EE443 - Embedded Systems

Experiment 4

LCD Module and Time Markers

Purpose: Display variables on an LCD (liquid crystal display) module. Monitor execution time of operations using pins of a microcontroller as time markers.

Programming Hints

LCD modules are affordable user interfaces for embedded systems to display information or to ask for user input when necessary. Most LCD modules can be controlled through a single 8-bit data port on a microcontroller by using common interface protocols and display instructions.

Tracing and debugging of embedded systems became more difficult with the increasing complexity of microcontrollers and real-time programs. Usage of software tools and laboratory equipment such as logic analyzers is not efficient for most debugging purposes. Debugging with these tools require familiarity with the assembly code generated by the compiler, and in case of ATmega328, keeping all 32 CPU registers under control is a demanding job. A practical debugging method is to print critical variables on a display connected to a microcontroller. As an alternative, debugging information can be sent to another computer with display and storage capabilities through a basic USART line or a similar data connection.

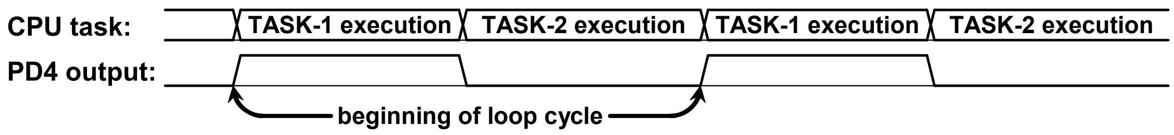
Using Time Markers

Real-time nature of processes on an embedded systems is another factor that makes their analysis challenging. It is not possible to suspend a program just to check the CPU status without changing normal course of real-time events. Instead, it is a good practice to reserve a few general purpose I/O pins for debugging purposes and connect them to test points on the development hardware. These pins can be probed with an oscilloscope to obtain critical timing information while the program is running on the real hardware.

In the following example, **PD4** output (pin-4 of port-D) is used to monitor the timing of a while loop. **PD4** remains high during execution of **TASK-1**, and it is cleared at the beginning of **TASK-2**.

```
while(1)
{ PORTD |= _BV(PORTD4);  // set time marker (OR with 0x10)
  ---  // statements for TASK-1
  ---  // statements for TASK-1
  PORTD &= ~_BV(PORTD4);  // clear time marker (AND with 0xEF)
  ---  // statements for TASK-2
  ---  // statements for TASK-2
}  // end of while loop
```

One can measure loop cycle time and execution time of `TASK-1` and `TASK-2` as shown in the following diagram. **PD4** output can also be a trigger source to capture other related events easily on an oscilloscope or another test device.



Preliminary Work

1. Download the files available for Experiment 4. The source file, **LCDmodule.c**, and the header file **LCDmodule.h**, provide the necessary library functions to initialize and use the LCD interface. Read the instructions given in **LCDmodule.h** to become familiar with the library functions.

Note that, execution of the two LCD module functions, **`LCD_Init(.)`** and **`LCD_Clear(.)`** takes a long time (**~4 ms**). These functions should be called **only once** in initialization section of the main program. Calling these functions in the main loop will cause long time delays.

Use the **`sprintf(...)`** C library function (include **`<stdio.h>`**) to obtain the character string that will be sent to the LCD module:

```
// Declare the character array for one LCD line:
char  LCDtext[16];
unsigned char  Abyte;
---
---
// Write 3-digit decimal number into LCDtext:
sprintf(LCDtext, "Abyte=%3d", Abyte);
// Place LCD cursor at column-1 of line-1:
LCD_MoveCursor(1, 1);
// Send LCDtext to the LCD module:
LCD_WriteString(LCDtext);
```

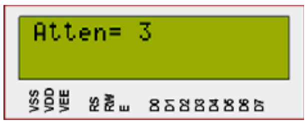
WinAVR compiler may give an error message on the **`sprintf(...)`** function on some computers depending on the system configuration. As an alternative, you can use the **`PrintByte(...)`** function provided in **PrintByte.c** to avoid errors resulting from **`sprintf(...)`**.

2. Write a main program to perform the following tasks.

- Initialize LCD interface through port-B.
- Setup port-D so that pin-0 and pin-1 are the inputs for detecting two switch settings, and pin-4 and pin-5 are the time marker outputs.
- In the main loop:

  1. Change the value of a one-byte variable (i.e. **Atten**) between **1** and **11** with two push-button switches:
  - Decrement **Atten** (if Atten > 1) every time **SW0** is pressed.
  - Increment **Atten** (if Atten < 11) every time **SW1** is pressed.

  2. Display the **Atten** result as a 3-digit decimal number on the LCD. The LCD display should appear as shown on the right.



  3. Use **`_delay_us(.)`** macro function to obtain **500 µs** delay time before the end of the loop.

Procedure

1. Create a new project directory and a new AVR project for Experiment 4. Copy **LCDmodule.c** and **LCDmodule.h** into the project directory, and add these files to the AVR project in Code::Blocks. Compile the main program you wrote in the preliminary work. Don't forget to include **LCDmodule.h** in your main program file.

Note: WinAVR compiler may give an error message related to long integer multiply operation when **`sprintf(...)`** function is used. In that case, use **`PrintByte(...)`** instead of **`sprintf(...)`** and skip step-4 below.

2. Copy the provided Proteus design file, **Exp4_LCD.dsn**, into the project directory and open it. Activate the **Edit Properties** window for **ATMEGA328P** microcontroller on the ISIS schematic and select the **.hex** file generated by the compiler for Experiment-4. Test your program looking at the animated simulation results, and debug it, if necessary.

3. Modify the main program adding two time markers at **`PD4`** and **`PD5`** pins. Initialize the corresponding **`DDRD`** register bits to use **`PD4`** and **`PD5`** as outputs. In the main loop, set **`PORTD`** bits, so that **`PD4`** will be high during execution of **`sprintf(...)`** function, and **`PD5`** will be high while **`LCD_MoveCursor(..)`** and **`LCD_WriteString(.)`** are running.

4. Place an oscilloscope (select the **Virtual Instruments** list on the main toolbar) and probe **`PD4`** and **`PD5`** outputs of ATmega328. Measure the following timing parameters on the oscilloscope window.

- main while loop cycle time
- execution time of **`sprintf(...)`**
- total execution time of LCD module functions

Check if the measured values are the same for all **`Atten`** values between **1** and **11**.

Note: If you close the oscilloscope window, you can activate it again by using the **Reset Popup Windows** option under the **Debug** menu of ISIS.

5. Replace the **`sprintf(...)`** function with **`PrintByte(...)`**. Measure execution time of **`PrintByte(...)`** and compare with the execution time of **`sprintf(...)`**.

6. Minimize the time spent in display operations with the following changes.

- Write the text "**Atten=**" once as part of the initialization, and update only the displayed number in the main loop.
- Update the displayed number only after the **`Atten`** value is changed.

Measure the loop cycle times with and without the display updates.