of two 8 bit numbers was executed.



B. 8 BIT SUBTRACTION

AIM:

To perform subtraction of two 8 bit data and store the result in memory.

ALGORITHM:

- 1. Clear the carry flag.
- 2. Initialize the register for borrow.
- 3. Get the first operand into the accumulator.
- 4. Subtract the second operand from the accumulator.
- 5. If a borrow results increment the carry register.
- 6. Store the result in memory.

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	4100	CLR	С	C3	Clear CY Flag
	4101	MOV	A,#0A	74 0A	Get the data1 in Accumulator
	4103	SUBB	A,#05	94 05	Subtract data2 from data1
	4105	MOV	DPTR,#4500	90 45 00	Initialize memory location
	4108	MOVX	@DPTR,A	F0	Store the difference in memory location
L1	4109	SJMP	L1	80 FE	Stop the program

Address	Output
4500	05

RESULT:

Thus the 8051 Assembly Language Program for subtraction of two 8 bit numbers was executed.

7



C. 8 BIT MULTIPLICATION

AIM:

To perform multiplication of two 8 bit data and store the result in memory.

ALGORITHM:

- 1. Get the multiplier in the accumulator.
- 2. Get the multiplicand in the B register.
- 3. Multiply A with B.

Store the product in memory **PROGRAM:**

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	4100	MOV	A,#05	74 05	Store data1 in accumulator
	4102	MOV	B,#03	75 F0 03	Store data2 in B register
	4105	MUL	AB	A4	Multiply both
	4106	MOV	DPTR,#4500	90 45 00	Initialize memory location
	4109	MOVX	@DPTR,A	F0	Store lower order result
	410A	INC	DPTR	A3	Go to next memory location
	410B	MOV	A,B	E5 F0	Store higher order result
	410D	MOVX	@DPTR,A	F0	
L1	410E	SJMP	L1	80 FE	Stop the program

Address	Output
4500	OF(LSB)
4501	00(MSB)

RESULT:

Thus the 8051Assembly Language Program for multiplication of two 8 bit numbers was executed.

9

FLOW CHART:



D. 8 BIT DIVISION

AIM:

To perform division of two 8 bit data and store the result in memory.

ALGORITHM:

- 1. Get the Dividend in the accumulator.
- 2. Get the Divisor in the B register.
- 3. Divide A by B.

Store the Quotient and Remainder in memory

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START:	4100	MOV	A,#15	74 15	Store data1 in accumulator
	4102	MOV	B,#03	75 F0 03	Store data2 in B register
	4105	DIV	AB	84	Divide
	4106	MOV	DPTR,#4500	90 45 00	Initialize memory location
	4109	MOVX	@DPTR,A	F0	Store remainder
	410A	INC	DPTR	A3	Go to next memory location
	410B	MOV	A,B	E5 F0	Store quotient
	410D	MOVX	@DPTR,A	F0	
L1	410E	SJMP	L1	80 FE	Stop the program
		1		1	

Input			Output	
Memory Location Data			Memory Location	Data
4500 (dividend)	0F		4502 (remainder)	05
4501 (divisor)	03		4503 (quotient)	00

RESULT:

Thus the 8051 8051Assembly Language Program for division of two 8 bit numbers was executed.

FLOW CHART:



D. MASKING BITS IN AN 8 – BIT NUMBER

AIM:

To write an assembly language program to mask bits o and 7 of an 8 – bit number and store the result in memory using 8051 microcontrollers.

APPARATUS REQUIRED:

8051 microcontroller kit

ALGORITHM:

Masking bits in a 8 bit number

- Start the process
- Get the two data values
- Get the second data
- Logically 'AND' the two data values.
- Initialize the memory value and store the result in memory.
- Start the process

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
START	4100	MOV	A,#87	74 87	
	4102	ANL	A,#7E	54 7E	
	4104	MOV	DPTR,#4500	90 45 00	
	4107	MOVX	@DPTR,A	F0	
L1	4108	SJMP	L1	80 FE	

Output					
Memory Location Data					
4500	06				

RESULT:

Thus the 8051assembly language program for masking bits was executed and verified

a) 1's and 2's complement



EXP NO:

DATE

SQUARE AND CUBE PROGRAM, FIND 2'S COMPLEMENT OF A NUMBER

AIM:-

To write an assembly language to perform arithmetic, logical and bit manipulation instruction using 8051.

ALOGRITHM:

a) 1's and 2's complement

- Get the value
- Get the complement value of data.
- Initialize the data pointer value as memory.
- Move the complemented value to memory of data pointer.
- Increment the value and memory.
- Store the result in memory.
- Stop the process.

a) 1's and 2's complement

PROGRAM:

Label	Address	Mnemonics		Hex code	Comments
		Opcode	Operand		
	4100	MOV	A, #02	74, 02	Get the initial value
	4102	CPL	А	F4	Complement the value
	4103	MOV	DPTR, # 4200	90, 42, 00	Initialize the memory
	4106	MOVX	@ DPTR, A	F0	Move the data to memory
	4107	INC	А	04	Increment Accumulator
	4108	INC	DPTR	A3	Increment the memory
	4109	MOVX	@ DPTR, A	F0	Move the value to memory
ECE:	410A	SJMP	ECE	80, FE	Continue the process.

1's and 2's complement

Output		
Memory Location	Data	
4200	FD (1's complement	
4201	FE(2'S Complement)	

Square of a number

Input		Output		
Memory Location	Data	Memory Location	Data	
4200	89	4201	51	
		4202	49	

b) SQUARE PROGRAM FOR 8051

\$MOD51 ORG 4100H MOV DPTR,#4200H MOVX A,@ DPTR MOV B,A MUL AB INC DPTR MOVX @ DPTR,A INC DPTR MOV A,B MOVX @ DPTR,A L:SJMP L

C). CUBE PROGRAM FOR 8051

\$MOD51 **ORG** 4100H MOV DPTR,#4200H MOVX A,@DPTR MOV B,A MOV R7,A MUL AB MOV R0,A MOV R1,B MOV A, R7 ANL A,#0FH MOV B,A MOV A, R0 MUL AB MOV R2,A MOV R3,B MOV A, R1 MOV B,A MOV A, R7 ANL A,#0FH MUL AB MOV R4,A MOV R5,B MOV A,R3 MOV B,R4 ADD A,B MOV R3,A

MOLLADO
MOV A,R0
MOV B,A
MOV A,R7
ANL A.#0F0H
SWAD A
MULAB
MOV R4,A
MOV R6,B
MOV A,R7
ANI A #0F0H
SWAD A
SWAFA
MOV B,RI
MUL AB
MOV R0,A
MOV R1,B
MOV A.R6
MOV B R0
MOV KU,A
MOV R7,A
MOV A,R4
SWAD A
ANLA,#0F0H
MOV R6,A
MOV A,R0
MOV A,R0 SWAP A
MOV A,R0 SWAP A ANL A,#0F0H
MOV A,R0 SWAP A ANL A,#0F0H MOV R0 A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV R4,A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV R4,A MOV A,R1 SWAP A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV A,R1 SWAP A MOV B A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV A,R1 SWAP A MOV B,A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV A,R1 SWAP A MOV B,A MOV B,A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV A,R1 SWAP A MOV B,A MOV B,A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV A,R1 SWAP A MOV B,A MOV B,A MOV A,R7 SWAP A ANL A,#0FH
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV A,R1 SWAP A MOV A,R1 SWAP A MOV B,A MOV A,R7 SWAP A ANL A,#0FH ADD A,B
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV A,R1 SWAP A MOV B,A MOV B,A MOV A,R7 SWAP A ANL A,#0FH ADD A,B MOV R0,A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV A,R1 SWAP A MOV B,A MOV B,A MOV A,R7 SWAP A ANL A,#0FH ADD A,B MOV R0,A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV A,R1 SWAP A MOV B,A MOV A,R7 SWAP A ANL A,#0FH ADD A,B MOV R0,A MOV A,R6 MOV B,R2
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV A,R1 SWAP A MOV A,R1 SWAP A MOV B,A MOV A,R7 SWAP A ANL A,#0FH ADD A,B MOV R0,A MOV A,R6 MOV B,R2 ADD A,B
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV A,R1 SWAP A MOV B,A MOV A,R7 SWAP A ANL A,#0FH ADD A,B MOV R0,A MOV A,R6 MOV B,R2 ADD A,B MOV R6,A
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV A,R1 SWAP A MOV B,A MOV A,R7 SWAP A ANL A,#0FH ADD A,B MOV R0,A MOV A,R6 MOV B,R2 ADD A,B MOV R6,A MOV A,R3
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV A,R1 SWAP A MOV A,R1 SWAP A MOV A,R7 SWAP A ANL A,#0FH ADD A,B MOV R0,A MOV A,R6 MOV B,R2 ADD A,B MOV R6,A MOV A,R3 MOV B,R4
MOV A,R0 SWAP A ANL A,#0F0H MOV R0,A MOV A,R4 SWAP A ANL A,#0FH MOV R4,A MOV B,R0 ADD A,B MOV R4,A MOV A,R1 SWAP A MOV A,R1 SWAP A MOV B,A MOV A,R7 SWAP A ANL A,#0FH ADD A,B MOV R0,A MOV A,R6 MOV A,R6 MOV B,R2 ADD A,B MOV R6,A MOV A,R3 MOV B,R4 ADDC A,B

MOV A,R0 MOV B,R5 ADDC A,B MOV R0,A MOV DPTR,#4500H MOV A,R6 MOVX @DPTR,A INC DPTR MOV A,R3 MOVX @DPTR,A INC DPTR MOV A,R0 MOVX @DPTR,A L:SJMP L

Cube of a number

Input		Output	
Memory Location	Data	Memory Location	Data
4200	89	4500	56
		4501	3 C
		4502	27

RESULT:

Thus the assembly language program to find 2's complement, Square and cube of a number was executed and verified successfully using 8051 microcontrolle

EXP NO:	A/D INTERFACE WITH 8051
DATE	

AIM:

To write an assembly language program for interfacing of ADC with 8051.

