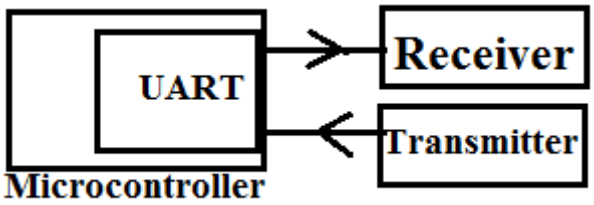
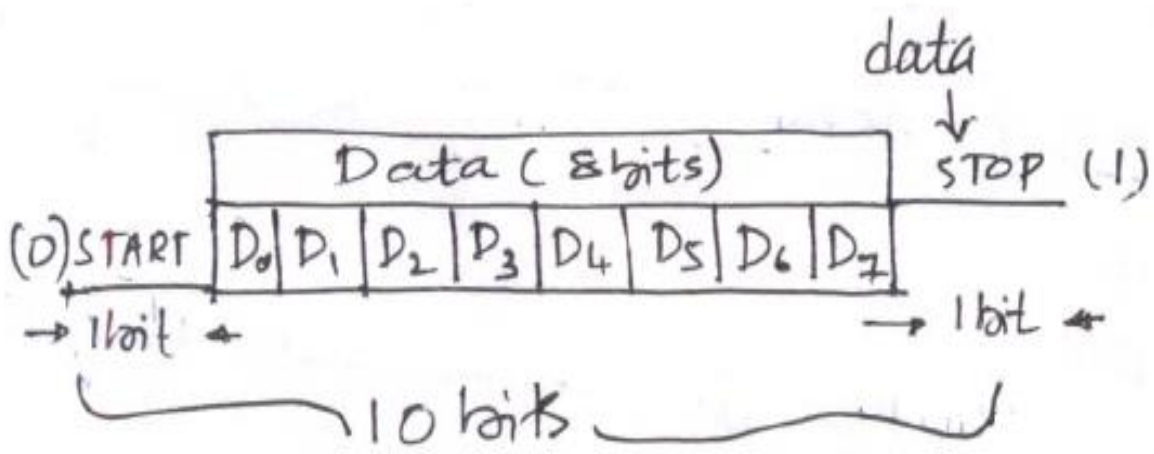
LPC 2148 UART Programming

LPC 2148 supports full duplex serial asynchronous communication, UART (universal asynchronous receiver and transmitter) is built in to the chip. LPC 2148 povides two UARTs, UART0 and UART1. They are popularly called serial ports. They follow Universal Asynchronous Serial Communication protocol, hence each port is referred called as Universal Asynchronous Receiver Transmitter. Every UART provides two pins, TxD for Transmission and RxD for Reception of data.



