Assignment No 2

Submitted by:

Name: Mukund Vishwas Chavan

Course: Embedded System Design

Student ID: 01011

Student E-mail: 2024ht01011@wilp.bits-pilani.ac.in

Q.1. Displaying BITS ID, Voltage Difference, and Date on LCD Using LPC2378 Kit

1. Introduction

The objective of this assignment is to design and implement a C program that displays t he BITS ID(2024HT01011), voltage difference between the terminals of a potentiometer, and the date in DD/M format on an LCD display connected to the LPC2378 kit. The pote ntiometer is connected to the AD0.0 pin (P0.23) of the LPC2378.

2. Hardware and Software Setup

Components Used:

- LPC2378 kit
- Potentiometer
- LCD display

Connections:

- Potentiometer to AD0.0 pin (P0.23)
- LCD connections:
  - DB4: P1.24
  - DB5: P1.25
  - DB6: P1.26
  - DB7: P1.27
  - RS: P1.28
  - RW: P1.29
  - E: P1.31

Software Tools:

- Keil uVision4/5 for code development and debugging.

3. Code Explanation

Overview: The program initializes the ADC and LCD, reads the ADC values from the pot entiometer, converts these values to voltage, and displays the BITS ID, voltage value, an d date on the LCD.

Key Sections:

- **Initialization of ADC and LCD**:
  - ADC is configured to read values from the potentiometer connected to AD 0.0.
  - LCD is initialized to display text.
- **Reading and Converting ADC Values**:
  - The ADC value is read and converted to a voltage.
- **Displaying Values on the LCD**:
  - The BITS ID, voltage value, and date are formatted as strings and displaye d on the LCD.

Challenges and Solutions:

- **Challenge**: Synchronizing ADC readings with LCD updates.
- **Solution**: Introduced delays to ensure proper timing.

4. Debugging and Testing

Entering Debug Mode:

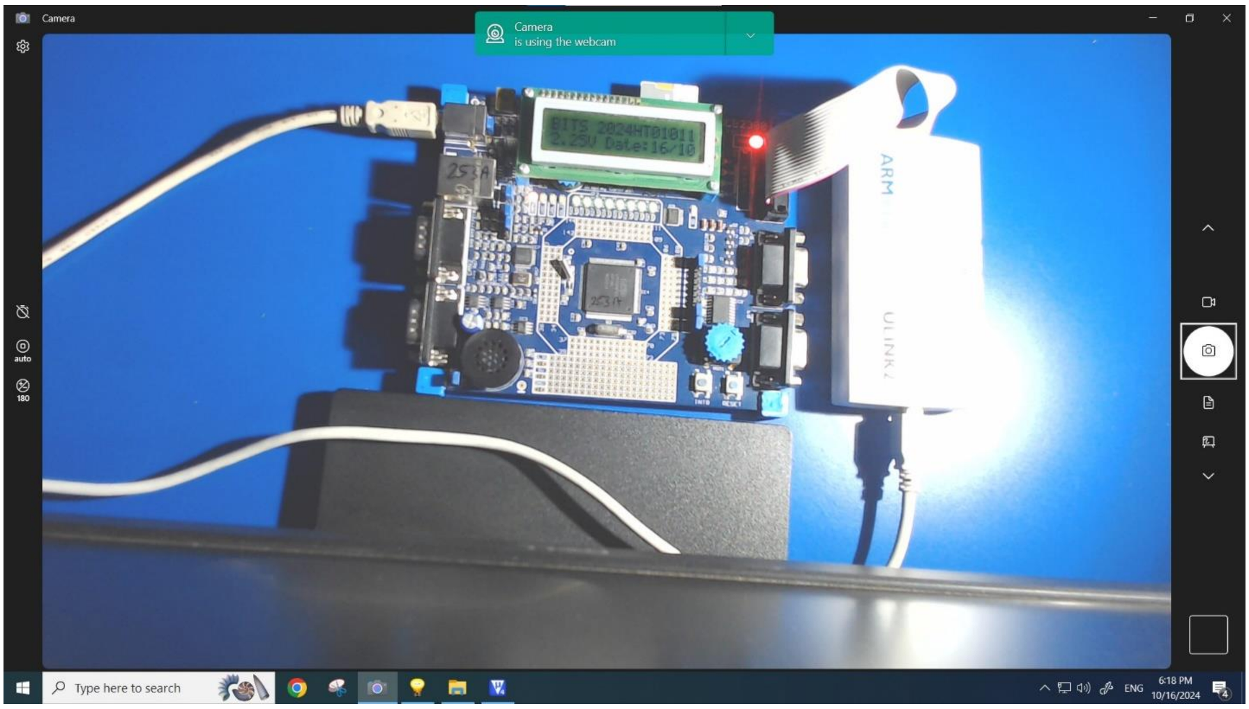
- In Keil, go to **Project** > **Options for Target** > **Debug** and select the appropriate de bugger.
- Click the **Debug** button or press Ctrl + F5 to enter debug mode.