ALGORITHM: -

- (i) Start the program
- (ii) Make CS=0 and send a low to high pulse to WR pin to start the conversion.
- (iii) Now keep checking the INTR pin. INTR will be 1 if conversion is not finished and INTR will be 0 if conversion is finished.
- (iv) If conversion is not finished (INTR=1), poll until it is finished.
- (v) If conversion is finished (INTR=0), go to the next step.
- (vi) Make CS=0 and send a high to low pulse to RD pin to read the data from the ADC
- (vii) Stop the program

PROCEDURE:

- (i) Place jumper J2 in C position
- (ii) Place jumper J5 in A position
- (iii) Enter and execute the program
- (iv) Vary the analog input (using trim pot) and view the corresponding digital value in LED display,

FLOWCHART:



PROGRAM:

Label	Address	Mnemonics	Comments
MAIN:	1000 1003 1005	MOV P1,#11111111B CLR P3.7 SETB P3.6 CLR P3.5 SETB P3.5	; initiates P1 as the input port ; makes CS=0 ;makes RD high ;makes WR low
WAIT:	1008 100A	JB P3.4, WAIT CLR P3.7 CLR P3.6 MOV A,P1 CPL A MOV P0,A SJMP MAIN END	;low to high pulse to WR for starting ;polls until INTR=0 conversion ;ensures CS=0 ;high to low pulse to RD for reading the data from ADC ;moves the digital data to accumulator ;complements the digital data
	100D		;outputs the data to P0 for the LEDs ;Jump to program
	100F		;End

Jumper Details:-



RESULT:

Thus, the assembly language program for performing the interfacing of ADC with 8051 has been verified.

EXP NO:	INTERFACING OF DAC
DATE	

AIM:

To interface the DAC with 8051 microcontroller and generate the square wave, saw tooth wave and triangular wave.

REQUIREMENTS:

S.No	Hardware & Software Requirements	Quantity
1	8051 Trainer Kit	1No
2	Power Chord	1No
3	DAC interfacing board	1 No
4	CRO	1 No

CIRCUIT DIAGRAM:



WAVEFORMS:



SQUARE WAVE

ADDRESS	LABEL	MNEMONICS	OPCODE	COMMENTS
4100		MOV DPTR, # E0C0	90,FF,C8	Move the immediate
4103	START	MOV A, # 00	74,00	Initialize the Accumulator
4105		MOVX @ DPTR, A	F0	To zero
4106		LCALL DELAY	12,41,12	Move the content of
4109		MOV A, # FF	74,FF	Accumulator to FF
410B		MOVX @ DPTR, A	F0	Long call delay
410C		LCALL DELAY	12,41,12	Move the 05 data
410F		LJMP START	02,41,03	To R register
4112	DELAY	MOV R1, # 05	79,05	Return to main program
4114	LOOP	MOV R2, # FF	74,FF	Short jump to start
4116	HERE	DJNZ R2, HERE	DA,FE	
4118		DJNZ R1, LOOP	D9,FA	
411A		RET	22	
411C		SZMP START	80,E3	

OBSERVATION:

AMPLITUDE	TIME PERIOD

DIGITAL CLOCK

AIM:-

To display the digital clock specifically by displaying the hours, minutes

and secondsusing 8051 kits

PROGRAM:

Address	Label field	Mnemonic field	Comments field
Main Pro	gram	ORG 8000H	Origin of the program from 8000H
8000		MOV DPTR,#2043H	; Load DPTR with Control port Address of 2 nd 8255
8003		MOV A,#80	; (A) = $80H$ = control word for all ports as output port
8005		MOV R7,#16H	; (R7) = $16H$ = to display 16^{th} year on years displays
8007		MOV DPTR,#2042H	; Load DPTR with port C Address of 2 nd 8255
800A		MOV A.R7	(A) = 16H
800B		MOVX @DPTR.A	: Display 16H at port C of 8255
800C		MOV R6.#00	$(R_{6}) = 00$
800E		ACALL Month	: Call month routine to display January
8010		MOV R5.#0	(R5) = 0 = first Day on Days displays
8012		ACALL Day	: Call the routine Day to display day at port A
8014		CINE R5.#1FH.Dav1	: Compare R5 with 1FH (i.e. 31_{10}), if R5 \neq 1FH then
			jump to Day1. No need of involving Accumulator in compare
8017		ACALL Month	: Call month routine to display February
8019		MOV R5.#0	(R5) = 0 = to display beginning of the Day
801B		ACALL Day	: Call the routine Day to display day at port A
801D		CINE R5.#1CH.Dav1	: Compare R5 with 1CH (i.e. 28_{10}), if R5 \neq 1CH then
0012			jump to Day1. In February month there are 28 days
8020		ACALL Month	: Call month routine to display March
8022		MOV R5.#0	(R5) = 0 = first Day on Days displays
8024		ACALL Day	: Call the routine Day to display day at port A
8026		CINE R5 #1FH Dav1	: Compare R5 with 1FH (i.e. 31_{10}), if R5 \neq 1FH then
0020		Conte Ro, In In, Duy I	iump to Dav1. No need of involving Accumulator in
			compare
8029		ACALL Month	: Call month routine to display April
802B	BACK	MOV R5 #0	(R5) = 0 = first Day on Days displays
802D	Differi	ACALL Day	: Call the routine Day to display day at port A
802F		CINE R5.#1EH.Dav1	: Compare R5 with 30_{10} if R5 \neq 1EH then jump to Day
8032		ACALL Month	: Call month routine to display May / July
8034		MOV R5 #0	(R5) = 0 = to display beginning of the Day
8036		ACALL Day	: Call the routine Day to display day at port A
8038		CINE R5 #1FH Dav1	: Compare R5 with 31_{10} if R5 \neq 1FH then jump to
			Day1.
803B		ACALL Month	; Call month routine to display June / August
803D		CJNE R6,#08,BACK	; Compare R6 with 07 (July), if R6≠07 then jump toBACK.
8040		MOV R5,#0	; $(R5) = 0 =$ to display beginning of the Day
8042		ACALL Day	; Call the routine Day to display day at port A
8044		CJNE R5,#1FH,Day1	; Compare R5 with 31_{10} , if R5 \neq 1FH then jump to Day1.
8047		ACALL Month	; Call month routine to display September
8049	BACK1:	MOV R5,#0	; $(R5) = 0 = $ first Day on Days displays
804B		ACALL Day	; Call the routine Day to display day at port A
804D		CJNE R5,#1EH,Day1	; Compare R5 with 1EH (i.e. 30_{10}), if R5 \neq 1EH then
			jump to Day1. No need of involving Accumulator in
			compare
8050		ACALL Month	; Call month routine to display October / December
8052		MOV R5,#0	; $(R5) = 0 =$ to display beginning of the Day
8054		ACALL Day	; Call the routine Day to display day at port A
8056		CJNE R5,#1FH,Day1	; Compare R5 with 31_{10} , if R5 \neq 1FH then jump to
			Day1.
8059		ACALL Month	; Call month routine to display November
805B		CJNE R6,#12,BACK1	; Compare R6 with 12H, if $R6 \neq 12H$ then jump to BACK1.
805E		INC R7	; Increment the year
805F		SJMP START	; Repeat the process again

Observation:

Input

r	
1200	00
1201	00
1202	00
1203	00
1204	00

Output:

Time is displayed in the RTC board as

	! Hour l Minutes			l seconds l		
Х	0	0	0	5	9	
Х	0	0	1	0	0	

RESULT:

Thus the digital clock program has been written and executed using 8086 microprocessorkit and the output of digital clock was displayed as [hours: minutes: seconds] successfully.

INTRODUCTION TO KEIL µ VISION 4 SOFTWARE

The μ Visio n4 IDE is a Windows-based software development platform that combines a robust editor, project manager, and makes facility. μ Visi on4 integrates all tools including the C compiler, macro assembler, linker/locator, and HEX file generator. μ Visio n4 helps expedite the development process of your embedded applications by providing the following:

- ✓ Full-featured source code editor
- ✓ Device database for configuring the development tool setting
- ✓ Project manager for creating and maintaining your projects
- ✓ Integrated make facility for assembling, compiling, and linking your embedded applications
- ✓ Dialogs for all development tool settings
- ✓ True integrated source-level Debugger with high-speed CPU and peripheral simulator
- ✓ Advanced GDI interface for software debugging in the target hardware and for connection to Keil ULINK
- ✓ Flash programming utility for downloading the application program into Flash ROM
- ✓ Links to development tools manuals, device datasheets & user's guides

The μ Vision4 IDE offers numerous features and advantages that help you quickly and successfully develop embedded applications. They are easy to use and are guaranteed to help you achieve your design goals.

The installation steps for keil software are given below:

- 1. Double click on Keil µvisio n4.exe file.
- 2. Then click on Next.
- 3. Tick the check box towards to license agreements and click **Next**.
- 4. Select Destination folder and click **Next**.
- 5. Fill the necessary text boxes and click **Next**.
- 6. Finally click on **Finish**.

Software Flow

First open the icon keil μ vis ion4 and the follow the steps are given below. The menu bar provides you with menus for editor operations, project maintenance, development tool option settings, program debugging, external tool control, window selection and manipulation, and on-line help. The toolbar buttons allow you to rapidly execute μ Vision4 commands. A Status Bar provides editor and debugger information. The various toolbars and the status bar can be enabled or disabled from the View Menu commands.

Creating a New Project

Below mentioned procedures will explains the steps required to setup a simple application and to generate HEX output.

STEP 1: Go to "Project" and close the current project "Close Project".



STEP 2: Go to the "Project" and click on "New µ vision Project"



STEP 3: A small window will pop up with name "**Create New Project**" and can be create and select destination path.



STEP 4: Create a folder and give a proper name that can be related to the Project.



STEP 5: A small window will pop up with name "**Select Device for Target 'Target 1**'", and select the data base NXP founded by Philips.