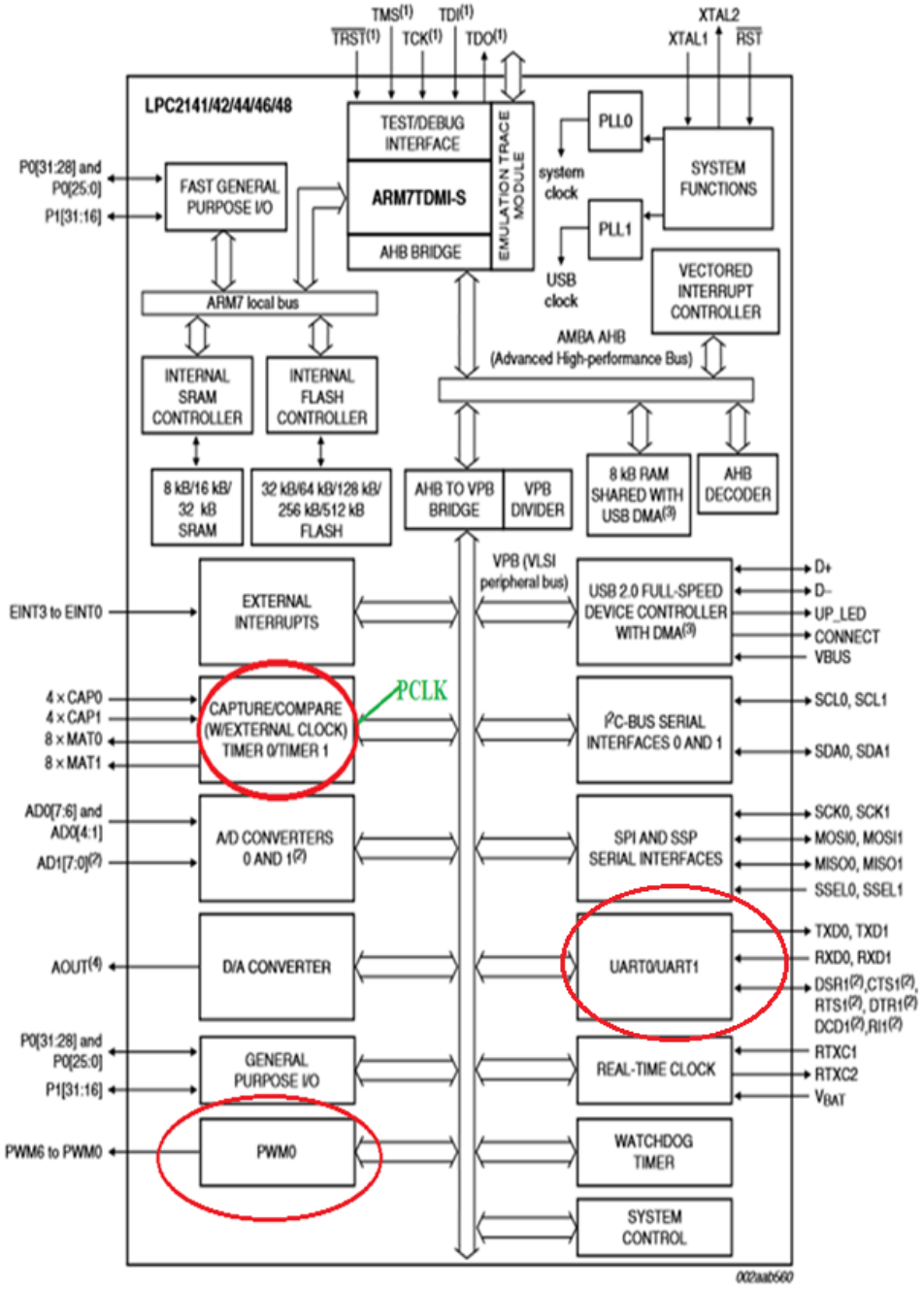
The asynchronous serial data format is shown below,



Here every character/ASCIIcode/8bit data is transmitted, with START and STOP bits. Optionally we can add parity bit for error checking during the transmission. Logic '0' represents the START. Logic '1' represents the STOP bit. When there is no transmission the line will be in the STOP logic.

There is no CLOCK signal used while transmitting/receiving the data. Hence both the transmitter and receiver should agree on the speed of data transmission (BAUD rate). Hence in Asynchronous serial communication, both the transmitter and receiver should work at the same baud rate.

Since, it provides, TxD and RxD, full duplex communication is supported. [UART also provides other handshaking signals like DSR,DTR,CTS,RTS,CD,RI, used while transmitting the data using the MODEM and telephone lines.]



Pin description

UART0 pin description

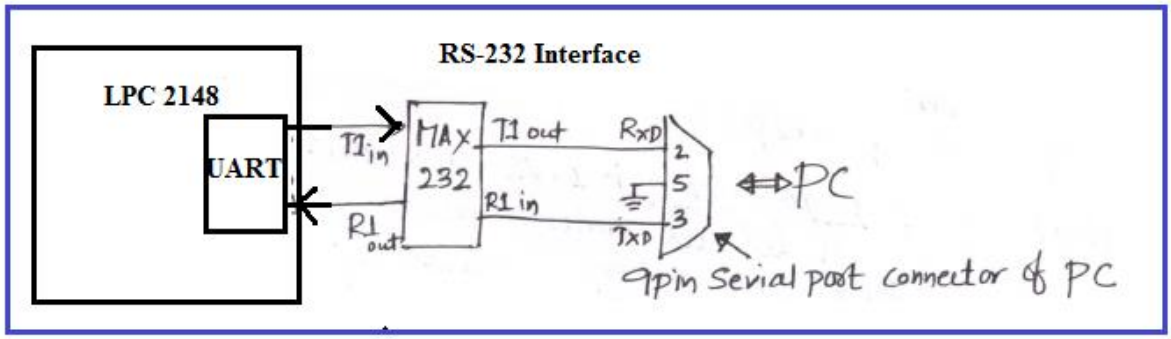
Pin	Type	Description
RXD0	Input	**Serial Input.** Serial receive data.
TXD0	Output	**Serial Output.** Serial transmit data.