Verifying Values:

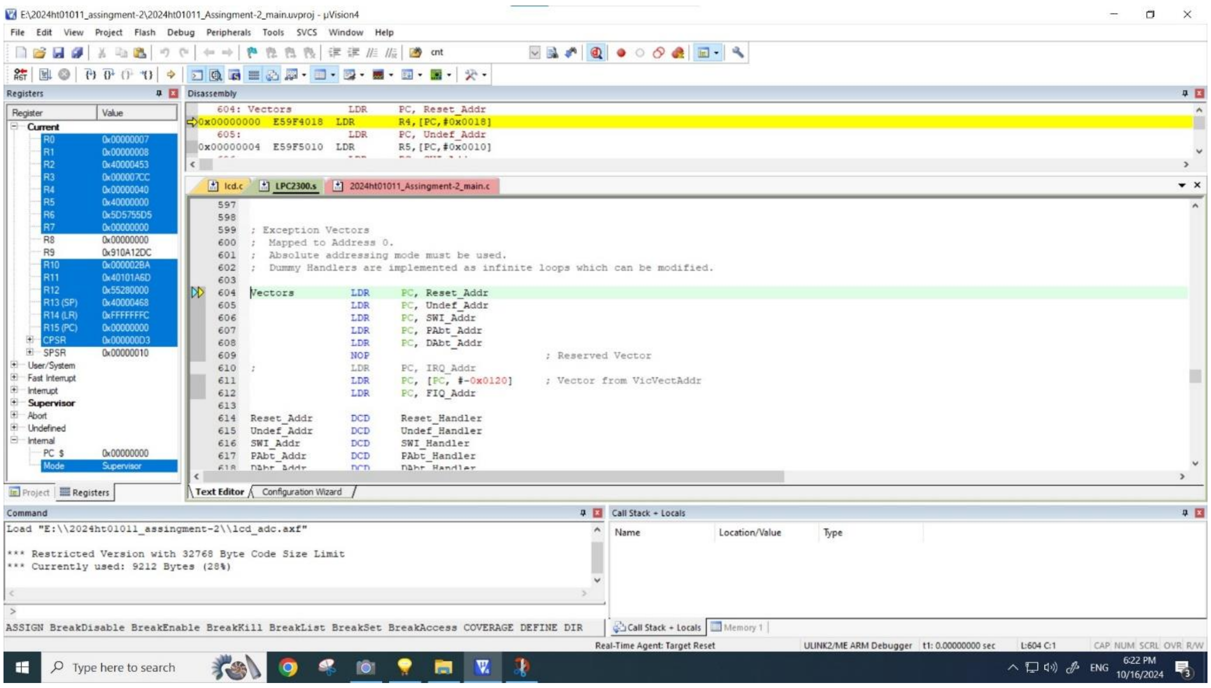
- Use the **Watch** and **Memory** windows to monitor ADC values and variables in rea l-time.
- Set breakpoints to pause execution and inspect variables at specific points in th e code.

