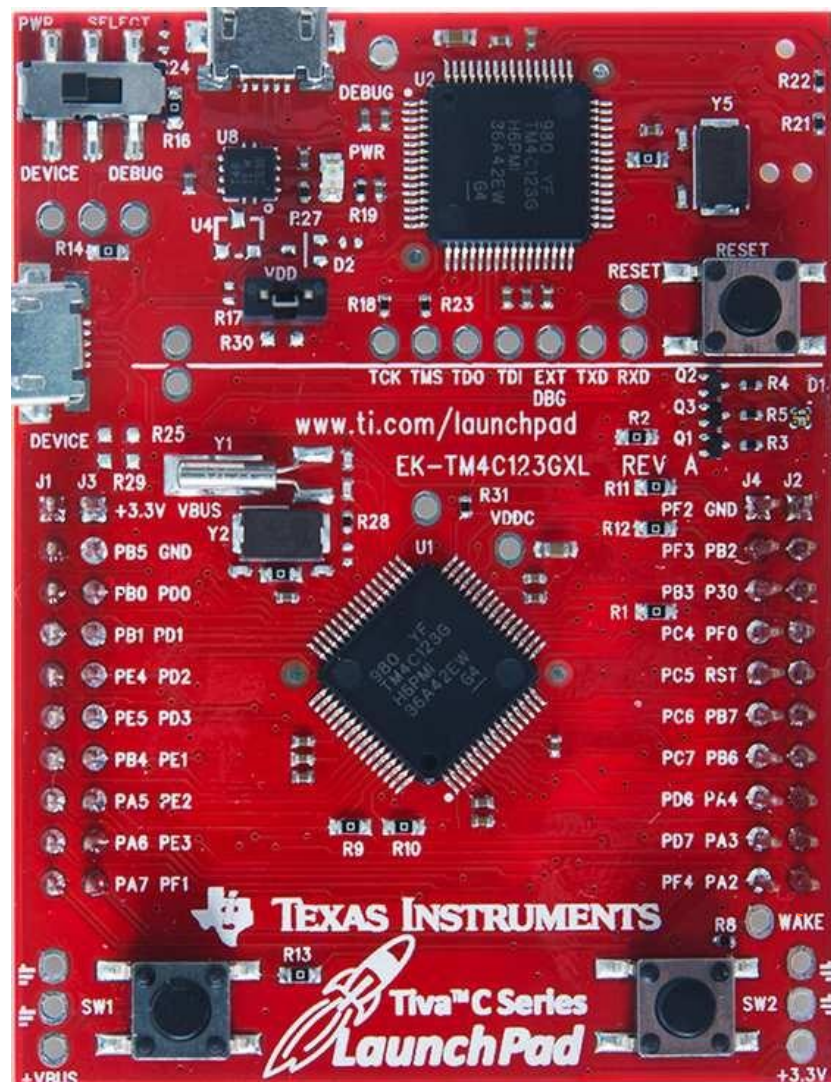
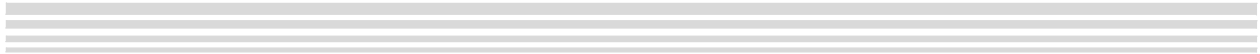


Embedded System Design using TM4C LaunchPad™ Development Kit



Experiment 3

UART - ECHO!



Topics	Page
3.1 Objective	83
3.2 Introduction	83
3.3 Component Requirements	84
3.4 Software	85
3.5 Procedure.....	88
3.6 Observation	89
3.7 Summary	89
3.8 Exercise.....	89

3.1 Objective

The main objective this experiment is to connect the EK-TM4C123GXL to a PC terminal and send an echo of the data input back to the PC using UART. In this experiment, we will also understand the basics of serial communication to send and receive data.

3.2 Introduction

The SCI (Serial Communication Interface) modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format. The SCI receiver and transmitter each have a 4-level deep FIFO for reducing servicing overhead. Each FIFO has its own separate enable and interrupt bits. Both can be operated independently for half-duplex communication, or simultaneously for full-duplex communication. The bit rate is programmable to different speeds through a 16-bit baud-select register.

