

EEE404/591 – Real Time DSP – Lab 4

Introduction to Processor Expert

-Hemanth Balaji Dandi

REPORT

Objective

The objective of this lab is to introduce students to the Processor Expert (PE) plug-in available with Metrowerks Codewarrior. Students will be introduced to the concept of components and will establish serial communication between the PC and the DSP.

Results and Analysis

A. Build the project and load it into the chip by pressing F5. Test the code by observing the green and red LEDs. What happens when IRQ_A push button is pressed? Try to change the blinking rate for the green LED.

- The project was built and loaded. When the IRQ_A button was pressed, the Red LED started to blink. This is because whenever we call the IRQ-A subroutine, the function Red_NegVal() starts to run, which is a toggling function for the Red LED. The Blink Rate of the Green LED was changed by varying the Interrupt Period in the Timer register created for the green LED from 1 sec to 0.5 sec and the corresponding results were seen.

B. Sending/Receiving Data

- 1) Describing briefly the memory mapped registers configured by Processor Expert when using the AsynchroSerial bean. Please refer to the generated code to identify what registers are initialized by PE and look those registers up in the “DSP5685x 16- Bit Digital Signal Processing User’s Manual”**

- **RecvChar**

Register Used : SCI Status Register (SCI_0_SR)

Role : Checks if a character is received by the board, else returns an error, ie, it checks if the receiver is empty, and if so, returns an error. This is done by checking the Overrun Flag (OR) bit or the Noise Flag(NF) bit, or the Framing Error Flag bit (FE) or the Parity Error Flag (PE) bit to see if anyone of the four are enabled, and if so, return an error.

It also checks the Receive Data Register Full Flag (RDRF) bit to see if there is no data to be received from the Data Register, and if so, return an error, otherwise continue in receiving the character

Register Used :SCI Data Register (SCI_0_SR)

Role : Reads a character from the receiver and whenever this is called, updates the corresponding received char value automatically by using a reference, ie, &Chr (Being called by a reference but declared as a pointer)

- **SendChar**

Register Used : SCI Status Register (SCI_0_SR)

Role : Checks if the transmitter is empty, and if so, returns an error. This means that it checks the r Data Register Empty Flag (TDRE) bit to see if there is any data in the transmitter, and if so, return an error.

Register Used :SCI Data Register (SCI_0_SR)

Role : Writes and stores the value of the character/data to be sent in this register

- **GetCharsInRxBuf**

Register Used : SCI Status Register (SCI_0_SR)

Role : Checks and returns the number of characters in the receiver . This is done by checking whenever the Receiver Data Register Full Flag (RDRF) is set to 1, ie, if a character is received from the SCI Data Register to the receiver buffer, this bit is set, hence the number of times the bit is set, is the number of characters received.

Baud Rate Experiment

1)The initial channel rate is 9600 bit/s. Try to change that, both on the PC side and DSP side, to higher rates such as 19200, 38400, 57600, 115200, 128000. Try different combinations where the rates are matched and also combinations where they are not. Tabulate your observations as shown in Table 2. Also try to experimentally determine the maximum baud rate on the PC side by keeping both PC and DSP matched and progressively increasing the baud rate until it crosses the calculated maximum Baud rate of the PC. Does the calculated value concur with what you observe? Record this in a separate table using the format shown in the below Table .

Baud Rate of Serial Port of PC	Baud Rate of DSP56858EVM Kit	Message Observed	Screenshot	Comments/Observations
--------------------------------	------------------------------	------------------	------------	-----------------------

19200	19200	Bullseye!	Fig 2	Message is displayed as expected
38400	38400	Hello World!	Fig 3	Message is displayed as expected
57600	57600	PE is Awesome!	Fig 4	Message is displayed as expected
115200	115200	PE is Awesome!	Fig 5	Message is displayed as expected
128000	128000	No Output	Fig 6	No Output as maximum baud rate has been reached of the PC
9600	38400	Garbage values	Fig 7	Garbage Values as the baud rates b/w PC and the board must be the same.
38400	57600	Garbage values	Fig 8	Garbage Values as the baud rates b/w PC and the board must be the same.