5. Screenshots

- **LCD Output**: LCD displaying the BITS ID, voltage value, and date.



- **Keil IDE in Debug Mode**:


E:\2024ht01011_assingment-2\2024ht01011_Assingment-2_main.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
lcd.c
LPC2300.s
2024ht01011_Assingment-2_main.c
Project
Registers
Text Editor
Configuration Wizard
Command
Call Stack + Locals
Memory 1
Type here to search

Register	Value
Current	
R0	0x00000007
R1	0x00000008
R2	0x40000453
R3	0x000007CC
R4	0x00000040
R5	0x40000000
R6	0x5D5755D5
R7	0x00000000
R8	0x00000000
R9	0x910A12DC
R10	0x000002BA
R11	0x40101A6D
R12	0x55280000
R13 (SP)	0x40000468
R14 (LR)	0xFFFFFFFC
R15 (PC)	0x00000000
CPSR	0x000000D3
SPSR	0x00000010
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000000
Mode	Supervisor

```
   604: Vectors          LDR      PC, Reset_Addr
0x00000000  E59F4018  LDR      R4,[PC,#0x0018]
   605:                  LDR      PC, Undef_Addr
0x00000004  E59F5010  LDR      R5,[PC,#0x0010]
```

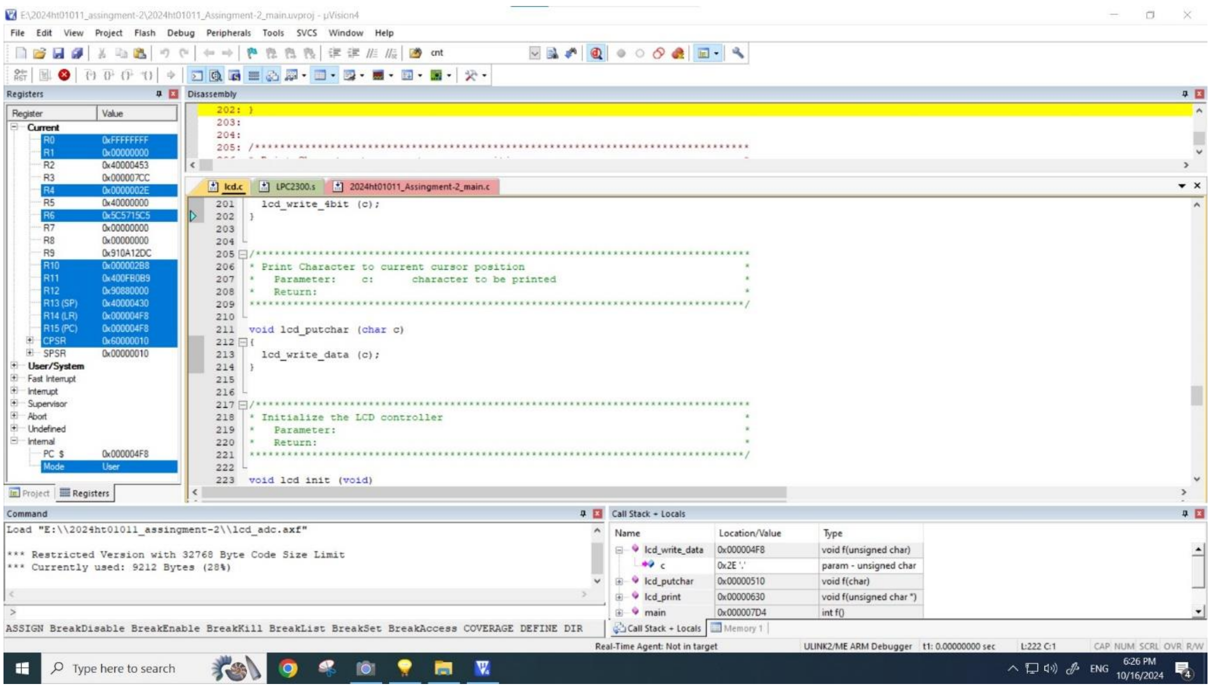
```


; Exception Vectors
;  Mapped to Address 0.
;  Absolute addressing mode must be used.
;  Dummy Handlers are implemented as infinite loops which can be modified.

Vectors         LDR     PC, Reset_Addr
                LDR     PC, Undef_Addr
                LDR     PC, SWI_Addr
                LDR     PC, PAbt_Addr
                LDR     PC, DAbt_Addr
                NOP                            ; Reserved Vector
;               LDR     PC, IRQ_Addr
                LDR     PC, [PC, #-0x0120]     ; Vector from VicVectAddr
                LDR     PC, FIQ_Addr

Reset_Addr      DCD     Reset_Handler
Undef_Addr      DCD     Undef_Handler
SWI_Addr        DCD     SWI_Handler
PAbt_Addr       DCD     PAbt_Handler
DAbt_Addr       DCD     DAbt_Handler
```

```
Load "E:\\2024ht01011_assingment-2\\lcd_adc.axf"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 9212 Bytes (28%)

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR
```