In this experiment, the SCI module is configured to transmit the received data from an external device along with some message to indicate the reception. The functional block diagram as shown in [Figure 7-1](#) illustrates the working principle of the experiment. The UART of the EK-TM4C123GXL is connected to the serial port of the PC that runs over the debug USB cable. The serial port can be monitored by the PC with the help of Tera Term or any other serial interface software. In this experiment, the UART receives serial data via the USB cable and displays the data received on the serial monitor (Tera Term) through serial communication.

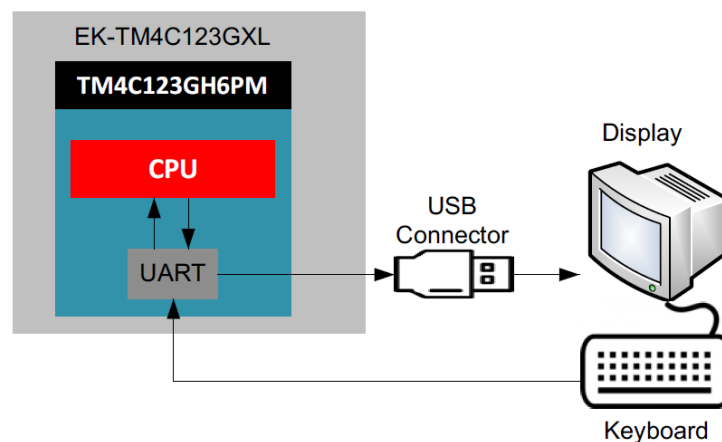


Figure 7-1 Functional Block Diagram

3.2.1 UART Module

The TM4C123GH6PM controller includes eight Universal Asynchronous Receiver/Transmitter (UART) with programmable baud-rate generator, allowing speeds of up to 5 Mbps for regular speed (divide by 16) and 10 Mbps for high speed (divide by 8). It has separate 16x8 transmit (TX) and receive (RX) FIFOs (First In First Out) to reduce CPU interrupt service loading. It has a programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface. The FIFO trigger levels available are of 1/8, 1/4, 1/2, 3/4, and 7/8. It also has standard asynchronous communication bits for start, stop, and parity and detects and generates line-break.

The characteristics of fully programmable serial interface are:

- 5, 6, 7, or 8 data bits

- Even, odd, stick, or no-parity bit generation/detection
- 1 or 2 stop bit generation

It supports Standard FIFO-level and End-of-Transmission interrupts and efficient transfers using Micro Direct Memory Access Controller (μ DMA)¹ which has separate channels for transmit and receive. The EK-TM4C123GXL has an internal USB to UART converter to connect to the UART of the processor.

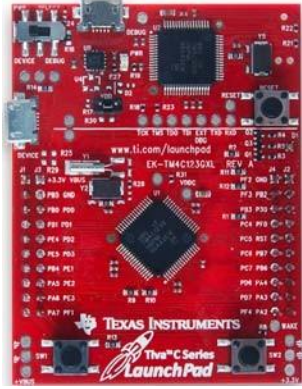
3.3 Component Requirements

3.3.1 Software Requirement

1. [Keil](#)
2. Serial Terminal Software (eg: Tera Term)
3. [TivaWare_C_Series](#)

3.3.2 Hardware Requirement

Table 7-1: Components Required for the Experiment

S.No	Components	Specifications	Images
1.	Tiva LaunchPad	EK-TM4C123GXL LaunchPad	
2.	USB cable		

1. The TM4C123GH6PM microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (DMA). The DMA controller provides a way to offload data transfer tasks from the Cortex™-M4F processor, allowing for more efficient use of the processor and the available bus bandwidth.

3.4 Software

The software for the experiment is written in C and developed using the CCS Integrated Development Environment (IDE). Refer to “[Project Creation and Build](#)” in the Getting Started section of this manual for project build and debug using CCS. The software is programmed into the target device TM4C123GH6PM on the EK-TM4C123GXL using the USB interface.