Hit File View 1	niaer Hater defina	Freemaly facts 325 Window 500	
11) 🗳 🖬 🥔 1/4	ng na a	· 克克克克 (安京市街 @]	• 12 # 10 • 12 A 🐨 🔦
Fraject	- T- + I - ▼ & X	14.82	
17-			
	_		
	_		
	_		
	_		
	_		
	_		
	_	U V1ston	
	_	3 Capyrelestruu A	tertup Loce to Alexan: I der and fide de rai-recent f
	_		
_			
Next go to "File	e" and click	"New".	
LED - µVision4	Derivat Flack Dahu	v Gannaraly Tack S//S Wincow Han	
E e Edg Wision4	<u>Project Fla</u> sh <u>D</u> eou Ctrl-H	g Fegipherals Ipols SVCS Window Heip	
E e Edit View	Project Flash Deou Ctrl+N Ctrl+D	g Fegipherals Iocs SVG Wincow Heip · · · · · · · · · · · · · · · · · · ·	- <u>] * Q o O & I</u> 3
E e Edg Vision4 Hex.	<u>Project Fljast D</u> eou Ctf+H Ctr+D	g Fegipherals Iocs SVCS Wincow Heip 章 「「「」」「」」 章 「「」」「」」 章 「「」」「」」 章 『 章 『 』 『 』 『 』 』 』 』 』 』 』 』 』 』 』 』 』	- 3 * 2 • 0 8 4 1 • 4
E e Edg (jew Hew.) Coen Ecse Sote	<u>Project Fljøst D</u> eou Ct1-N Ct1+D Ct1+D	g Fegipherals Toos SVCS Window Help 日本 「「「「」」」 「「」「」」 「」「」」 「」」 「」」 「」」	- 3 * 2 • • • • •
LED - pylision4 E e Edg Jiew Hew. Coen Dore Save Save ga.	<u>Project Flash D</u> eou Ctri-H Ctri+D Ctri+D	g Fegipherals Iocs SVCS Wincow Heip 中国 中国 中国 中国 王 王 王 王 王 王 王 王 王 王 王 王 王 王	
E e Edd (Jew Hew) Coen Dese Stytes, Stytes,	<u>Project Fljøst D</u> eou Ctf+H Ctr+D Ctr+D	g Fegipherals Iocs SVCS Wincow Heip 章 「「「」」 章 「「」」 章 「「」」 章 「」 章 『 章 『 章 『 』 』 』 『 』 『 』 『 』 『 』 『 』 『 』 『 』 『 』 『 』 『 』 『 』 『 』 『 』 『 』 『 』 『 』 』 』 』 』 』 』 『 』 』 』 』 』 』 』 』 』 』 』 』 』	
E e Edg verv Hewwww Coen Coen Coen Coen Coen Coen Coen Coen	Project Flash Deou Oth-N Crr+D Crr+S e.	g Fegipherals [pocs <u>S</u> YCS <u>W</u> ncow <u>H</u> ep	
LED - pVIsion4 Let Idt View Hew. Coen Coen Coen Save gal Save A Save A Device Databas Ucerse Marage	<u>Project Flash D</u> eou Ctri+N Ctri+O Ctri+O Ctri+S e.	g Fegipherals Jocs SVG Wincow Heip 章 唐 唐 唐 唐 道 章 家 書	
E E Edd - pVision4 E e Edd View Hew Coen Dose Save ga Save ga Save ga Coen Device Databas Ucense Manage	<u>Project Flash D</u> eou Ctri+N Ctri+O Ctri+O Ctri+S e.e. e.e.	g Fegipherals]ocs SvCS ∑ncow Hep ■ 2 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	
LED - pVision4 Let idn View Hew. Coen Dose Sore gal. Sore Al Device Databas Ucerse Marage Fjint Setup. Pint.	Project Flash Deou Ctri+N of Ctri+D Ctri+S Ctri+S e. e. e.	g Fegipherals [Jocs SYCS Wincow Heip * K & # *	
LED - pVIsion4 Let Sdt View New Coen	<u>Project Flash D</u> eou Ctri+N Ctri+D Ctri+D Ctri+S enerti-	g Fegipnerals Iocs SVG Wincow Heip 正言派 《	
E LED - pVision4 E e LED - pVision4 Per Set	Project Flash Deou Ctri+N of Ctri+C Ctri+C Ctri+C	g Fegipnerals Toos SVCS Wincow Hep * K & # # # // // /#	
E LED - pVIsion4 E e Idt View Hew. Coen Coen Coen Sove Ay Device Databas Vicerse Marage First Setup First Setup Coeffur Vices 1 Cheffur Vices	<u>Project Flash D</u> eou Ctri+N Ctri+D Ctri+D Ctri+S e. e. e. e. e. e. e. e. e. e. e. e. e.	g Fegipnerals Jocs SVG Wincow Heip E E Is Is V M * X M	
E E Edit Vision4 E Edit View Wex Coen Qoei Sove Sove Sove Sove A) Device Database Ucerse Marage Pint Setup Fint Project Egit Classical All	<u>Project Flash D</u> eou Ctri+N Ctri+Q Ctri+Q Ctri+S enert. Ctri+P	g Fegipnerals Iocs SVG Mncow Hep 正言派 《 通	
E tel - pVision4 E tel - side View Hew Coen Dose Sore gal. Sore Al Device Databas Ucerse Marage Fjint Setup. Fill, Prejes 1 Chiell Mea	Project Flash Deou Ctri+N of Ctri+D Ctri+D Ctri+S e. e. enents. Ctri+P	g Fegipnerals Jocs SVG Wincow Heip I I I I I I I I I I I I I I I I I I I	
LED - pVision4 Let Sdt View New., Coen	<u>Project Flash D</u> eou Ctri+N Ctri+Q Ctri+Q Ctri+S exercts ctri+P	g Fegipnerals Iocs SVG Muncow Hep 正常加加 。	

STEP 9: There are the main three windows are available in the keil IDE. First one is Project Workspace, second one is Editor Window and third one is Output Window.

District Teal Lister with	Teens infrauent face Acc. Flucan Tea	- R. C. Q. C.
1966 EEE EEE KOLEMONT - TAUAN	- 28 I 🕐 🕞	
rojant 🛶 🖬 W	(1.1 (Marc)	
+ 13 (2001)		
Project Window	-Editor Window	
hai 9: <mark>43</mark> 0: Ο Γ.ε [Π _α τι	1.1	
alls Octact		

STEP 10: Can be start to write *.asm/c code on the editor window.

19 11 III -	µVision4			
File Edit	t View Project Tadh Cori-N Diff-C Cori-N Cori-N Cori-N Cori-N Cori-N Cori-N Cori-N Cori-N	Debug *sishersis Tools SVCS Window * X & E E /* / * X & E Void deley(unsigned long to unsigned, inc s-Ex0000001; int wais.()	s leip 1910 1911	• 3 Ø & o o o @ 🗖 🔦
2000 2000 	e Datak see se tianagement set ap , Etti-P fleg ev fleg ev (e h Measure Se ial 1	TOTLIE - TXFFFFFFFF; TOTLIE - TXFFFFFFFF; TOTLIE - XFFFFFFFF; TOTLE - A; TOTLE - A; T	// Trrr 6 /* Trrir 14 /* Jurt1 16-33 E126 2* Jort1 16-33 E126 2* Jort1 16-33 E126 /* Jort1 16-33 E126 2* Jort1 16-23 E126 a=000000000000000000000000000000000000	83 on ourgent/c 1 ig f/ #/ #/ #/
		<pre></pre>	ต]เ	

STEP 11: Can be save the file, if the program is in "C" save as "filename.C" or else save as "filename.ASM".

이 (신) (분 (아이지) (SV)) cys:1	- 8 A -		
τ. • Α 8 □ Tarr≉*	<pre>c least*</pre>	VAL;; > // ⊃gr=1, 6-8; /- at a -	Απ «υκυκ/ +/ [] = [] +- [].
	er tie instru er tie	E - + + + E - E	* *

STEP 12: Add this source file to Group1, Go to the "**Project Workspace**" drag the +mark "**Target 1**" in that right click on "**Source Group1**" and click on "**Add Files to Group "Source Group1**".

1 10 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4	A. (77.77 /r 4/ 04) (10) (A. (77.74) (10) (A. (77.74) (10) (10) (10) (10) (10) (10) (10) (10
17 Ph	
i dia mandri il	energianen pund se (s
🚿 Aglion for Gray Course Group S Mis	77 (ñ.nstindní):
dimber -	
Control Line Pilot	
A Spreech Access to the	TTTFFFFF: /* Sinti id-15 as output-/
	2777 5500 P
Escuid a respectives	그가 이렇게 잘 못했던 그 것처럼 한 것을 것 같아. 것처럼 가지 않는 것을 수 있는 것을 것을 수 있는 것을 것을 것을 수 있는 것을 수 있다. 것을 것을 것을 것 같이 있는 것을 것 않은 것을 것 같이 않는 것을 수 있는 것을 것 같이 않는 것을 것 같이 없다. 것을 것 같이 않는 것 같이 없다. 것 같이 없는 것 같이 없는 것 같이 없는 것 같이 없다. 것 같이 않는 것 같이 없는 것 같이 없는 것 같이 없다. 것 같이 않는 것 같이 않는 것 같이 없는 것 같이 없다. 것 같이 않는 것 같이 없는 것 같이 않는 것 않는 것 같이 않는 것 않는 것 같이 않는 것 않는 것 않는 것 않는 것 않는 것 않는
Example and the first state of the state of	" .; /* 5:utl 10-23 Ziga: */
ignitize the	17
111 107 D 0	11 /* Sezil 14 25 Usin */
- tart	1 /* 5:112 10-23 200 */
🔍 Add I lies to Group 'Source Group 1' 🥠	x00000 #=_x00001;
Borne - Down - The end and the Files	1000
📕 Massage Contoporte de	
	(med long val)
Shaw (poude) re Dependence:	
N	A CARANA CARANA
18 4 8.	
1 X	
10000	

STEP 13: A small window will pop up with name "Add Files to Group Source Group1", by default, the Files of type will be in C source Files (*.C). If the program is in C, have to select C source Files (*.C) or select ASM Source file (*.s,*.src,*.a*).

Na 121 dt 681 \$27 (100.11	· A & =	
u	- Dime	
	<pre>1 Clarts b / point acts/ read (A Symbol merginshift) ansigned int pristored inty Vall; this status Tool IX - Introduced inty this status Tool IX - Introduced inty Sinte (I) Tool IX - Introduced inty Tool IX - Introduced inty Sinte (I) Tool IX - Introduced inty Tool IX - Inty</pre>	¹⁴ · son a fortre-Werner onstander ¹⁴
	POROS	

STEP 14: Then go to "**Project**" click on "**Build Target**" or **F7**. Output window will display related error and warning messages.

[] 🚅 (국 설) - 2014년 - 1888 (전 문 - 2014년		New Josie Pujet. New Multi Project Whitepare Open Project .	
1 200 Langet 1		Grade Project	
		gapun Linnarge Select Device for Target Target	• # ~~.): • * *
	1.5	themage them	/* Forth/ (d-85 as outputs
	150	Calculation for side in a specific and the second	A second seco
	1.0	Clear tair ef	the second
	0	Eu ld sarget)	41 LUCES 32 12 78107 50
	益	Bebard eff target files	CONTRACTO DUDAN ADDA (ADDA)
	42	Hattis Hitlers	At Bourse 20-20 Magin 47
	20	Translate Col Pr. 2 (Int CDs) F5 C Citri+C	
	1451	Stop W_Ud	p) a=0x0000001;
	12	1 U. LIX	
	12	7 Pokel', ABM/Example/(Measure), MeasureJuopm;	11111111111111111111111111111111111111
	1	D CokelAARDARDRIDETATISamoletaTeatticUV2	
		E Chican Altorit Examples (Hellor Hello, UV2)	
		E CHKOLARM BOUIDS SCINICB2130 BLinky Blinky CV2	
J. 146-2	6 1		
and the contraction			
udli naryan Dolamoling St Sompling EED	ч ты да С 	гунд 1' ар-Э 	
Simulation Part:

STEP 15: Go to "**Project**" menu, click on "**Rebuild all target Files**" and start **Debug**. From **View** menu can get **Memory Window** and from **Peripherals** can get I/O ports, Serial etc. For access internal memory type i:0x_memory location example: i:0x30 and for external and program memory x:0x_memory location, c:0x_memory location respectively. From **Register** window we can edit and access the values also.