Observation:

We can observe that only if the Baud Rates between the DSP56858EVM board and the PC are the same, the message can be displayed, else we get an error output filled with Garbage values.

The Maximum Baud Rate of the PC is-

PCmaxbaudrate- $\text{UARTClock}/16 = 1.8342/16 = 11500$ baud.

Hence a value of 128000 baud could not be used to display the message. 115200 could however be used as it is close to the 11500 baud.

Screenshots

```
clear all;
s1 = serial('COM1'); % Creating serial object
s1.BaudRate = 9600; % Setting Baud Rate
s1.Parity='none'; % Setting parity
s1.Stopbits = 1; % Enable stop bit
s1.Terminator='LF'; % setting the terminator

% Sending Hello World!
Tx_str = 'Bullseye!!\n';
fopen(s1); %Open serial port for communication
fprintf(s1,Tx_str); % sending data thro' the serial port
Rx_str = fscanf(s1,'%c'); %receiving characters thro' the serial port
msgbox(Rx_str,'Serial Com'); %outputting received message
fclose(s1); %closing communication
delete(s1); % Deleteing s1 object
clear all; % clearing memory
```

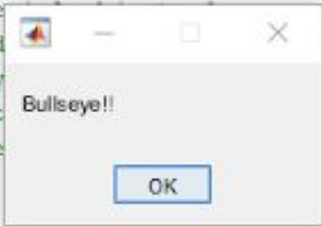


Fig1

```
1 - clear all;
2 - s1 = serial('COM1'); % Creating serial object
3 - s1.BaudRate = 19200; % Setting Baud Rate
4 - s1.Parity='none'; % Setting parity
5 - s1.Stopbits = 1; % Enable stop bit
6 - s1.Terminator='LF'; % setting the terminator
7 -
8 - % Sending Hello World!
9 - Tx_str = 'Bullseye!!\n';
10 - fopen(s1); %Open serial port for communication
11 - fprintf(s1,Tx_str); % sending data thro' the serial port
12 - Rx_str = fscanf(s1,'%c'); %receiving characters thro' the serial port
13 - msgbox(Rx_str,'Serial Com'); %outputting received message
14 - fclose(s1); %closing communication
15 - delete(s1); % Deleteing s1 object
16 - clear all; % clearing memory
17 -
```