3.4.1 TeraTerm Setup

The serial port configuration for the Tera Term is:

Baud rate is 115200, Parity is none, 8 Data Bits and Stop Bit is 1.

Select the configuration and Press **OK** as shown in [Figure 7-2](#).

Note: COM Port varies in different PCs based on the hardware

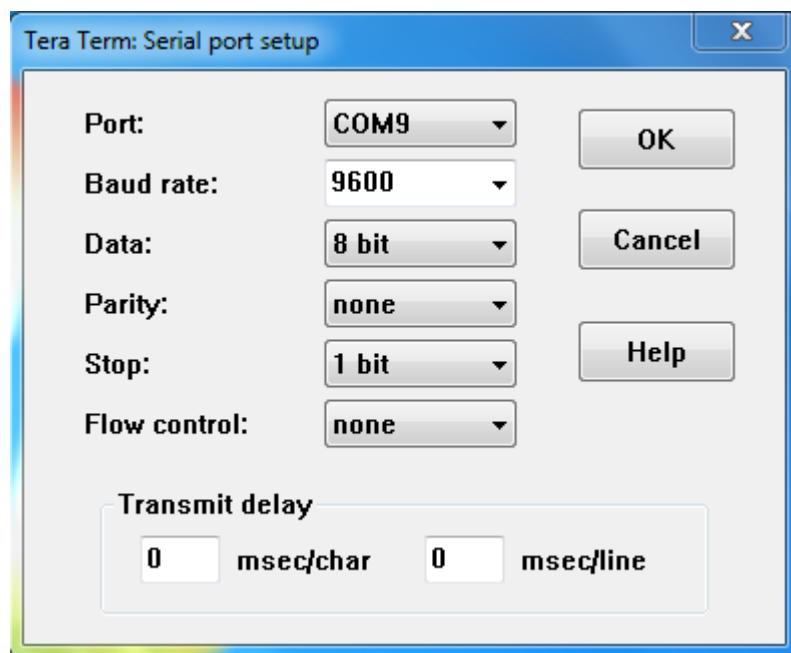


Figure 7-2 Tera Term Serial Port Setup Window

3.4.2 Flowchart

The flowchart for the program is shown in [Figure 7-3](#). The system clock is configured and enabled to 40MHz. The GPIO Port A pins 0 and 1 are enabled and configured as UART receiver and transmitter respectively. The UART peripheral is configured with Baud rate 115200, Parity none, 8 Data Bits and Stop Bit 1. The default message "Echo output" is transmitted by the UART to the serial terminal in the PC. The program then waits for serial input from the PC using a while loop. When the UART receives an input, it sends the same data to PC terminal as output. Thus, the input is echoed on the Terminal window.