HEX file creation:

STEP 16: Follow the **STEP** up to **14**, then go to "**Project**" and click on "**Option for Group 'Source Group1**". There a small window will open with name "**Option for Target 'Target1'**". In that window, go to first menu "Device", can be select LPC2148.



STEP 17: Next go to **Target** menu, set clock frequency as 12.0 MHz and select Thumb mode in the code generation selection box.

a Elener Deves deves deve	of name) and investigation.	un (oli Doblib), Calacci	
	of all New YY		
	See protest (20)	Carateria and an	
Opensiand during N	e v 🖃	Saturda he "pineta	
		T TOP CR T A DATE OF	
		🗁 Lite Erfletting Code Generalion	
-Karz aykeam	State point statement	-R st 201 (L) + ((2) K	
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	stat sta st to	the store state of the state	
		(b)	
- K K - "		-6FX	
- Jagan 11+1 1 ++ 40	and the second	ara. ji	400000000000000000000000000000000000000
C 47.0 2 11.121 /	<u>r 199-30 (r</u> 48	S 1 71-4 11-10 10 10 10	-
The second secon		1 14/2	
	100 CONTRACTOR 100 CONTRA	10. Y Y 10. Y 10. Y 10. Y	1

STEP 18: Then go to Output menu and click on create HEX file.

Jo Est Ser Extert I	"get Sebus Perscherele Itor 2003 Ender Usip	
🔁 🖾 🖓 🕼 🖉 🖓 🖓 🖓 🖓	いのないからたたたた 非正応に通	
(李山西)(二) 年	"ngel" - 📉 🧥 📩	
-up-	- L 🖉 🔁 LED.C 🖂 Statupe	
at Ball tan at 1	Options for Target 'Target 1'	
R ∰ Successi ΩS alt is F ∰utsic	Devices Taget (1997) Julies CrC++ Arrin Linker Debu; Urilies Suria: Fullis In Highers	
	 Ø fande Factor die UFD M Sobuginium Bon Ø ander die Sobuginium Bon Ø Sobuginium Bon Ø Sobuginium Bon Ø Gescharter, dieb UB 	Creste E st. t. 71-
En. CEv., IJ F		a comparations
FallCol a		
ent i land fier i sseudoling toacouple nelling ar prosent		
linking	OK Cencel Doleuta	lles

STEP 19: Then go to **Listing** menu and select C preprocessor Listing.

់ 🖬 ៉ៃ 🛃 🕹 🕹 🕹 🖬	이 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이 이	10 - A
	A BUR Salar	
Harce: 1 → Marino : + Marino : + Marino : + Marino :	aninan far Tangel (Tangel)' Genal Tanjel (Dag at Colom) sen [Tafael (S.)] ister[Tafael (S. Hillien)] Saka Friendlating: Saka State (Z.) Bage 206 in [Z.)	
	P /see r.&r. kiteg <*ts P UessRc:Rc:Rc:Rc:Rc:Rc:Rc:Rc:Rc:Rc:Rc:Rc:Rc:R	
	Policie and the second	
	Filmeritang MCDinap Filmeritang MCDinap	
<pre>N. @A _ HA _ I DUX: Lid tauget (Tanget entr) for Stations</pre>	FUlderagen	•
sting freezeeds: giling 132,574, drimp	IK Cence Delaux Helo	

STEP 20: Finally in the Linker menu, click on use memory layout from target dialog and click ok.

24 34 34 34 3	asi Benit ventaran intersita Erna Bal	The second second second second second
15-21-01-01-01-01		
E Baltoca .	Continue for Langel Lingues 1	8
	Sevies Target Output Listing Sevies Target Durang ILless Sevies Califier Name Name Name Name Name Image Maxee Target Social with the formation of the second secon	0000
	inda dz	
■ඇ. 後 近、 (9 c.) [Se dia Lafo ani da Mic : E a tak	
周47.1 1 (3 t) 2.1: 2.1: 2.1: 2.1: 2.1: 2.1: 2.1: 2.1:	Sic #1 a [LED > 3 	
同本、愛た。(3 t) 3.k 3.kpc: 3.k 3.kpc: 3.k.1.d topget Tarrow assembling Charbarts	Sie dia 	

STEP 21: For creating Hex file go to "Project" menu and click on "Rebuild all target Files"

\$ (d 🕷 👘	μ (2) soon Proyed	
1 Contraction	New Multi Prived Workspace.	
	Quin Injel.	
Target 1	Class Project	
ባባል 😜	Export	
1 Math	Managa	3
고 최대대	TOTO IS PROVIDE INSPECTO	
	Select Device for larget larget 21,	
1 10 10	Keinogellein	
- TRell	🗙 Options in Target Target P.,	規度
5 <u>7</u> 162	Osta juga	
E ARM	a Buildmant	() () () () ()
The C	Rebuild al tares: fles	
a Jooni	1 Denn	
∔ 3g os c 🧃	Congristate Church State State Control State Control (Control 2018) A CONTROL (CONTROL CONTROL CON	(1049)
- Bush	Stor light	
1 31 05 J/ Th 05 H	C C U Sund (SED/Durrely also 1027-48, 0020) 522-48, 0020 (PD, name)	
- TUSH	1.0", street iSEP Drawnlands Parting of mints 1.0.5 Bit and in ASM reasons distances in Section 2017. To see 120	Kolling of California D.C. Clarge Service
1 05.00	The weight of the second distribution of the sec	and a city of the weat of the standard
- Thosaty	C	

FLASH MAGIC

Introduction:

NXP Semiconductors produce a range of Microcontrollers that feature both on-chip Flash memory and the ability to be reprogrammed using In-System Programming technology. Flash Magic is Windows software from the Embedded Systems Academy that allows easy access to all the ISP features provided by the devices.

These features include:

- ✓ Erasing the Flash memory (individual blocks or the whole device)
- ✓ Programming the Flash memory
- ✓ Modifying the Boot Vector and Status Byte
- ✓ Reading Flash memory
- \checkmark Performing a blank check on a section of Flash memory
- ✓ Reading the signature bytes
- ✓ Reading and writing the security bits
- ✓ Direct load of a new baud rate (high speed communications)
- ✓ Sending commands to place device in Bootloader mode

Flash Magic provides a clear and simple user interface to these features and more as described in the following sections. Under Windows, only one application may have access the COM Port at any one time, preventing other applications from using the COM Port. Flash Magic only obtains access to the selected COM Port when ISP operations are being performed. This means that other applications that need to use the COM Port, such as debugging tools, may be used while Flash Magic is loaded. Note that in this manual third party Compilers are listed alphabetically. No preferences are indicated or implied.

Installation Procedure for Flash Magic software:

Installation steps are given below:

- 1. Double click on Flash Magic.exe.
- 2. Then click on Next.
- 3. Next accept the agreement and click on Next.
- 4. Select the destination folders and click on Next then Install.
- 5. Finally click on Finish.

Programming with communication port (COM1):

STEP 22: For programming with communication port, first select the device LPC2148 in ARM 7 category, COM port will be COM 1, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.

🐲 Hash Magic - HER DRIDOT LIDN TISE ONLY		🐖 Device Database	
Image: Second	Estan bis: 0 20000000000000000000000000000000000	ANN 7 TP 2011 UP2002 UP2002 UP2002 UP2005 UP2005	 Wa'l'T popicade: CURB ILLUK (C) prefucinist DAY picks: CARONOCCC -> LWLLL/FEF 5 groups: 32/3000000 (MACOFF2F) Trainistenced value 3-1" Cube Haad Mulection Locaton Cocccccific High speed can increator especialed
Sep 0-Lev Fie In the File Medical University	Dict	LPDSTD LPDSTD LPDSTD LPDSTD LPDSTD	
Sm4 Farm	Ster 5 Stock	💮 (P15155 🦣 (P1214)	
Veily after angrowing I Turking Tesh Jen yout chaptoon Second	5. . .	PT5145 - LPE2144 - PT2144 - PT2144 - PT2144 - PT214	
Technical on-line enlicies about 005 and XA programming www.enabademy.com//ag/docs	0	TPTC/IFE LPTC/IS4 LPT2/IS4 LPT2/IS4 LPT2/IS4 LPT2/IS4	
			Creed 1

STEP 23: Under the menu Options, go to Advanced options.



STEP 24: Under the menu Advanced options, go to Hardware configuration, click on the Use DTR and RTS to control RST and ISP pin.



🔆 Pauli Magu: - HON PRIMUS TON USE	our 🕘 🛛 🔣 Select	Um He	7
DI QIAVEN T	04		🚽 ឲ្យ 🕫 💷
Sections. 192 4	Ster for Million Statistic 🕷	3 👜	4 M m
LLM at LUH1	First we had a set of the burner of the burn	8.33.74 	
Network Nord (SEL	Liese a set a Wall 2011 - 2017 - 11 💌 🖉	3	
Conference of the	website node/ weigs	1948 	
Story Peekso	Brane Ville	Den ande	
No.1.d.Cr.ax.on	Role de	30	
Vice & Appendix The state of game vice	State State	179.12	
 Lan A., Skill Lan M., A of a box abox 25 		Herana Leo vas	🛛 🚺
Vertice Club Hard Trace provincible where	ter ordeler 👘 by S	anans Russian. extensions	X (1.144)
seasna mynasis, st∮na :	2		
6: After selecting ISP mode	on the Hardware Kit a	and click on start.	
Se Flash Magic	- NON PRODUCTION US	E ONLY	
File ISP Option	s Tools Help		
	1 an a 🗐 🗐 🖉	a 🙆 🙉	
Step 1 - Communi	cations	Step 2 - Erase	
Select Device	LPC2148	Etase block 0 (0x000000-0)	ONOFFE)
COM Port	СОМ 1	Erase Block 1 (0x001000-0)	001FFF)
Paud Pater	0000	Erase block 3 (0x003000-0x	003FFF)
bau nae.		Erase block 4 (0x004000-0x Erase block 5 (0x005000-0x	004FFF) 005FFF1
Interrace:		Erase all Flash+Code Rd	Prot
Oscillator (MHz):	12.0	Erase blocks used by He	k File
Step 3 - Hex File			
Step 3 - Hex File Hex File: D:\LPC	_2148\LED\LED.hex		Browse
Step 3 - Hex File Hex File: D:\LPC Modified	_2148\LED\LED.hex I: Thursday, August 25, 2011, 2	2:51:11 AM <u>more info</u>	Browse
Step 3 - Hex File Hex File: D:\LPC Modified	_2148\LED\LED.hex I: Thursday, August 25, 2011, 2	2:51:11 AM <u>more info</u> Step 5 - Start	Browse
Step 3 - Hex File Hex File: D:\LPC Modified Step 4 - Options	_2148\LED\LED.hex ! Thursday, August 25, 2011, 3	2:51:11 AM more info	Browse
Step 3 - Hex File Hex File: D:\LPC Modified Step 4 - Options Verify after prog	_2148\LED\LED.hex I: Thursday, August 25, 2011, 3 ramming	2:51:11 AM <u>more info</u> Step 5 - Start Star	Browse
Step 3 - Hex File Hex File: D:\LPC Modified Step 4 - Options Verify after prog Fill unused Flast Gen block chec	_2148\LED\LED.hex I: Thursday, August 25, 2011, 3 ramming 1 iksums	2:51:11 AM more info Step 5 - Start Star	Browse
Step 3 - Hex File Hex File: D:\LPC Modified Step 4 - Options Verify after prog Fill unused Flast Gen block chec Execute	_2148\LED\LED.hex I: Thursday, August 25, 2011, 3 ramming 1 iksums	2:51:11 AM more info	Browse
Step 3 - Hex File Hex File: D:\LPC Modified Step 4 - Options Verify after prog Fill unused Flast Gen block chec Execute Rotating, fully cust	_2148\LED\LED.hex I: Thursday, August 25, 2011, 3 ramming n ikcums omizable, remotely updated Int	2:51:11 AM <u>more info</u> Step 5 - Start Start Start	Browse
Step 3 - Hex File Hex File: D:\LPC Modified Step 4 - Options Verify after prog Fill unused Flast Gen block chec Execute Rotating, fully cust application!	_2148\LED\LED.hex I: Thursday, August 25, 2011, 3 ramming n iksums omizable, remotely updated Int	2:51:11 AM more info	Browse
Step 3 - Hex File Hex File: D:\LPC Modified Step 4 - Options Verify after prog Fill unused Flast Gen block one Execute Rotating, fully cust application! www.embeddedhil	_2148\LED\LED.hex I: Thursday, August 25, 2011, 2 ramming h kcsums omizable, remotely updated Int	2:51:11 AM more info	Browse