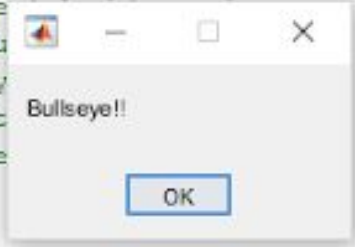


Fig2

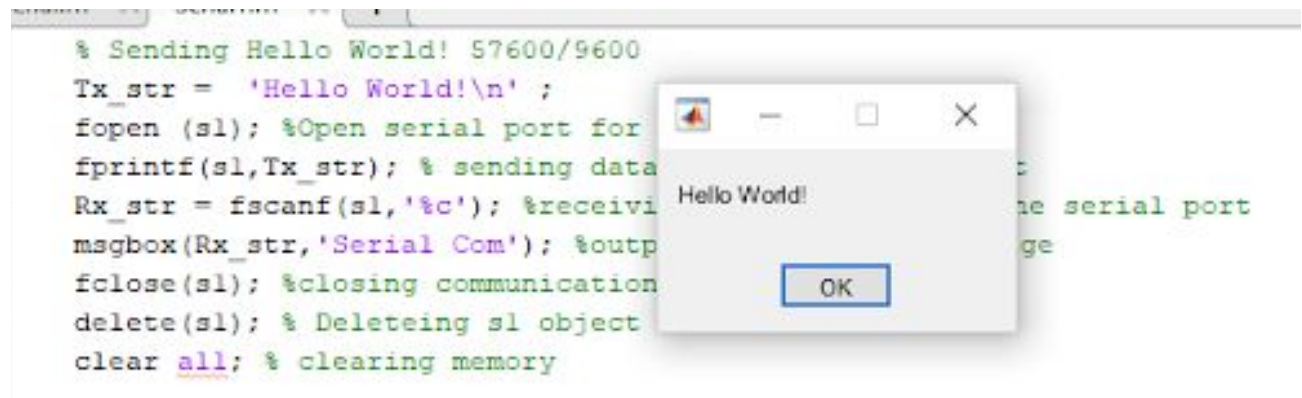


Fig3

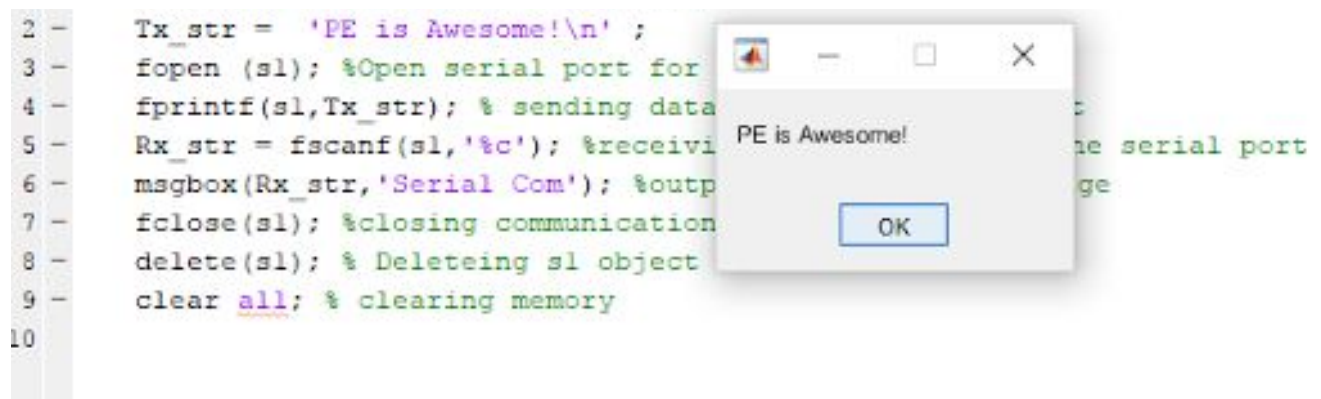


Fig4

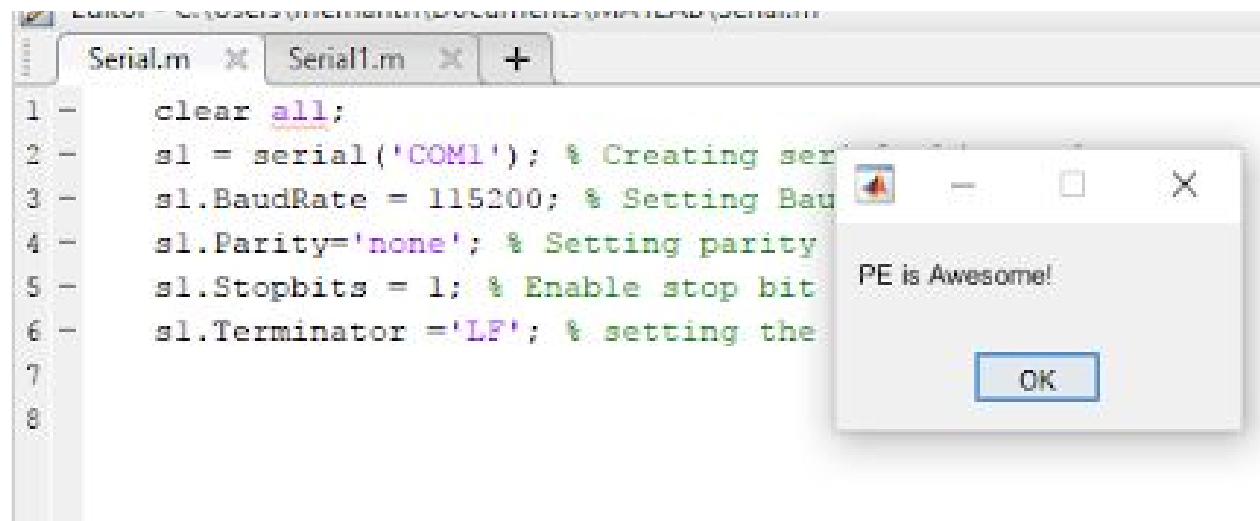
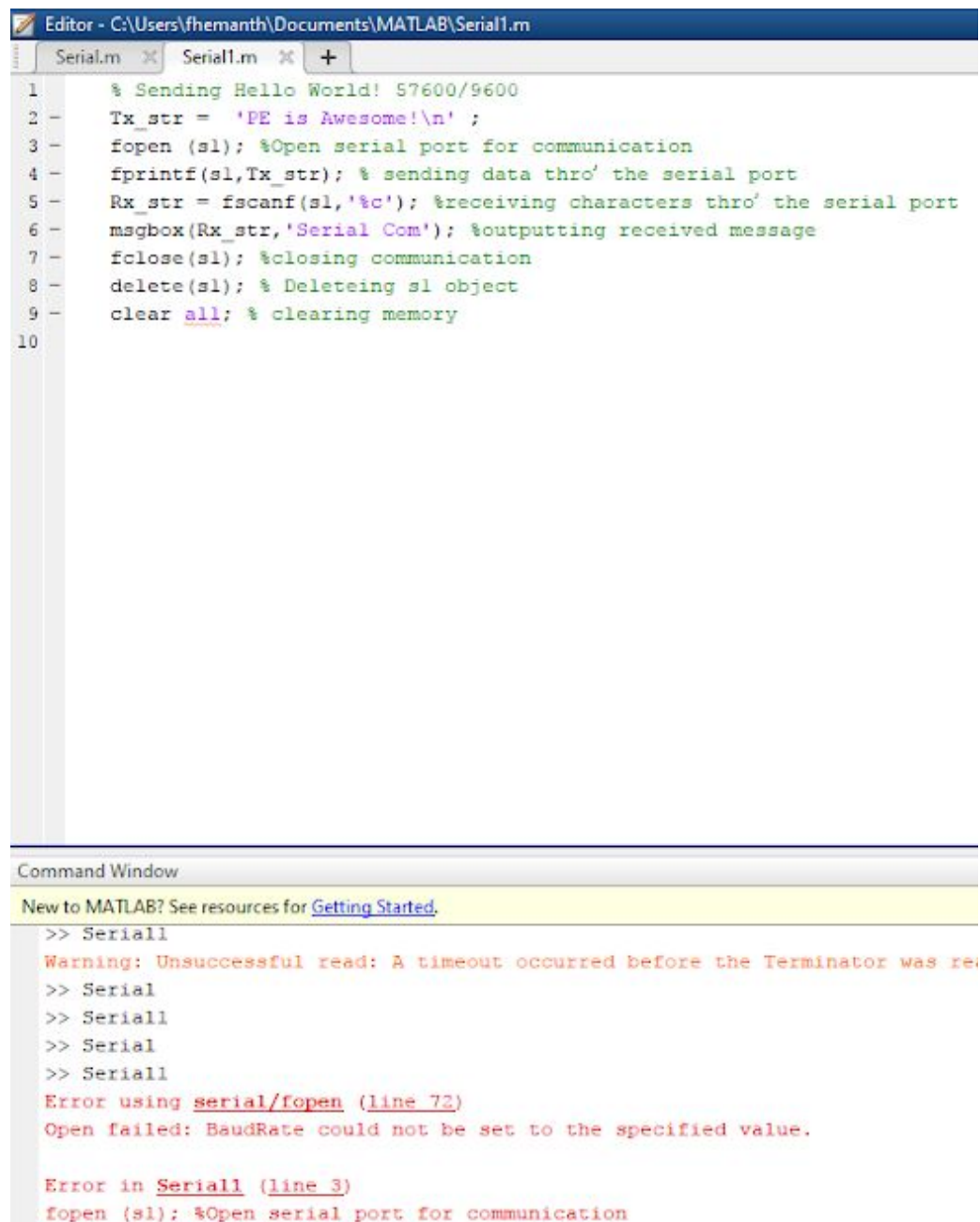


Fig5



The image displays the MATLAB environment. The top window is the Editor, showing a script named Serial1.m. The script contains the following code:

```
1 % Sending Hello World! 57600/9600
2 Tx_str = 'PE is Awesome!\n' ;
3 fopen (s1); %Open serial port for communication
4 fprintf(s1,Tx_str); % sending data thro' the serial port
5 Rx_str = fscanf(s1,'%c'); %receiving characters thro' the serial port
6 msgbox(Rx_str,'Serial Com'); %outputting received message
7 fclose(s1); %closing communication
8 delete(s1); % Deleteing s1 object
9 clear all; % clearing memory
10
```