Name	Location/Value	Type

Real-Time Agent: Target Reset   ULINK2/ME ARM Debugger   t1: 0.00000000 sec   L:604 C:1   CAP NUM SCRL OVR R/W

ENG 6:22 PM 10/16/2024


E:\2024ht01011_assingment-2\2024ht01011_Assingment-2_main.uvproj - µVision4

Register	Value
Current	
R0	0xFFFFFFFF
R1	0x00000000
R2	0x40000453
R3	0x000007CC
R4	0x0000002E
R5	0x40000000
R6	0x5C5715C5
R7	0x00000000
R8	0x00000000
R9	0x910A12DC
R10	0x000002B8
R11	0x400FB0B9
R12	0x90880000
R13 (SP)	0x40000430
R14 (LR)	0x000004F8
R15 (PC)	0x000004F8
CPSR	0x60000010
SPSR	0x00000010
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x000004F8
Mode	User

Disassembly

```
202: }
203:
204:
205: /*****************************************************************************
```

lcd.c | LPC2300.s | 2024ht01011_Assingment-2_main.c

```c
  lcd_write_4bit (c);
}


/*****************************************************************************
* Print Character to current cursor position                                  *
*   Parameter:    c:      character to be printed                             *
*   Return:                                                                   *
*****************************************************************************/

void lcd_putchar (char c)
{
  lcd_write_data (c);
}


/*****************************************************************************
* Initialize the LCD controller                                               *
*   Parameter:                                                                *
*   Return:                                                                   *
*****************************************************************************/

void lcd init (void)
```

Command

```
Load "E:\\2024ht01011_assingment-2\\lcd_adc.axf"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 9212 Bytes (28%)
```

```
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR
```

Call Stack + Locals

Name	Location/Value	Type
lcd_write_data	0x000004F8	void f(unsigned char)
c	0x2E '.'	param - unsigned char
lcd_putchar	0x00000510	void f(char)
lcd_print	0x00000630	void f(unsigned char *)
main	0x000007D4	int f()

Real-Time Agent: Not in target   ULINK2/ME ARM Debugger   t1: 0.00000000 sec   L:222 C:1

6. Conclusion

Summary:

- Successfully implemented a C program to display BITS ID, voltage value, and dat e on the LCD using the LPC2378 kit.
- Debugged and tested the program using Keil uVision.

Learnings:

- Gained experience in configuring and utilizing ADC and LCD modules in embedd ed systems.
- Learned the importance of precise timing and synchronization in hardware interf acing.