STEP 27: After the above steps device will start to program.



STEP 28: Finally can be see the finished indication and Reset the device into running mode.

Step 1 - Communi	cations	Step 2 - Frase	
Seleci Device]	LPC2143	Erase block 0 (0x203)	100 0x000FFF(
COM Port	сом і 🔗	Energe block 1 (0x001) Energe block 2 (0x002)	100-0x002F-FI
Baud Rate	3600 💌	Erase block 3 (0x303) Erase block 4 (0x304)	000.0000000000000000000000000000000000
Interface	None (ISP) 😽 😽	Enaise block 5 (0x005)	000-05005FFF(
Decillator (MHz)	12.0	Erase blocks uped	ty Hex File
Mocified	t: Thursday, August 25, 2011,	2:51:11 AM	<u>zinlo</u>
Mocified	r: Frumday, August 25, 2011,	2:51:11.AM	3.0.00.2
		SUCTION OF THE OWNER OWNER OF THE OWNER OF THE OWNER OF THE OWNER OWNER OWNER OF THE OWNER OWNE	
Verry after prog	ramming h		Stat
Fill unused Flas			
Fill unused Flas Leon block chec Familie	skauma		

Programming through USB:

Installation Procedure for USB Cable Driver:

The installation steps are given below:

STEP 1: Connect the USB cable between ARM-7 LPC2148 Trainer Kit and PC. Connect the power supply to the trainer kit and power up. After can see a popup window with name "Found New Hardware CP2102 USB to UART Bride Controller".



STEP 2: Select on Install from a list or specific location (Advanced) and click "Next".



STEP 3: Browse for the driver file location and select the folder, click next.



pund New I Please wa	lardware Wizard it while the wizard installs the software
Z	Silicon Labs CP210x USB to UART Bridge
	WdfCoinstaller01005.dll To C:\WINDOWS\system32
	Cancel

STEP 4: After that can see New Hardware wizard, and then click on finish.



STEP 5: Then right click on My Computer and select manage.



STEP 6: Then go to device manager, in that we have to select ports (COM and LPT). There we have to find the COM port which has been selected for USB cable. The same COM has to be select for Flash Magic.

Exampletar Wanngement	📳 🕾 🔯 🐼 Flash Magic - NON PRODUCTION USF ONLY
Computer Vernauentent Computer Vernauententent Computer Vernauententententententententententententente	Verland Market - NON PRODUCTION USE ONLY Verland Market - NON PRODUCTION USE ONLY Verland Market - NON PRODUCTION USE ONLY Verland - Verlan
Sorrado and Spokutors Sorrado and Spokutors	Loci + Lither control DELIGICAL Provide Subsection Violand Theodor Control C

STEP 7: Repeat the **STEP** from 22 to 28

ARM 7 LPC2148 TRAINER KIT

Introduction:

Thank you for using ARM 7 LPC2148 Trainer Kit designed for educational training purpose and the embedded device from NXP. This document is a User's guide which describes the complete hardware design of the ARM 7 LPC2148 Trainer Kit.



- 1. The device LPC2148 from NXP.
- 2. 12.00 MHz Crystal Oscillator Device speed.
- 3. 32.768 KHz crystal oscillator for RTC clock.
- 4. VREF jumper setting for ADC external Voltage reference.
- 5. 5V Red LED indication.
- 6. 3V3 Green LED indication.
- 7. Battery for RTC backup running.
- 8. Power ON/OFF switch.
- 9. 9VDC input Power Jack.
- 10. Input switch for external interrupt.

Optional: can be use for ISP manual Programming Mode.

- 11. Input switch for Reset the device.
- 12. ART1 connectivity.

13. UART0 connectivity.

Optional: Can be use for ISP programming mode.

- 14. LEDs for PWM output.
- 15. PWM enable switch.
- 16. USB connector for ISP programming and Serial port connectivity.
- 17. LED indication for to confirm the USB connection has been established.
- 18. Switch selection for ISP/UART. The selection will be for ISP if the switch has been pressed else for UART.
- 19. Switch for to enable the ISP.
- 20. I/O port 10 pin FRC connector (FRC5).
- 21. I/O port 10 pin FRC connector (FRC4).
- 22. I/O port 10 pin FRC connector (FRC3).
- 23. I/O port 16 pin FRC connector (FRC2).
- 24. I/O port 10 pin FRC connector (FRC1).
- 25. LPC2148 Device Daughter card.
- 26. USB ISP programmer daughter card.

Note: For In System Programming (ISP), the switch which has been mentioned in the description line 18 and 19 has to be pressed.

Features:

- 1. Device daughter card, easy and flexible to upgrade the device.
- 2. Four 10 pin individual digital or analog I/O ports are available.
- 3. One16 pin digital I/O port is available.
- 4. Inbuilt LEDs are available for PWM output.
- 5. Inbuilt push to on switch for Reset.
- 6. Inbuilt push to on switch for External Interrupt.
- 7. USB ISP programmer inbuilt.
- 8. On board Serial to USB bridge is available

Device Daughter Board Details:

CN6		
PIN NO:	DESCRIPTION	
1	P0.21/PW M5/AD1.6/CAP1.3	
2	P0.22/AD1.7/CAP0.0/MAT0.0	
3	RTCX1	
4	P1.19/TRACEPKT3	
5	RTCX2	
6	VSS	
7	VDD A	
8	P1.18/TRACEPKT2	
9	P0.25/AD0.4/AOUT	
10	D+	
11	D-	
12	P1.17/TRACEPKT1	
13	P0.28/AD0.1/CAP0.2/MAT0.2	
14	P0.29/AD0.2/CAP0.3/MAT0.3	
15	P0.30/AD0.3/CAP0.0/EINT3	
16	P1.16/TRACEPKT0	

CN8		
PIN NO:	DESCRIPTION	
33	P0.8/TXD1/PWM4/AD1.1	
34	P0.9/RXD1/PWM6/EINT3	
35	P0.10/RTS1/CAP1.0/AD1.2	
36	P1.23/PIPESTAT2	
37	P0.11/CTS1/CAP1.1/SCL1	
38	P0.12/DSR1/MAT1.0/AD1.3	
39	P0.13/DTR1/MAT1.1/AD1.4	
40	P1.22/PIPESTAT1	
41	P0.14/DCD1/EINT1/SDA1	
42	VSS	
43	VDD	
44	P1.21/PIPESTAT0	
45	P0.15/RI1/EINT2/AD1.5	
46	P0.16/EINT0/MAT0.2/CAP0.2	
47	P0.17/CAP1.2/SCK1/MAT1.2	
48	P1.20/TRACESYNC	

CN7	
PIN NO:	DESCRIPTION
17	P0.31/UP_LED/CONNECT
18	VSS
19	P0.0/TXD0/PWM1
20	P1.31/TRST
21	P0.1/RXD0/PWM3/EINT0
22	P0.2/SCL0/CAP0.0
23	VDD
24	P1.26/RTCK
25	VSS
26	P0.3/SDA0/MAT0.0/EINT1
27	P0.4/SCK0/CAP0.1/AD0.6
28	P1.25/EXTIN0
29	P0.5/MISO0/MAT0.1/AD0.7
30	P0.6/MOSI0/CAP0.2/AD1.0
31	P0.7/SSEL0/PWM2/EINT2
32	P1.24/TRACECLK

CN5	
PIN NO:	DESCRIPTION
49	VBAT
50	VSS
51	VDD
52	P1.30/TMS
53	P0.18/CAP1.3/MISO1/MAT1.3
54	P0.19/CAP1.2/MOSI1/MAT1.2
55	P0.20/EINT3/SSEL1/MAT1.3
56	P1.29/TCK
57	RESET
58	P0.23/VBUS
59	VSSA
60	P1.28/TDI
61	XTAL2
62	XTAL1
63	VREF
64	P1.27/TDO

FRC PIN DETAILS:

FRC 1		
PIN NO:	DESCRIPTION	
1	P0.16	
2	P0.17	
3	P0.18	
4	P0.19	
5	P0.20	
6	P0.21	
7	P0.22	
8	P0.23	
9	+5V	
10	GND	
FRC 5		
PIN NO:	DESCRIPTION	
1	P0.25	
2	P0.28	

P0.29

P0.30

P0.31

NA

RS232 TX1

RS232 RX1

+5 V

GND

3

4

5

6

7 8

9

10

FRC 2	
PIN NO:	DESCRIPTION
1	P0.8
2	P0.9
3	P0.10
4	P0.11
5	P1.16
6	P1.17
7	P1.18
8	P1.19
9	P1.20
10	P1.21
11	P1.22
12	P1.23
13	+5V
14	NA
15	+3V3
	GN
16	D