RS 232 Standard and Interfacing LPC 2148 to PC

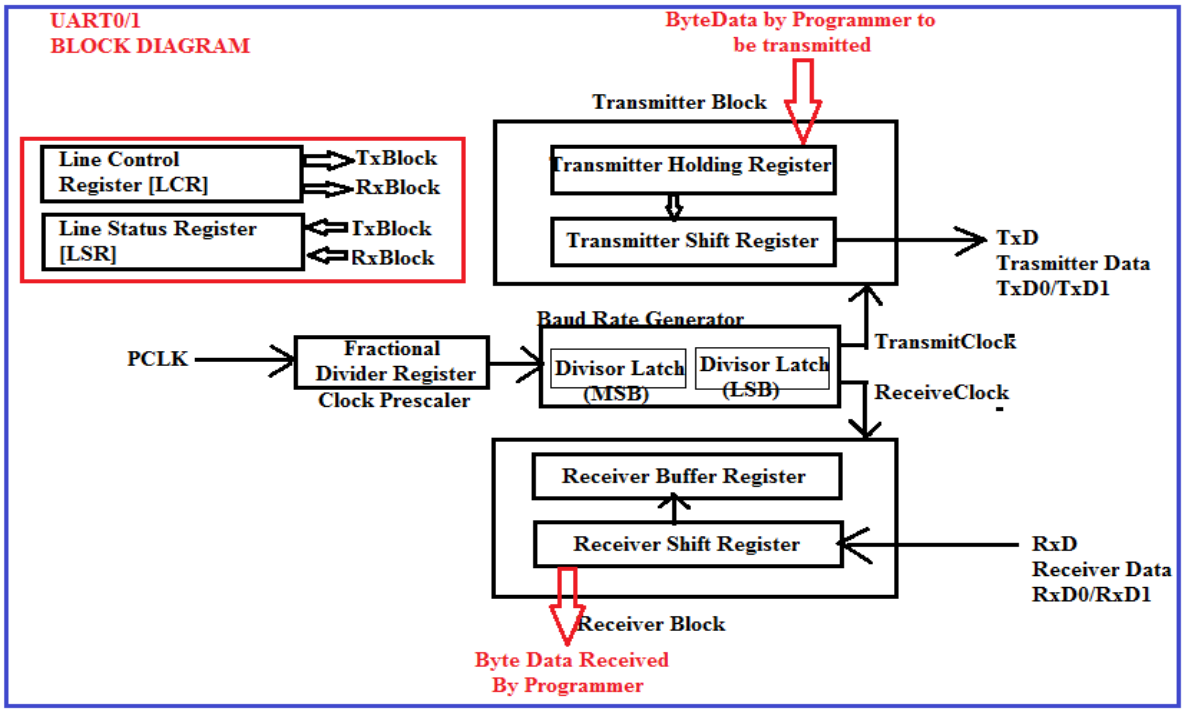
RS 232 is a popular industry accepted standard for asynchronous serial communication. IBM PC provides RS-232 compatible serial ports COM1/COM2.

RS 232 follows
-12V to +12V voltage levels to represent data
follow negative logic to represent the data
logic '1' -> -12V
logic '0' -> +12V
whereas, normal Microcontrollers use TTL voltage levels TTL,
logic '1' -> +3.3V/+5V
logic '0' -> 0V

LPC 2148 can be connected to RS-232 ports like PC's COM port by using RS–232 converters / translators (TTL to RS 232, vice versa), like MAX232



UART Architecture



Each of the UART provides, following blocks:

Transmitter Block: This block is accessible for the user in the form of the register TxTHR, where the user writes 8 bit data (Parallel). Then this data is converted into serial format by the shift register, TxTSR and sent on the line TxD.

Receiver Block: This block receives the serial data coming from external devices like PC on the RxD line. Then the receiver shift register, TxRSR converts the serial data to 8 bit parallel data and copies that to TxRBR. The programmer can read this register, to receive 8 bit data.

BaudRate Generation Unit: Baud Rate is defined as the rate at which serial data is transmitted or received on serial lines TxD and RxD. This is measured as bits per second. Most popular baud rates are 2400, 4800, 7200, 9600, 14400,19200, 28800, 38400,57600, 115200. The input clock to this block is from PCLK. Since this frequency is very high compared to the baud rate, it is required to further divide to realise the standard buadrate. Since baud rates are fixed, getting the proper baud rate requires, proper division/multiplication to arrive at the correct values. Fractional Divider Register and Divisor Latch(DLM:DLL) are provided to this job.