Future Improvements:

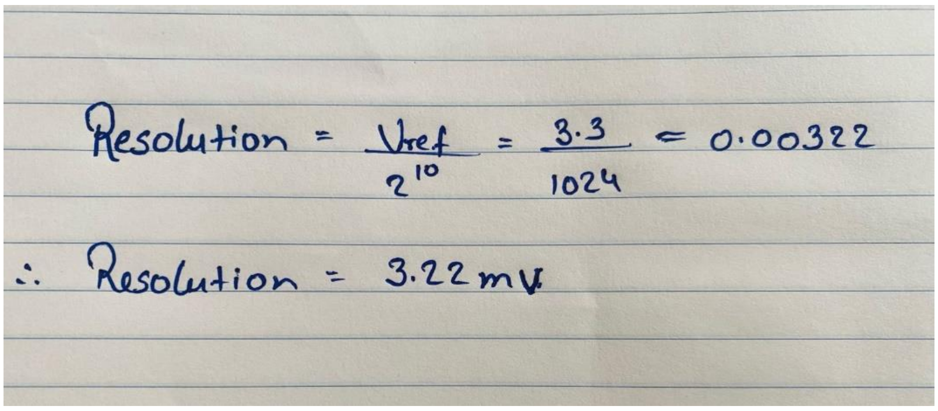
- Automating the date update process.
- Improving the accuracy of voltage readings.

Q.2. Answer the following questions related to LPC2378

a) What is the smallest change in input voltage that the ADC can detect?

(+Vref =3.3 V)

The LPC2378's ADC is a 10bit ADC, which means it can distinguish 2^10=1024 different levels. Given a reference voltage (Vref) of 3.3V:



$$\text{Resolution} = \frac{V_{ref}}{2^{10}} = \frac{3.3}{1024} = 0.00322$$

$$\therefore \text{Resolution} = 3.22\,mV$$

So, the smallest change in input voltage the ADC can detect is approximately **0.00322V** or **3.22mV**.

b) What is the maximum clock frequency needed by ADC of LPC2378?

The ADC in the LPC2378 requires a clock frequency between 4.5MHz and 30MHz to fun ction correctly. However, it is recommended to use a clock frequency of 4MHz to achiev e the best accuracy.

C) Give the steps to program timer for 2 second delay generation with calculation. Assume CCLK=48MHz.

Given:

- **CCLK = 48MHz**
- **We need a 2-second delay.**

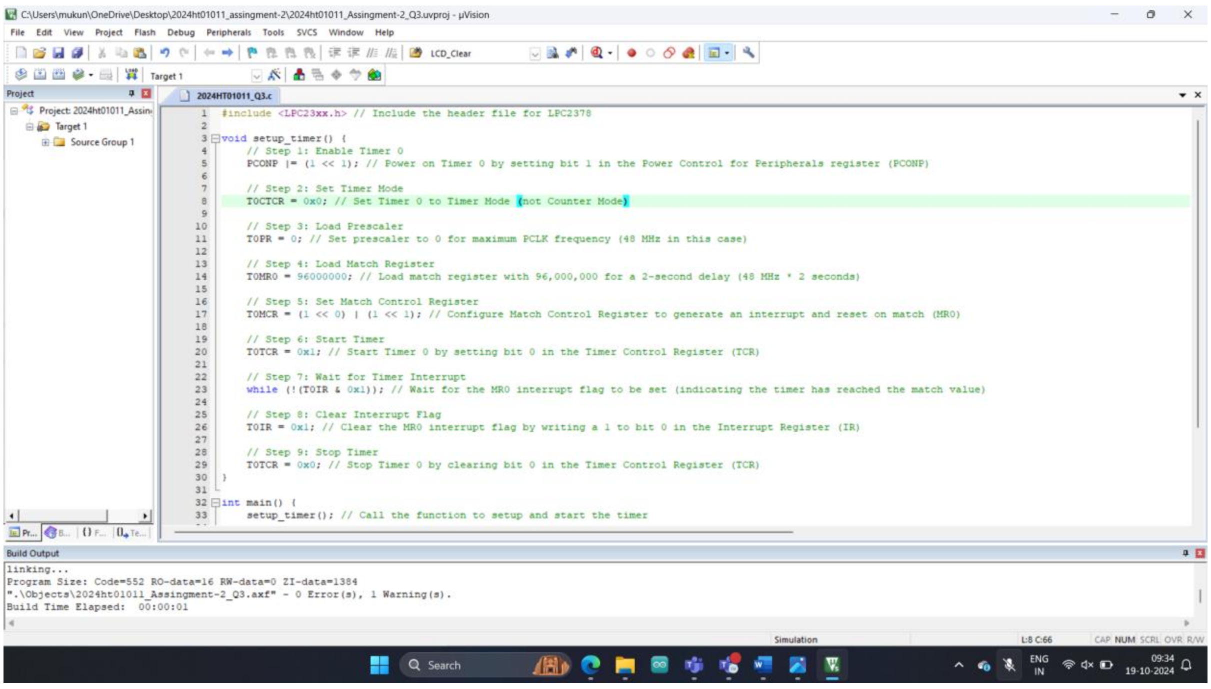
The timer on the LPC2378 uses a prescaler to divide the input clock (CCLK). To get a 1second delay, we need the timer counter to count up to a value where the elapsed time equals 1 second.