FRC 3		
PIN NO:	DESCRIPTION	
1	P0.3(SDA0)	
2	P0.2(SCL0)	
3	P0.14(SDA1)	
4	P0.11(SCL1)	
5	P0.16 ENT1	
6	P0.15 ENT2	
7	RS232 TX0	
8	RS232 RX0	
9	+5V	
10	GND	
10	GND	

FRC 4	
PIN NO:	DESCRIPTION
1	P0.0
2	P0.1
3	P0.2
4	P0.3
5	P0.4
6	P0.5
7	P0.6
8	P0.7
9	+5V
	GN
10	D







INTERFACE CARD DETAILS:

LED MODULE	
PIN NO	DESCRIPTION
1	Led1
2	Led2
3	Led3
4	Led4
5	Led5
6	Led6
7	Led7
8	Led8
9	+5V
10	GND

MATRIX KEYPAD		
PIN NO	DESCRIPTION	
1	Row1	
2	Row2	
3	Row3	
4	Row4	
5	Column1	
6	Column2	
7	Column3	
8	Column4	
9	+5V	
10	GND	

STEPPER MOTOR		
PIN NO	DESCRIPTION	
1	Input1	
2	Input2	
3	Input3	
4	Input4	
5	NA	
6	NA	
7	NA	
8	NA	
9	+5V	
10	GND	

	EEPROM	ADC
PIN NO	DESCRIPTION	PIN N
1	SDA	1
2	SCL	2
3	NA	3
4	NA	4
5	NA	5
6	NA	6
7	NA	7
8	NA	8
9	+5V	9
10	GND	10

ADC / '	ADC / TEMPERATURE	
PIN NO	DESCRIPTION	
1	NA	
2	AD01	
3	NA	
4	NA	
5	NA	
6	NA	
7	NA	
8	NA	
9	+5V	
10	GND	

ZIGBEE		
PIN NO	DESCRIPTION	
1	NA	
2	NA	
3	NA	
4	NA	
5	NA	
6	NA	
7	RXD	
8	TXD	
9	+5V	
10	GND	

	LCD		DAC				
PIN NO	DESCRIPTION	PIN NO	DESCRIPTION				
1	NA	1	NA				
2	RS	2	NA				
3	R/W	3	NA				
4	EN	4	NA				
5	D0	5	A8				
6	D1	6	A7				
7	D2	7	A6				
8	D3	8	A5				
9	D4	9	A4				
10	D5	10	A3				
11	D6	11	A2				
12	D 7	12	A1				
13	+5V	13	+5V				
14	-5V	14	-5V				
15	+3V3	15	NA				
16	GND	16	GND				

TO	GGLE SWITCH		DC MOTOR	S	SEVEN SEGMENT		
		PIN					
PIN NO	DESCRIPTION	NO	DESCRIPTION	PIN N	O DESCRIPTION		
1	SW1	1	INPUT1	1	SHIFT CLK		
2	SW2	2	ENABLE	2	DATA IN		
3	SW3	3	NA	3	LATCH CLK		
4	SW4	4	INPUT2	4	NA		
5	SW5	5	NA	5	NA		
6	SW6	6	NA	6	NA		
7	SW7	7	NA	7	NA		
8	SW8	8	NA	8	NA		
9	+5V	9	+5V	9	+5V		
10	GND	10	GND	10	GND		

FRC CONNECTION DETAILS:

S.No	Experiment	FRC1	FRC2	FRC3	FRC4	FRC5
1	ADC		LCD			ADC (Change mode)
2	DAC		DAC			
3	LED				LED	
4	PWM				LED	ADC (Change mode)
5	RTC		LCD			
6	EPROM	Keypad	LCD	EPROM		
7	Interrupt	LED				
8	Stepper Motor	Stepper Motor				
9	Temperature Sensor		LCD			ADC (Change mode)
10	Zigbee Transmitter	Keypad	LCD	Zigb ee		
11	Zigbee Receiver	LED		Zigb ee		
12	Seven Segment Display	Seven Segment				
13	Switch with LED	Toogle Switch			LED	
14	DC Motor	Toogle Switch			DC Motor	
15	LCD		LCD			
16	Keypad	Keypad	LCD			

EXP NO:

DATE

LED & FLASHING OF LED'S

AIM:

To write and execute the program for LED & Flashing Led's with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor board	1
2	USB/FRC Connector	few
3	LED Module	1
4	Power Supply adaptor (5V, DC)	1
5	Keil & flash magic Software	1

PROCEDURE

- 1. Create a New project, Go to "Project" and close the current project "Close Project".
- 2. Next Go to the Project New µvision Project Create New Project Select Device for Target.
- 3. Select the data base NXP LPC2148.
- 4. Add Startup file and Next go to "File" and click "New".
- 5. Write a program on the editor window and save as "Main.c".
- 6. Add this source file to Group and click on "Build Target" or F7.
- 7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".

8. Open Flash magic and select the device LPC2148 in ARM 7 category, Choose the hardware connected COM port, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.

9. Next browse the path of hex file and select the file.

10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program

11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

LED INTERFACING:

CIRCUIT DIAGRAM:



St. Anne's CET PROGRAM:

ET3491 EMBEDDED AND IOT LAB

```
#include <lpc214x.h>
int i;
int main()
{
IODIR0=(1<<3);
while(1)
{
IOSET0=(1<<3);
for(i=0;i<120000;i++);
IOCLR0=(1<<3);
for(i=0;i<120000;i++);
}
}</pre>
```

FLASHING OF LED: CIRCUIT DIAGRAM:

L	Л			
62	XTAL*	P0.0/TxD0/PWM1	19	
	XTALZ	P0.1/FxD0/PWM3/EINT0	22	
3	RTXC1	P0.3/SDA0/MAT0_0/EINT1	25	
-	RTXCZ	PC.4/SCK0/CAPC 1/AD0.6	24	
57	DOT	P0.5/MISO0/VATC.1/AD0.7	30	
	Rai	P0.7/SSEL0/PWM2/EINT2	31	
I		P0.5/TxC1/PWM4/AD1.1	33	
I		P0.9/FxD1/PWM6/EINT3	35	
I		P0 11/CTS1/CAP1 1/SCL1	37	hard and and and and and
I		F0.12/DSR1/MAT1.0/AD1.3	33	(卒) ((卒) ((卒) ((本) ((本) ((本) ((本) ((本) (
I		P0.13/DTR1/MAT1.1/AD1.4	41	$\psi \psi \psi \psi \psi \psi \psi \psi \psi$
I		P0.14.DCD1/EINT1/SDA1	45	
I		t at two is channel that in the	148	
I		PD 16/EINTC/MATC 2/CAP0 2	40	<u>+</u>
I		PC.17/CAP1.2/SCK1/WAT1.2	53	•
I		P0.19/MAT1.2/MOSI1/CAP1.2	54	
I		P0.20/MAT .3/SSEL1/EINT3	55	
I		P).21/PWM5/AD1.6/CAP1.3	2	
I		P0.22/AD1 //CAP0.0/VA10.0 P0.23	58	
I		TRANSFORMATION AND A STATE		
I		PC.25/ADC.4/AOUT	10	
I		P0 27/AD0 0/CAP0, 1/VAT0, 1	11	
I		P0.28/AD0 1/CAP0.2/MAT0.2	13	
I		P0 29/AD0 2/CAP0.3/VAT0.3	15	
I		P0.30AD0.5/EIN1.3/CAP0.0	17	
I			15	
I		P1.16/THACEPKT0	12	
49	VBAT	P1.18/TRACEPKT2	B	
65	10.12	P1.19/TRACEPKT3	44	
7	VREF	P1.20TRACESYNC	44	
51	1/3	P1 22/PIPESTAT1	40	
23	V3	P1.23/PIPESTAT2	35	
4.9	1.3	P1.24/TRACECLK	23	
59	VSSA	P1.25/EXTINO P1.25/RTCK	24	
50	V33	P*.27/TDO	54	
25	VSS	P1.28/TDI	58	
18	VSS	PT29/TCK P130/TMS	52	
0	VSS	P1.31/TRST	20	
L .	PC2138			

PROGRAM:

TYPE-I:

```
#include <lpc214x.h>
int i;
int main()
{
IODIR0=0x000000FF;
while(1)
{
IOSET0=0x000000AA;
for(i=0;i<120000;i++);
IOCLR0=0x000000AA;
for(i=0;i<120000;i++);
}
}</pre>
```

```
TYPE-II:
#include <lpc214x.h>
int i,b;
int main()
{
IODIR0=0x00000FF;
while(1)
{
for(b=0;b<8;b++)
{
IOSET0=(1<<b);
for(i=0;i<120000;i++);
IOCLR0=(1<<b);
for(i=0;i<120000;i++);
}
}
}
```

| Prepared By Mr.B.ARUN KUMAR

www.stannescet.ac.in

TYPE-III

```
\#include <lpc214x.h>
int i;
int main()
{
IODIR0 = (1 << 0)|(1 << 1)|(1 << 2)|(1 << 3)|(1 << 4)|(1 << 5)|(1 << 6)|(1 << 7);
while(1)
{
IOSET0=(1<<0);
for(i=0;i<120000;i++);
IOCLR0 = (1 << 0);
for(i=0;i<120000;i++);
IOSET0 = (1 << 1);
for(i=0;i<120000;i++);
IOCLR0 = (1 << 1);
for(i=0;i<120000;i++);
IOSET0 = (1 < < 2);
for(i=0;i<120000;i++);
IOCLR0=(1<<2);
for(i=0;i<120000;i++);
IOSET0=(1<<3);
for(i=0;i<120000;i++);
IOCLR0=(1<<3);
for(i=0;i<120000;i++);
IOSET0 = (1 < <4);
for(i=0;i<120000;i++);
IOCLR0 = (1 < <4);
for(i=0;i<120000;i++);
IOSET0=(1<<5);
for(i=0;i<120000;i++);
IOCLR0=(1<<5);
for(i=0;i<120000;i++);
IOSET0=(1<<6);
for(i=0;i<120000;i++);
IOCLR0 = (1 < < 6);
for(i=0;i<120000;i++);
IOSET0=(1<<7);
for(i=0;i<120000;i++);
IOCLR0=(1<<7);
for(i=0;i<120000;i++);
}
}
```

St. Anne's CET FRONT AND BACK LED:

ET3491 EMBEDDED AND IOT LAB

```
\#include <lpc214x.h>
int b,i;
int main()
{
IODIR0=0x00000FF;
while(1)
{
for(b=0;b<8;b++)
{
IOSET0=(1<<b);
for(i=0;i<120000;i++);
IOCLR0=(1<<b);
for(i=0;i<120000;i++);
}
for(b=7;b>=0;b--)
{
IOSET0=(1<<b);
for(i=0;i<120000;i++);
IOCLR0=(1<<b);
for(i=0;i<120000;i++);
}
}
}
```