Control/Status Registers : Line Control Register, is provided to properly frame the serial data, as per the required format. This register decides the number of bits per character, number of stop bits, optional parity bit. Line Status Register is used to indicate the running condition of the serial port. Whether it has transmitted the data, or whether it has received any data or does it encounter any error while receiving the data.

UART Registers

Following are the important registers associated with UART, for basic programming. Additional registers are enclosed at the end of the notes for reference.

UxRBR : Receiver Buffer Register : The U0RBR/U1RBR contains the recently received data. The DLAB in LCR must be zero in order to access the RBR.

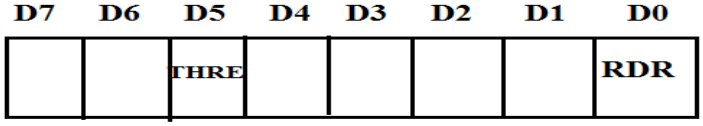
UxTHR: Transmit Holding Register : The U0THR/U1THR contains the data to be transmitted. The DLAB in LCR must be zero in order to access the RBR.

UxLCR: Line Control Register : U0LCR/U1LCR determines the format of the data character that is to be transmitted or received.

Line Control Register - Bit allocation

Bit	Symbol	Value	Description	Reset
1:0	Word Length Select	00	5 bit character length	0
		01	6 bit character length	
		10	7 bit character length	
		11	8 bit character length	
2	Stop Bit Select	0	1 stop bit.	0
		1	2 stop bits (1.5 if U0LCR[1:0]=00).	
3	Parity Enable	0	Disable parity generation and checking.	0
		1	Enable parity generation and checking.	
5:4	Parity Select	00	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.	0
		01	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.	
		10	Forced "1" stick parity.	
		11	Forced "0" stick parity.	
6	Break Control	0	Disable break transmission.	0
		1	Enable break transmission. Output pin UART0 TXD is forced to logic 0 when U0LCR[6] is active high.	
7	Divisor Latch Access Bit (DLAB)	0	Disable access to Divisor Latches.	0
		1	Enable access to Divisor Latches.	

UxLSR: Line Status Register: U0LSR/U1LSR provides status information on the UART0 TX and RX blocks. Here, two bits D0 and D5 are very important, which is explained in the programming section. Other bits are used to indicate error conditions, while receiving the data.



RDR - Receiver Data Ready
THRE - Transmitter Holding Register Empty

Line Status Register

Bit 0 – RDR: Receive Data Ready
This bit will be set when there is a received data in RBR register. This bit will be automatically cleared when RBR is empty.
0-- The UARTn receiver FIFO is empty.
1-- The UARTn receiver FIFO is not empty.

Bit 1 – OE: Overrun Error
The overrun error condition is set when the UART Rx FIFO is full and a new character is received. In this case, the UARTn RBR FIFO will not be overwritten and the character in the UARTn RSR will be lost.
0-- No overrun
1-- Buffer over run

Bit 2 – PE: Parity Error
This bit is set when the receiver detects a error in the Parity.
0-- No Parity Error
1-- Parity Error

Bit 3 – FE: Framing Error
This bit is set when there is error in the STOP bit(LOGIC 0)
0-- No Framing Error
1-- Framing Error

Bit 4 – BI: Break Interrupt
This bit is set when the RXDn is held in the spacing state (all zeroes) for one full character transmission
0-- No Break interrupt
1-- Break Interrupt detected.

Bit 5 – THRE: Transmitter Holding Register Empty
THRE is set immediately upon detection of an empty THR. It is automatically cleared when the THR is written.
0-- THR register is Empty
1-- THR has valid data to be transmitted

Bit 6 – TEMT: Transmitter Empty
TEMT is set when both UnTHR and UnTSR are empty; TEMT is cleared when any of them contain valid data.
0-- THR and/or the TSR contains valid data.
1-- THR and the TSR are empty.

Bit 7 – RXFE: Error in Rx FIFO
This bit is set when the received data is affected by Framing Error/Parity Error/Break Error.
0-- RBR contains no UARTn RX errors.
1-- RBR contains at least one RX error.

BaudRate Calculation..