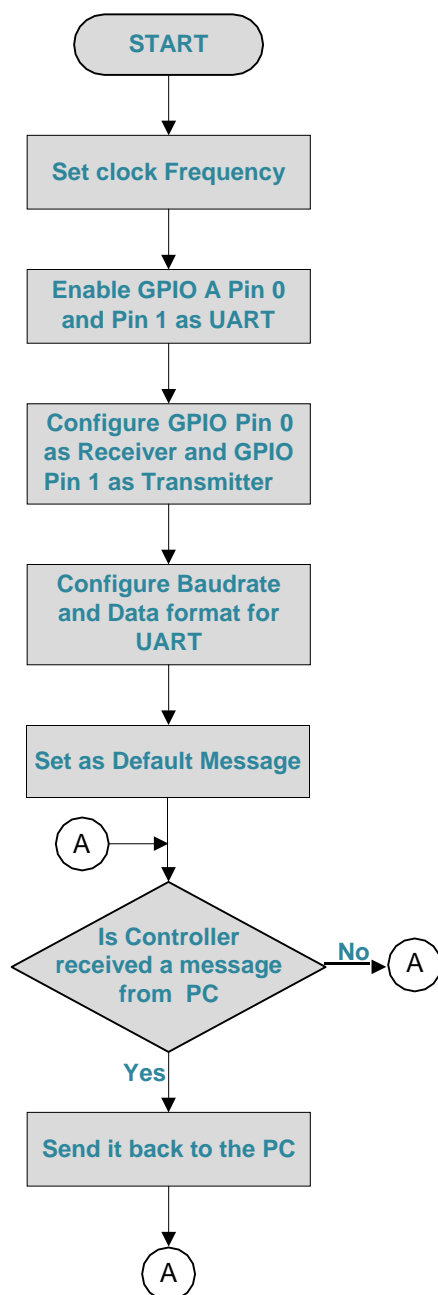


Figure 7-3 Flowchart for UART - Echo

3.4.3 C Program Code for UART - Echo

```

#include<stdint.h>
#include<stdbool.h>
#include"inc/hw_memmap.h"
#include"inc/hw_types.h"

```

```
#include"driverlib/gpio.h"

#include"driverlib/pin_map.h"

#include"driverlib/sysctl.h"

#include"driverlib/uart.h"

#define GPIO_PA0_U0RX  0x00000001

#define GPIO_PA1_U0TX  0x00000401

intmain(void)

{
SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|
SYSCTL_XTAL_16MHZ);
SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);

SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

GPIOPinConfigure(GPIO_PA0_U0RX);
GPIOPinConfigure(GPIO_PA1_U0TX);
GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200,
(UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));


UARTCharPut(UART0_BASE, 'E');
UARTCharPut(UART0_BASE, 'c');
UARTCharPut(UART0_BASE, 'h');
UARTCharPut(UART0_BASE, 'o');
UARTCharPut(UART0_BASE, ' ');
UARTCharPut(UART0_BASE, 'O');
UARTCharPut(UART0_BASE, 'u');
UARTCharPut(UART0_BASE, 't');
UARTCharPut(UART0_BASE, 'p');
UARTCharPut(UART0_BASE, 'u');
UARTCharPut(UART0_BASE, 't');
UARTCharPut(UART0_BASE, ': ');
UARTCharPut(UART0_BASE, ' ');
UARTCharPut(UART0_BASE, '\n');
```

```

while (1)
{
    if (UARTCharsAvail(UART0_BASE)) UARTCharPut(UART0_BASE,
UARTCharGet(UART0_BASE));
}
}

```

Table 7-2: API Functions Used in the Application Program

API Function	Parameters	Description
GPIOPinTypeUART(uint32_t ui32Port, uint8_t ui8Pins)	<ul style="list-style-type: none"> • ui32Port is the base address of the GPIO port • ui8Pins is the bit-packed representation of the pin(s) 	Configures pin(s) for use by the UART peripheral
UARTConfigSetExpClk(uint32_t ui32Base, uint32_t ui32UARTClk, uint32_t ui32Baud, uint32_t ui32Config)	<ul style="list-style-type: none"> • ui32Base is the base address of the UART port • ui32UARTClk is the rate of the clock supplied to the UART module • ui32Baud is the desired baud rate • ui32Config is the data format for the port (number of data bits, number of stop bits, and parity) 	Sets the configuration of a UART.
UARTCharsAvail(uint32_t ui32Base)	<ul style="list-style-type: none"> • ui32Base is the base address of the UART port • ucData is the character to be transmitted 	Waits to send a character from the specified port.
UARTCharsAvail(uint32_t ui32Base)	<ul style="list-style-type: none"> • ui32Base is the base address of the UART port 	Determines if there are any characters in the receive FIFO.
UARTCharGetNonBlocking(uint32_t ui32Base)	<ul style="list-style-type: none"> • ui32Base is the base address of the UART port 	Receives a character from the specified port.