ET3491 EMBEDDED AND IOT LAB

St. Anne's CET <u>SWITCH CONTROLLED LED (1-SWITCH):</u> <u>CIRCUIT DIAGRAM:</u>

<u>82</u> <u>01</u>	ATAL1 XTAL2	P0.0/Tx30/PWM1 P0 =/RxD0/PWM3/E/NT0	38 21 22
. 5	CRAMER C	P0.2/SCL0/CAP0.0	26
5	PTYCE	POJISDADIMATO JOEINI 1	#27
200	(a) when	PO 5NESOCIMATO 1/ADO 7	#29
2/4	RST	PE E/MOSIO/CAPO 2/AD1 0	(3)
		P0.7/S9EL0/PWM2/EINT2	
	I	P3.8/TxD1/PWM4/AD1 1	
	I	F0.9/RxD1/PWM6/EINT3	
	I	PU.10/HTS1/GAP1.0/AD1 2	$ = (x)^{-}(x)^$
	I	PUTING IS DURPT. PSGL	$\square + + + + + + + +$
		P0.15/DTR1/MAT1.1/AD1.4	
	I .	P0.14/DCD1/EINT1/SDA1	41
	1	PD.15/R01/EINT2/AD1.5	<u>A</u>
	1		•46 • • •
	1	P0 10/EINTO/MAT0 2/CAP0 2	#47 O O
	I	PD.17/CAP12/SCK1/MAT12	•53
	I .	PO IONAP SMOSTICAPT 2	#54
	I .	P0.20/MAT1.3/SSEL1/EINT3	eq.
	I .	P0.21/PWM5/AD1 9/CAP1 3	
	I .	P0.22/AD1 7/CAP0.0/MAT0 0	
		P0.23	
		P0 25/AD0 4/AOUT	9
	I .	PUL/EALUS	<u>10</u>
	I .	P0.27/AD0 0/CAP0.1/MAT0 1	
	I .	P0.28/AD0 1/CAP0.2/MAT0 2	#14
		P0.28AD0.2/CAP0.3/MAT0.3	1 16
		P0.30 ADD. Selen 3CAPO 0 P0.31	<u>*17</u>
		P1:16/TRACEPKT0	=16
451		P1.17/TRACEPICT*	B B
	VBAT	F1.18TRACEPKT2	=4
63	VOCT	DI SOTDATERVIS	#4ð
7	VaA	P121/PESTATO	#44
51	- Va	P1.22PIPESTAT1	*40
43	V3	P1.23/PIPESTAT2	-30
-A	V3	P1.24/TRACECLK	100 m 00
54	Vestoo	P1.25/EXTINO	=24
50	Vee	P126/RTCK	#64
42	Vag	Pt 96/TO	=6)
25	VSS	P1.25/TCK	*58
18	VSS	P1 30/TMS	52
- e:	V33	P1.31/TRST	
	LPC2138	1	

St. Anne's CET PROGRAM:

ET3491 EMBEDDED AND IOT LAB

```
\#include <lpc214x.h>
int i,b;
int main()
{
IODIR0=0x00000FF;
IODIR0=~(1<<16);
while(1)
{
if((IOPIN0&(1<<16))==0)
{
for(b=0;b<8;b++)
{
IOSET0=(1<<b);
for(i=0;i<120000;i++);
IOCLR0=(1<<b);
for(i=0;i<120000;i++);
}
}
else
{
IOCLR0=0x00000FF;
}
}
}
```

ET3491 EMBEDDED AND IOT LAB

St. Anne's CET SWITCH CONTROLLED LED (2-SWITCH): CIRCUIT DIAGRAM:



65

```
PROGRAM:
#include <lpc214x.h>
int i,b;
int main()
{
IODIR0=0x00000FF;
IODIR0=~(1<<16)&~(1<<23);
while(1)
{
if((IOPIN0&(1<<16))==0)
{
for(b=0;b<8;b++)
{
IOSET0=(1<<b);
for(i=0;i<120000;i++);
IOCLR0=(1<<b);
for(i=0;i<120000;i++);
}
}
else if((IOPIN0&(1<<23))==0)
{
for(b=7;b>=0;b--)
{
IOSET0=(1<<b);
for(i=0;i<120000;i++);
IOCLR0=(1<<b);
for(i=0;i<120000;i++);
}
}
else
{
IOCLR0=0x00000FF;
}
}
}
```

| Prepared By Mr.B.ARUN KUMAR

www.stannescet.ac.in

RESULT:

67

EXP NO:	INTERFACING OF LCD
DATE	

AIM:

To write and execute the program for LCD with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor board	1
2	USB/FRC Connector	few
3	LED Module	1
4	Power Supply adaptor (5V, DC)	1
5	Keil & flash magic Software	1

PROCEDURE

- 1. Create a New project, Go to "Project" and close the current project "Close Project".
- 2. Next Go to the Project New µvision Project Create New Project Select Device for Target.
- 3. Select the data base NXP LPC2148.
- 4. Add Startup file and Next go to "File" and click "New".
- 5. Write a program on the editor window and save as "Main.c".
- 6. Add this source file to Group and click on "Build Target" or F7.
- 7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".

8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.

9. Next browse the path of hex file and select the file.

10. After selecting ISP mode on the Hardware K it and click on start then device will start to program

11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

٦

INTERFACING LCD:

CIRCUIT DIAGRAM:

62 XTAL1	P0.0/TxD0/PWM1	ENGG COLLEGE
XTAL2	P0.1/RxD0/PWM3/EINT0	=21
3 BTXC1	P0.2/SCL0/CAP0.0 P0.3/SDA0/MAT0_0/EINT1	
5 RTXC2	P0.4/SCK0/CAP0.1/AD0.6	<u>■27</u> >>> ∞∞∞ 0000000
7.8	P0.5/MISO0/MAT0.1/AD0.7	
RST	P0.6/MOSI0/CAP0.2/AD1.0	131
	P0.7/SSEL0/PWM2/EIN12 P0.8/TyD1/PW/M4/AD1.1	=33
	P0.9/RxD1/PWM6/EINT3	<u>•34</u>
	P0.10/RTS1/CAP1.0/AD1.2	-35
	P0.11/CTS1/CAP1.1/SCL1	=3/
	P0.12/DSR1/MAT1.0/AD1.3	=39
	P0.13/DTR1/MAT1.1/AD1.4	#41
	P0.14/DCD1/EINT1/SDA1	#45
	THE PARTY PREPAREMENT OF DETER	
	P0.16/EINT0/MAT0.2/CAP0.2	#46
	P0.17/CAP1.2/SCK1/MAT1.2	153 153
	P0.18/CAP1.3/MISO1/MAT1.3	1954
	P0.19/MAT1.2/MOSI1/CAP1.2	#55
	P0.20/MAT1.3/SSEL1/EINT3	•
	PO 22(AD1 7/CAPO 0/MATO 0	<u>=2</u>
	P0.23	* 58
	DO DEIADO AIACUT	#9
	P0.23/AD0.4/AD01	=10
1	P0 27/AD0.0/CAP0.1/MAT0.1	att .
	P0.28/AD0.1/CAP0.2/MAT0.2	113 1144
	P0.29/AD0.2/CAP0.3/MAT0.3	#15
	P0.30/AD0.3/EINT3/CAP0.0	#17
	P0.31	
	P1.16/TRACEPKT0	10
9	P1.17/TRACEPKT1	#8
VBAT	P1.18/TRACEPKT2	#4
VREF	P1.20/TRACESYNC	* 48
V3A	P1.21/PIPESTATO	*44
V3	P1.22/PIPESTAT1	# 4U
23 \V3	P1.23/PIPESTAT2	32
V3	P1.24/TRACECLK	28
59 VSSA	P1.25/EXTINO	24
50 VSS	P1 27/TDO	#64
42 VSS	P1.28/TDI	# 60
VSS	P1.29/TCK	56
6 VSS	P1.30/TMS	=20
VSS	P1.31/TRST	

PROGRAM:

#include <lpc214x.h>
#include <lcd.h>

int main()

{

}

```
LCD_INIT();
LCDSTR(0x00000084,"St.ANNES");
LCDSTR(0x000000C2,"ENGG COLLEGE");
while(1)
{
}
```

LCD LAYOUT:

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
C0	C 1	C2	C3	C4	C5	C6	C 7	C8	C9	CA	CB	CC	CD	CE	CF

OUTPUT:

RESULT:

71

EXP NO:	INTERFACING OF MATRIX KEVROARD
DATE	

AIM:

To write and execute the program for Matrix Keyboard with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor board	1
2	USB/FRC Connector	few
3	LCD Module	1
4	Power Supply adaptor (5V, DC)	1
5	Keil & flash magic Software	1
6	Matrix Keypad Module	1

PROCEDURE

- 1. Create a New project, Go to "Project" and close the current project "Close Project".
- 2. Next Go to the Project New µvision Project Create New Project Select Device for Target.
- 3. Select the data base NXP LPC2148.
- 4. Add Startup file and Next go to "File" and click "New".
- 5. Write a program on the editor window and save as "Main.c".
- 6. Add this source file to Group and click on "Build Target" or F7.
- 7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
- 8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.
- 9. Next browse the path of hex file and select the file.
- 10. After selecting ISP mode on the Hardware K it and click on start then device will start to program

11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.
KEYBOARD INTERFACING:

CIRCUIT DIAGRAM:



| Prepared By Mr.B.ARUN KUMAR

www.stannescet.ac.in

PROGRAM:

```
#include <lpc214x.h>
#include <lcd.h>
#include <keyboard.h>
int main()
{
LCD_INIT();
LCDSTR(0x0000080, "Matrix Keypad");
 LCDSTR(0x000000C0, "Key Pressed: ");
while(1)
  {
      IOOCLR = CLR;
      IO0SET = C1;
      delay_ms(10);
      if(scan(R1))LCDSTR(0x000000CC, "0"); //K1
      if(scan(R2))LCDSTR(0x000000CC, "4"); //K5
      if(scan(R3))LCDSTR(0x000000CC, "8"); //K9
      if(scan(R4))LCDSTR(0x000000CC, "C"); //K13
      IOOCLR = CLR;
      IOOSET = C2;
      if(scan(R1))LCDSTR(0x000000CC,"1"); //K2
      if(scan(R2))LCDSTR(0x000000CC, "5"); //K6
      if(scan(R3))LCDSTR(0x000000CC, "9"); //K10
      if(scan(R4))LCDSTR(0x000000CC, "D"); //K14
      IOOCLR = CLR;
      IO0SET = C3;
      if(scan(R1))LCDSTR(0x000000CC,"2"); //K3
      if(scan(R2))LCDSTR(0x000000CC, "6"); //K7
      if(scan(R3))LCDSTR(0x000000CC, "A"); //K11
      if(scan(R4))LCDSTR(0x000000CC, "E"); //K15
      IOOCLR = CLR;
      IOOSET = C4;
      if(scan(R1))LCDSTR(0x000000CC, "3"); //K4
      if(scan(R2))LCDSTR(0x000000CC,"7"); //K8
      if(scan(R3))LCDSTR(0x000000CC, "B"); //K12
      if(scan(R4))LCDSTR(0x000000CC, "F"); //K16
    }
 }
```

RESULT:

| Prepared By Mr.B.ARUN KUMAR

www.stannescet.ac.in

75

EXP NO:	INTERFACING OF STEPPER MOTOR
DATE	

AIM:

To write and execute the program for Stepper Motor with ARM7 (LPC2148) processor.