The bottom window is the Command Window, showing the execution of the script. It displays several warnings and errors:

```
>> Serial1
Warning: Unsuccessful read: A timeout occurred before the Terminator was re
>> Serial
>> Serial1
>> Serial
>> Serial1
Error using serial/fopen (line 72)
Open failed: BaudRate could not be set to the specified value.

Error in Serial1 (line 3)
fopen (s1); %Open serial port for communication
```

Fig6

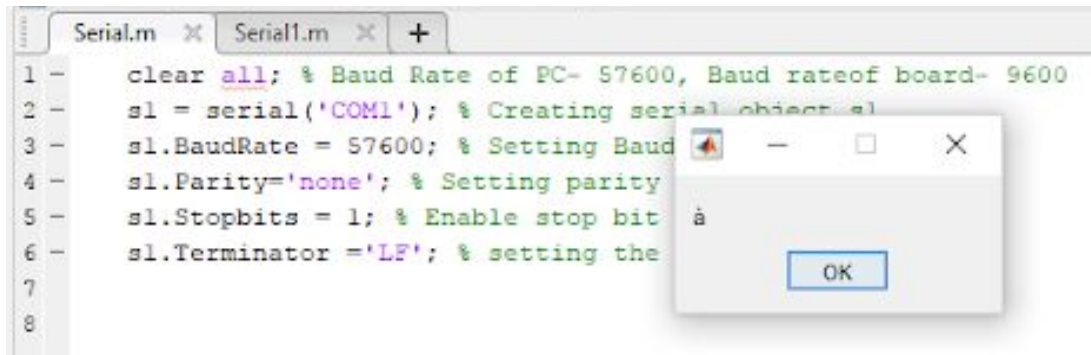


Fig7

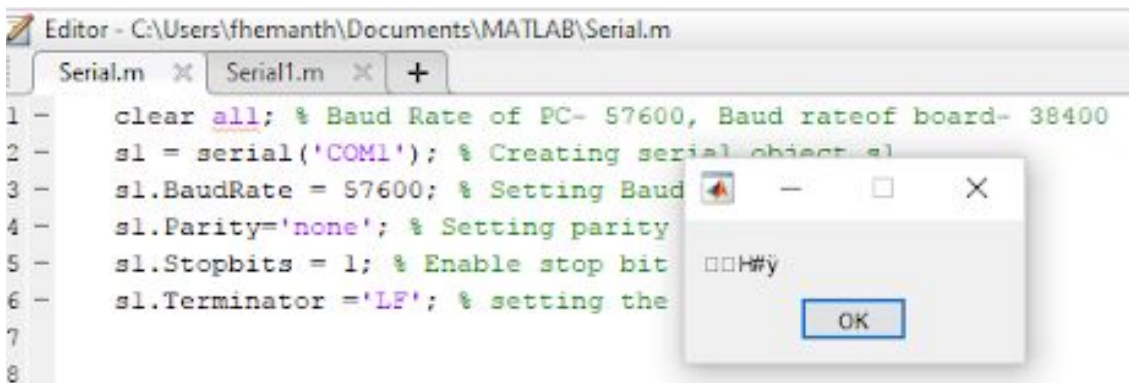


Fig8

2) Using the material presented in Tutorial 1 & 2, create a project including both 'BitIO' and 'AsynchroSerial' components. Write the necessary C code to control the LEDs as shown in Table 3. You must send the desired LED choice (1 -6) from MATLAB, receive that value in the C code, and use the value to perform the operation given in Table 3.