1. **Determine the Prescaler Value:**
    - **Timer clock (PCLK) can be derived by dividing the CCLK by the prescaler.**
    - **Assuming the prescaler is set to 1 (no division), PCLK = CCLK = 48MHz.**

2. **Calculate the Timer Count Value:**
    - **Timer Count = PCLK × Delay Time**
    - **For a 1-second delay: Timer Count = 48,000,000 ticks**
    - **For a 2-second delay: Timer Count = 48,000,000 × 2 = 96,000,000 ticks**

Steps to Program:

1. **Enable Timer**
2. **Set Timer Mode**
3. **Load Prescaler**
4. **Load Match Register**
5. **Set Match Control Register**
6. **Start Timer**
7. **Wait for Timer Interrupt**
8. **Clear Interrupt Flag**
9. **Stop Timer**


C:\Users\mukun\OneDrive\Desktop\2024ht01011_assingment-2\2024ht01011_Assingment-2_Q3.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
LCD_Clear
Target 1
Project
Project: 2024ht01011_Assin
Target 1
Source Group 1
2024HT01011_Q3.c
Build Output
Simulation
Search

```c
#include <LPC23xx.h> // Include the header file for LPC2378

void setup_timer() {
    // Step 1: Enable Timer 0
    PCONP |= (1 << 1); // Power on Timer 0 by setting bit 1 in the Power Control for Peripherals register (PCONP)

    // Step 2: Set Timer Mode
    T0CTCR = 0x0; // Set Timer 0 to Timer Mode (not Counter Mode)

    // Step 3: Load Prescaler
    T0PR = 0; // Set prescaler to 0 for maximum PCLK frequency (48 MHz in this case)

    // Step 4: Load Match Register
    T0MR0 = 96000000; // Load match register with 96,000,000 for a 2-second delay (48 MHz * 2 seconds)

    // Step 5: Set Match Control Register
    T0MCR = (1 << 0) | (1 << 1); // Configure Match Control Register to generate an interrupt and reset on match (MR0)

    // Step 6: Start Timer
    T0TCR = 0x1; // Start Timer 0 by setting bit 0 in the Timer Control Register (TCR)

    // Step 7: Wait for Timer Interrupt
    while (!(T0IR & 0x1)); // Wait for the MR0 interrupt flag to be set (indicating the timer has reached the match value)

    // Step 8: Clear Interrupt Flag
    T0IR = 0x1; // Clear the MR0 interrupt flag by writing a 1 to bit 0 in the Interrupt Register (IR)

    // Step 9: Stop Timer
    T0TCR = 0x0; // Stop Timer 0 by clearing bit 0 in the Timer Control Register (TCR)
}

int main() {
    setup_timer(); // Call the function to setup and start the timer
```

```
linking...
Program Size: Code=552 RO-data=16 RW-data=0 ZI-data=1384
".\Objects\2024ht01011_Assingment-2_Q3.axf" - 0 Error(s), 1 Warning(s).
Build Time Elapsed:  00:00:01
```