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No	Hardware & Software Requirements	Quantity
1	ARM Processor board	1
2	USB/FRC Connector	few
3	Stepper Motor Module	1
4	Power Supply adaptor (5V, DC)	1
5	Keil & flash magic Software	1

PROCEDURE

- 1. Create a New project, Go to "Project" and close the current project "Close Project".
- 2. Next Go to the Project New µvision Project Create New Project Select Device for Target.
- 3. Select the data base NXP LPC2148.
- 4. Add Startup file and Next go to "File" and click "New".
- 5. Write a program on the editor window and save as "Main.c".
- 6. Add this source file to Group and click on "Build Target" or F7.
- 7. Create a Hex file from "Project" menu and click on "Rebuild all target Files".
- 8. Open Flash magic and select the device LPC2148 in ARM 7 category, COM port will be COM 3, baud rate 9600, interface None [ISP], Oscillator frequency 12.0 MHz and click on erase of flash code Rd plot.

9. Next browse the path of hex file and select the file.

10. After selecting ISP mode on the Hardware Kit and click on start then device will start to program

11. Finally can be see the finished indication and values in SPJ Terminal and Reset the device into running mode.

STEPPER MOTOR: (FOR WARD ROTATION) CIRCUIT DIAGRAM:



Pin No Clock	0.16	0.17	0.18	0.19

| Prepared By Mr.B.ARUN KUMAR

www.stannescet.ac.in

ET3491 EMBEDDED AND IOT LAB

St. Anne's CET PROGRAM:

```
\#include <lpc214x.h>
#include <delay.h>
int main()
{
IODIR0 = (1 << 16)|(1 << 17)|(1 << 18)|(1 << 19);
      while(1)
      {
       //FORWARD DIRECTION
      IOCLR0=(1<<16);
      IOCLR0=(1<<17);
      IOSET0=(1<<18);
      IOSET0=(1<<19);
      delay_ms(10);
      IOCLR0=(1<<16);
      IOSET0=(1<<17);
      IOSET0=(1<<18);
      IOCLR0=(1<<19);
      delay_ms(10);
      IOSET0=(1<<16);
      IOSET0=(1<<17);
      IOCLR0=(1<<18);
      IOCLR0=(1<<19);
      delay_ms(10);
      IOSET0=(1<<16);
      IOCLR0=(1<<17);
      IOCLR0=(1<<18);
      IOSET0=(1<<19);
      delay_ms(10);
      }
}
```

| Prepared By Mr.B.ARUN KUMAR

STEPPER MOTOR: (REVERSE ROTATION) CIRCUIT DIAGRAM:



Pin No Clock	0.16	0.17	0.18	0.19

| Prepared By Mr.B.ARUN KUMAR

PROGRAM:

}

```
\#include <lpc214x.h>
#include <delay.h>
int main()
{
IODIR0 = (1 << 16)|(1 << 17)|(1 << 18)|(1 << 19);
      while(1)
      {
      //REVERSE DIRECTION
      IOSET0=(1<<16);
      IOCLR0=(1<<17);
      IOCLR0=(1<<18);
      IOSET0=(1<<19);
      delay_ms(10);
      IOSET0=(1<<16);
      IOSET0=(1<<17);
      IOCLR0=(1<<18);
      IOCLR0=(1<<19);
      delay_ms(10);
      IOCLR0=(1<<16);
      IOSET0=(1<<17);
      IOSET0=(1<<18);
      IOCLR0=(1<<19);
      delay_ms(10);
      IOCLR0=(1<<16);
      IOCLR0=(1<<17);
      IOSET0=(1<<18);
      IOSET0=(1<<19);
      delay_ms(10);
      }
```

St. Anne's CET

RESULT:

Miniprojects for IOT:

- 1. Garbage Segregator and Bin Level Indicator
- 2. Colour based Product Sorting
- 3. Image Processing based Fire Detection
- 4. Vehicle Number Plate Detection
- 5. Smart Lock System

Objective:

To apply the knowledge, they gained in doing the experiments.

Team constitution:

A team size may be from 3 to 4 students.

Guidelines:

1.Students shall from a group and can do their mini project.

2.Student must buy their own hardware setup for doing Miniprojects.

3.If they are utilizing the college resource, they should get approval from HoD.

4.At the end, a report along with hardware must be submitted to college.

5.If required students need to present their work as presentation.

Garbage Segregator and Bin Level Indicator

IDEA:

With progress in human technology we have seen a substantial progress in the amount of waste generated. Recycling is the only way to manage this huge amount of waste. But recycling requires garbage to be segregated. Without segregation garbage cannot be recycled because different type of garbage requires different recycling processes.

Also It is important to educate users and instruct them every time they come near the dustbin about instructions about throwing the trash. For this purpose we design a garbage disposal system that uses multiple dustbins with a voice based system that speaks to the user each time he she stands before the dustbin.

The system makes use of a camera to detect presence if any person in front of the dustbin. If a person is detected, the system issues voice instructions to the user about throwing right garbage in the right bin. In case the dustbin is full it instructs the user to find another dustbin to throw garbage in.

To develop this system we make use of a raspberry Pi controller. The controller is interfaced with a camera and a voice speaker for detection and communication. The controller gets dustbin level input using ultrasonic level sensors each having LED indicators interfaced to it. The level sensors are used to constantly feed the raspberry pi with bin levels.

The raspberry pi is also interfaced with a Wifi module to transmit the level data over the internet. The Level sensor panels are made to be easily mounted over any dustbin. This allows the system to be easily screwed over any dustbin for instant installation.

The data is transmitted over IOT to IOT gecko platform which displays the bin level data over internet. This indication can be used to alert the authorities that the garbage bins need to be emptied. Thus the system automates garbage segregation and level monitoring to help counter the garbage crisis using IOT.

Note: The Dustbins are not included in this kit. The sensors can be mounted over any open dustbins.

Components

Raspberry Pi

Wifi Module

Ultrasonic Level Sensors

LED Indicators

Camera

Speaker

Wiring

Supporting Frame

Buttons & Switches

Screws & Bolts

Resistors

Capacitors

Diodes

IC's

Transistors

Connectors

PCB

Colour based Product Sorting

IDEA:

Color Based Object Sorting has a wide usage in fruit sorting as well as candy sorting industries. This system puts forward a mechanism to detect color and sort items through image processing. Once identified a mechanism is used to sort the candies into particular bins baskets. We here demonstrate this mechanism using a camera with electronic circuitry along with sorting mechanism using 3 bins. The system uses raspberry pi connected to a controller circuit to achieve this task. The controller circuit consists of a camera attached to it that detects color of a small object in front of it. A motor is used to feed an object to the camera chamber. As soon is the color is detected a signal is sent to the sorter mechanism which uses a motor to position the sorting tube towards respective section. A feeder is then used to push the object towards the tubs so that it gets sorted and next object is pulled in by the feeder. The action details are sent to the IOT server using iotgecko platform to keep track of the number of objects sorted in each section. Thus, we achieve a completely automated IOT based sorting system.

Hardware Specifications

Raspberry Pi 3

Camera

Servo Motor

LCD Display

Resistors

Capacitors

Transistors

Cables and Connectors

Diodes

PCB and Breadboards

LED

Transformer/Adapter

Push Buttons

Switch

IC

IC Sockets

Connector Shaft

Bed Frame

Tubes

Screws & Joints

Supporting Frame

Software Specifications

Programming Language:

Python

IOTGecko

BLOCK DIAGRAM:



Image Processing based Fire Detection

IDEA:

The main advantage of Image Processing Based Fire Detection System is the early warning benefit. This system can be installed just about anywhere in a commercial building, malls and at many more public places for fire detection. This system uses camera for detecting fires. So we do not need any other sensors to detect fire. System processes the camera input and then processor processes it to detect fires. The heat signatures and fire illumination patterns are detected in images to determine if it is a fire and take action accordingly. On detecting fire system goes into emergency mode and sounds an alarm. Also displays the status on the LCD display informing about the system.

Hardware Specifications

Raspberry Pi 3 Camera Buzzer LCD Display Resistors Capacitors Transistors Cables and Connectors Diodes PCB and Breadboards LED Transformer/Adapter Push Buttons Switch IC **IC** Sockets

Software Specifications

Linux

Programming Language: Python

BLOCK DIAGRAM:



Vehicle Number Plate Detection

IDEA:

OBJECTIVE

Project Code :TEMBMA3101

The main objective of this project is to design an efficient automatic authorized vehicle identification system by using the vehicle number plate.

ABSTRACT

The basic idea of this project is to build a **number plate recognition system using python.** Real-Time license plate detection and recognition can be very useful for automating toll booths, finding out traffic rule breakers, and for addressing other vehicle-related security and safety issues.

The system uses a camera interfaced to a Raspberry Pi. The system constantly processes incoming camera footage to detect any trace of number plates. On sensing a number plate in front of the camera, it processes the camera input, extracts the number plate part from the image. Processes the extracted image using OCR and extracts the number plate number from it. The system then displays the extracted number on the monitor.

Keywords: Raspberry Pi, Ultrasonic sensor, web camera, power supply

SPECIFICATIONS

Hardware components:

- Raspberry pi
- · Memory Card
- Ultrasonic sensor
- Web camera
- Monitor
- 5V 2A Adapter

Software requirements:

- Python3 IDLE
- NOOBS OS
- VNC Viewer
- · Advanced IP Scanner



SMART LOCKSYSTEM

IDEA:

From connected cars to connected wearables to home security, the Internet of Things is rapidly marking its presence in every field. Now we have IoT enabled home automation and security devices that can be controlled from anywhere in the world using the Internet of Things. There are many kinds of Wi-Fi door lock available in the market which makes your home more secure and saves time in finding the keys. Here we are also building a similar Wi-Fi door lock which can be controlled from the Smartphone

So in this project, we are going to make an **IOT based Door Lock System using NodeMCU**, Solenoid Lock, and Adafruit IO. Here NodeMCU will act as the main controller and connect the user to the door lock system using the Internet. This allows the user to lock/unlock his Home's door lock by using a smartphone from anywhere in the world.

Components Required

NodeMCU ESP8266

Solenoid Lock

Relay Module

Buzzer