LPC2148 generates the baud rate depending on the values of,

- 8 bit registers DLM, DLL (Divisor Latch MSB & LSB bytes), together represent Divisor Latch value.
- The value in the Fractional Divider Register, represents two 4bit values DivAddVal and MulVal. Useful in fine tuning the actual baud rate to the required baudrate

$$\text{BaudRate} = \frac{\text{PCLK}}{(16 * (\text{DLM:DLL}) * (1 + \text{DivAddVal}/\text{MulVal}))}$$

By disabling fraction divider, (taking default values on reset, DivVal=0,MulVal=1)

$$\text{BaudRate} = \frac{\text{PCLK}}{16 * (\text{DLM:DLL})}$$

$$\text{DLM:DLL} = \frac{\text{PCLK}}{16 * \text{BaudRate}} = \text{Result}$$

DLL = Lower 8 bits of Result
DLM = Upper 8 bits of Result

Ex: DLL = Result & 0xFF
DLM = (Result >> 8) & 0xFF

Configure UART for 115200 baud rate, Given, PCLK = 15MHz and Write a suitable program.

Calculation:

DLM:DLL = 15000000 / 16 * (115200) = 8

Hence, DLL = result % 256 = 8 (lower 8 bits of result)

DLM = result / 256 (upper 8 bits of result)

(or , use calculator, convert answer in decimal to HEX, take lower two digits put into DLL, next two digits to DLM)

```
void uart_init(void)

{
 //configurations to use serial port, UART0
 PINSEL0 |= 0x00000005;  // P0.0 & P0.1 ARE CONFIGURED AS TXD0 & RXD0
//programming the baud rate
 U0LCR = 0x83;   /* 8 bits, no Parity, 1 Stop bit    & DLAB = 1 */
 U0DLM = 0; U0DLL = 8; // 115200 baud rate, PCLK = 15MHz
 U0LCR = 0x03;  /* 8 bits, no Parity, 1 Stop bit    & DLAB = 0 */

}
```

Programming the UART / Serial Port

Steps for Configuring UART0 & Transmitting/Receiving data

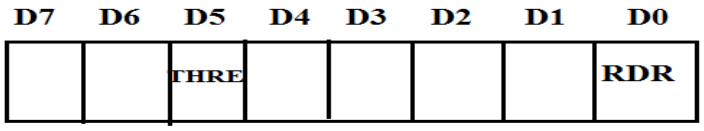
Below are the steps for configuring the UART0.

1. Configure the GPIO pin for UART0 function using PINSEL register.
2. Configure the FCR for enabling the FIFO and Reset both the Rx/Tx FIFO. (By default FIFO is disabled, without using this also the program works)
3. Configure LCR for 8-data bits, 1 Stop bit, Disable Parity and Enable DLAB.
4. Calculate the DLM,DLL values for required baudrate from PCLK.
5. Update the DLM,DLL with the calculated values.
6. Finally clear DLAB to disable the access to DLM,DLL.
7. Transmit / Receive the data

After this, the UART will be ready to Transmit/Receive Data at the specified baudrate. Use the following registers to receive/transmit the data.

UxRBR	Contains the recently received Data
UxTHR	Contains the data to be transmitted

Use UxLSR register, before transmitting next character or receiving a character from UART.



RDR - Receiver Data Ready
THRE - Transmitter Holding Register Empty

Line Status Register

Transmitting a character: While transmitting a character, look for D5 bit, if this bit '0', transmitter is still sending the previous character. Wait till this bit becomes '1', then we can write the next character to be transmitted into UxTHR register. The D5 bit is automatically cleared, when we write new data into THR.