- **The C Code is:**

```

void main(void)
{
    byte temp;
    word r;
    LED0_ClrVal(); // Clearing all the LEDs initially
    LED1_ClrVal();
    LED2_ClrVal();
    LED3_ClrVal();
    LED4_ClrVal();
    LED5_ClrVal();
}

```

```

PE_low_level_init(); /** End of Processor Expert internal initialization. */
/* Infinite Loop */
for(;;) { /*check if a character has been received in the receive buffer */
r = Serial_com_GetCharsInRxBuf();
if (r==1) { /* A character has been received. */
Serial_com_RecvChar(&temp); /* Read char from receive buffer into temp. */

if(temp=='1') // If the character sent is 1, switch on LED0, and turn the rest off.
{ LED0_SetVal();
LED1_ClrVal();
LED2_ClrVal();
LED3_ClrVal();
LED4_ClrVal();
LED5_ClrVal();
}
else if(temp=='2') // If the character sent is 2, switch on LED1, and turn the rest off.
{LED0_ClrVal();
LED1_SetVal();
LED2_ClrVal();
LED3_ClrVal();
LED4_ClrVal();
LED5_ClrVal();
}
else if(temp=='3') // If the character sent is 3, switch on LED2, and turn the rest off.
{LED0_ClrVal();
LED1_ClrVal();
LED2_SetVal();
LED3_ClrVal();
LED4_ClrVal();
LED5_ClrVal();
}
else if(temp=='4') // If the character sent is 4, switch on LED3, and turn the rest off.
{LED0_ClrVal();
LED1_ClrVal();
LED2_ClrVal();
LED3_SetVal();
LED4_ClrVal();
LED5_ClrVal();
}
else if(temp=='5') // If the character sent is 5, switch on LED4, and turn the rest off.
{LED0_ClrVal();
LED1_ClrVal();
LED2_ClrVal();

```



```

LED3_ClrVal();
LED4_SetVal();
LED5_ClrVal();
}
else if(temp=='6') // If the character sent is 6, switch on LED5, and turn the rest off.
{LED0_ClrVal();
LED1_ClrVal();
LED2_ClrVal();
LED3_ClrVal();
LED4_ClrVal();
LED5_SetVal();
}
/* Send char in temp to send buffer */
Serial_com_SendChar(temp);} }

```

Output

If we write in any number from (1-6) in MATLAB, the respective LEDS(0-5) turn on while the rest, off.

3) GRAD ONLY: Control the blinking rate of the LED 0 during runtime depending on the value received from the PC. Keep all other LEDs turned off. For example, the PC can send digits 1 to 4 corresponding to a blinking rate going from 0.25s to 1s with a 0.25s step size. The blink rate of LED 0 should change as the digit is received.

- We create a Timer0123 Compare, and change the Interrupt period to a list of values with 0.25 sec-1 sec with step size 0.25 sec.

C Code:

```

/**This is if the user sends (1-4), the respective blinking rate (0.25,0.5,0.75,1sec) gets activated for LED0
if(temp=='1') // If the user sends '1', blink-rate-0.25sec
{ TI1_SetPeriodMode(0);
}
else if(temp=='2') // If the user sends '2', blink-rate-0.50sec
{TI1_SetPeriodMode(1);
}
else if(temp=='3') // If the user sends '3', blink-rate-0.75sec
{TI1_SetPeriodMode(2);

}
else if(temp=='4') // If the user sends '4', blink-rate-1.00sec
{TI1_SetPeriodMode(3);
}

```

```
}***/
```

Output

If we write in any number from (1-4) in MATLAB, the respective Blink rates (0.25sec - 1sec) of LED0 activates while the rest of the LEDs remain off.

Lab Evaluation:

Difficulties encountered:

- Understanding why to pass &Char instead of simply 'char' through the function call when the values are received into the buffer.

Errors or Unclear Statements in Lab Manual:

- Every objective was clearly explained. This wasn't an error, but as mentioned by the TA at the start, the MATLAB code was to be split into two for better compilation and to make the board understand.

Suggestions:

Level of problems given were fine considering that we have just begun understanding assembly language.

Conclusion

Learnt about

- PE and its functionalities
- Access LEDs through PE and control the toggling behaviour
- Serial Communication through PE via the board and the PC (MATLAB)
- Controlling the baud rates of sent message from PC to the board
- Controlling the LEDs through the PC (MATLAB) to the board
- Controlling the Blinking rates of an LED from the PC directly via the board

Liked Most?

Serial Communication

Liked Least?

Writing this report (Just kidding!) Not much NOT to like. Assembly language is quite interesting!