3.5 Procedure

1. Connect the EK-TM4C123GXL to the PC using the USB cable supplied.
2. Build, program and debug the code.
3. Open Tera Term UART terminal window and configure which is explained in [Section 7.4.1](#).
4. Type characters on the keyboard and observe the terminal window.

3.1 Observation

After compiling and running the program the Serial Terminal displays - '**Echo Output:**'. As the characters are typed in, they will be displayed on the terminal.

The input entered from the keyboard is echoed back and shown on the terminal window. The keyboard entries in Tera Term are generally streamed only to the serial port and not to the display. Only the characters received via the serial port are displayed on this window. Any alphanumeric string could be sent for testing.

To confirm this, hold the RESET key on the EK-TM4C123GXL and try typing the input string. You will observe no display of the characters typed. We can thus infer that the program in the EK-TM4C123GXL is doing the echo operation.

Figure 7-4 shows display of typed data (UART input) **TEXAS INSTRUMENTS INC.** being echoed back on the serial terminal window.

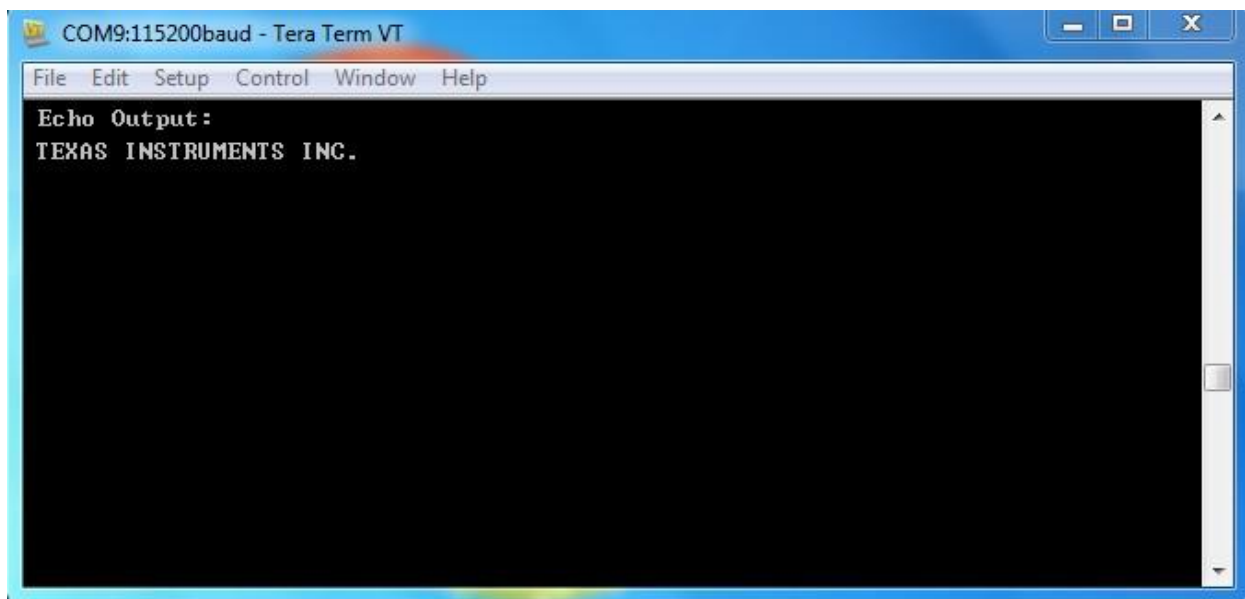


Figure 7-4 Output on Serial Terminal Window

3.2 Summary

We have successfully configured the serial port to send and receive bytes and monitor the data on a serial terminal. We have also learnt how the UART peripheral can be used for serial interface.

3.3 Exercise

1. Change the baud rate to 19200 and repeat the experiment.

Hint: If the serial terminal baud rate has not changed, junk characters will be received. So, it is important that two serially communicating partners to have the same configuration.

2. What is the maximum baud rate that can be set in the UART peripheral of Tiva?
3. Modify the software to display "Switch pressed" by pressing a user input switch on the Launch-Pad.