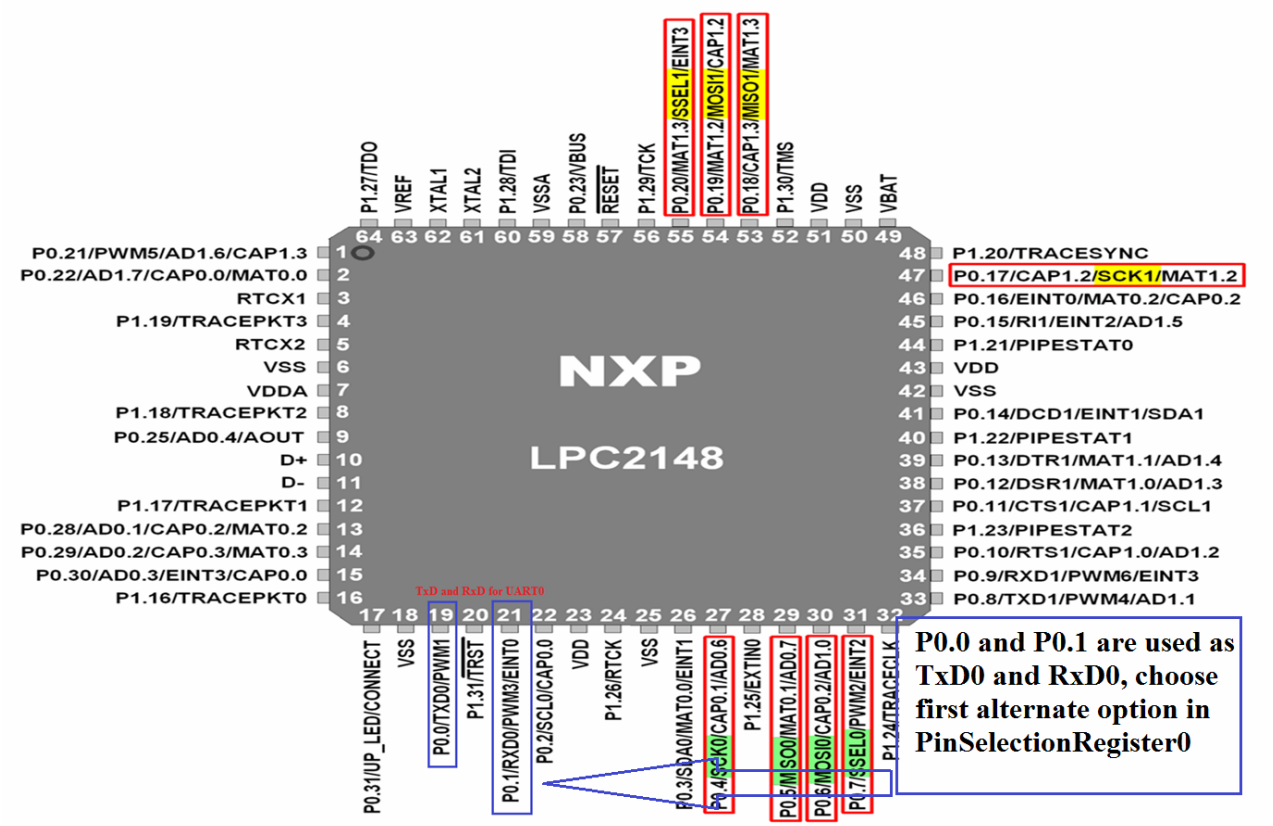
```
while((U0LSR & (0x01<<5))== 0x00);  // Wait for Previous transmission complete

U0THR  =  character to be transmitted
```

Receiving a character: While receiving a character, look for D0 bit, if this bit is '0', still UART has not received any data from PC/external device. When this bit becomes '1', UART has received the data and it is available in the register UxRBR. The D0 bit is auto cleared, when we read the data from RBR.

```
while((U0LSR & (0x01<<0))== 0x00){};  // wait till, a character (8bit) is received from PC

variable = U0RBR
```


NXP
LPC2148
P0.21/PWM5/AD1.6/CAP1.3
P0.22/AD1.7/CAP0.0/MAT0.0
RTCX1
P1.19/TRACEPKT3
RTCX2
VSS
VDDA
P1.18/TRACEPKT2
P0.25/AD0.4/AOUT
D+
D-
P1.17/TRACEPKT1
P0.28/AD0.1/CAP0.2/MAT0.2
P0.29/AD0.2/CAP0.3/MAT0.3
P0.30/AD0.3/EINT3/CAP0.0
P1.16/TRACEPKT0
P1.20/TRACESYNC
P0.17/CAP1.2/SCK1/MAT1.2
P0.16/EINT0/MAT0.2/CAP0.2
P0.15/RI1/EINT2/AD1.5
P1.21/PIPESTAT0
VDD
VSS
P0.14/DCD1/EINT1/SDA1
P1.22/PIPESTAT1
P0.13/DTR1/MAT1.1/AD1.4
P0.12/DSR1/MAT1.0/AD1.3
P0.11/CTS1/CAP1.1/SCL1
P1.23/PIPESTAT2
P0.10/RTS1/CAP1.0/AD1.2
P0.9/RXD1/PWM6/EINT3
P0.8/TXD1/PWM4/AD1.1
P1.27/TDO
VREF
XTAL1
XTAL2
P1.28/TDI
VSSA
P0.23/VBUS
RESET
P1.29/TCK
P0.20/MAT1.3/SSEL1/EINT3
P0.19/MAT1.2/MOSI1/CAP1.2
P0.18/CAP1.3/MISO1/MAT1.3
P1.30/TMS
VDD
VSS
VBAT
P0.31/UP_LED/CONNECT
VSS
P0.0/TXD0/PWM1
P1.31/TRST
P0.1/RXD0/PWM3/EINT0
P0.2/SCL0/CAP0.0
VDD
P1.26/RTCK
VSS
P0.3/SDA0/MAT0.0/EINT1
P0.4/SCK0/CAP0.1/AD0.6
P1.25/EXTIN0
P0.5/MISO0/MAT0.1/AD0.7
P0.6/MOSI0/CAP0.2/AD1.0
P0.7/SSEL0/PWM2/EINT2
P1.24/TRACECLK
TxD and RxD for UART0
P0.0 and P0.1 are used as TxD0 and RxD0, choose first alternate option in PinSelectionRegister0

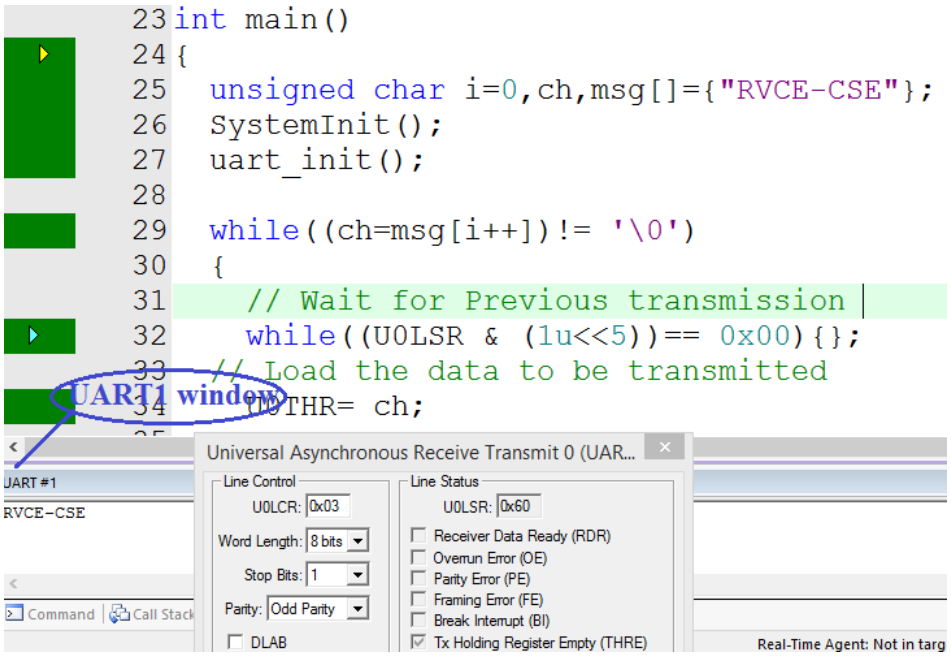
Write a Program to transmit a string from LPC 2148 to PC using Serial port / UART0, at the baud rate 115200. (Assume PCLK = 15MHz)

```
#include <lpc214x.h>

int main()
{       unsigned char i=0,ch,msg[]={"RVCE-CSE"};

//Configure P0.0/P0.1 as RX0 and TX0
        PINSEL0 = (1 << 0)|(1 << 2); //P0.0-TxD0,P0.1-RxD0,2ndOption:01
// Enable FIFO and reset Rx/Tx FIFO buffers
        U0FCR = (1<<0) | (1<<1) | (1<<2);
// 8bit data, 1Stop bit, No parity
        U0LCR = (0x03<<0) | (1<<7);
//PCLK = CCLK / 4, CCLK = 60MHz, for FOSC = 12MHz (default),Baud=115200
        U0DLL =  8;  // =97  for 9600 baud
        U0DLM = 0;
// Clear DLAB after setting DLL,DLM
        U0LCR = (0x03 << 0);
// Program to send the string
        while((ch=msg[i++])!= '\0')
        {
          while((U0LSR & (0x01<<5))== 0x00); // Wait for Previous transmission
          U0THR= ch;        //// Load the data to be transmitted
        }
        while(1);
}
```

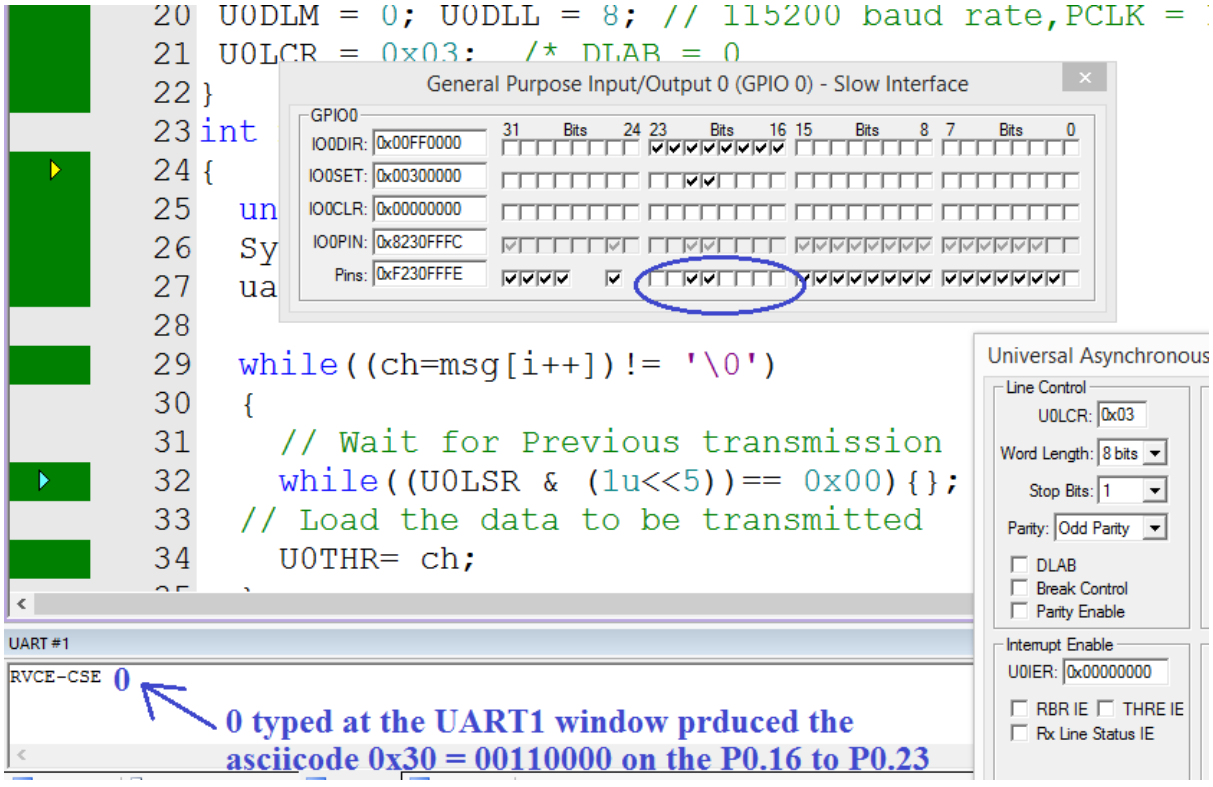
Note: In Keil you can see the output in serial window UART1, (choose peripherals windows)



Write a Program to receive a character from PC and output the 8 bit data on P0.16 to P0.23.

```
int main()
{
  unsigned char i=0;
  SystemInit();
  uart_init();

 IO0DIR = 0XFF << 16;  // configure P0.16 to P0.23 as output pins
  do
   {
    while((U0LSR & (0x01<<0))== 0x00){};  // wait till, a character (8bit) is received from PC
     i = UORBR;  // read from the  UART0
     IO0CLR = 0XFF << 16;IO0SET = U0RBR << i; // output the  i(8bit code) to P0.16 – P0.23
   }
  while(1);
}
```



Extra Information :

(Please refer LPC 2148 User manual for extra information)

UxFCR –FIFO Control Register: The LPC 2148 provides 16 byte Receive and Transmit FIFOs. The UxFCR controls the operation of the UARTx Rx and Rx FIFOs. Used to enable the Transmitter and Receiver FIFOs. By default the FIFOs disabled. By enabling the FIFO, CPU throughput time while doing serial data transmission increases. Using this register, we can clear the contents of FIFO.

UxIER – Interrupt Enable Register : Is used to enable interrupt sources in UART. Interrupt can be generated when the data received in UxRBR and interrupt is generated when UxTHR is empty. By default, interrupts are disabled.

UxIIR – Interrupt Identification Register: Used, by the ISR to identify the source of interrupt and the appropriate action can be initiated in the ISR.