UART

oduction to UART

UART → universal Asynchronous Receiver Transmitter.

→ UART was designed by an American engineer named Gordon Bell for Serial Communication.

:e for Both Short & long distance Serial Communication

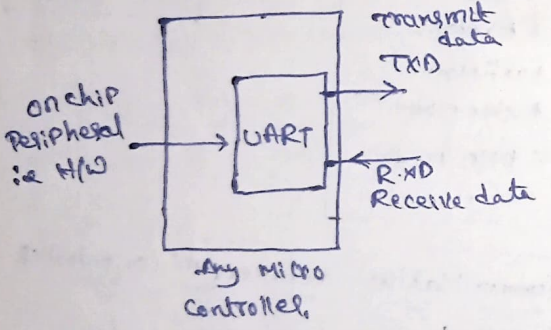
→ every micro controller comes with the inbuilt UART

What is UART :

UART is an hardware as well as protocol.

→ UART is Parallel-in, Serial out & Serial-in, Parallel out

Note : while transmitting data using UART it will acts as Parallel-in Serial out. And while receiving the data, UART will acts as Serial-in, Parallel out device



Protocol : * set of standard rules.

→ every microcontroller, UART should follow the standard rules while transmitting & receiving the data

* Rules are same for all Microcontroller

→ Registers (SFRs) are only difference, For different controllers.

But rules are same

→ there are some rules, while before sending (or) receiving the data. For that we need set (or) fix the communication speed for both receiver & transmitter that is called as Baud rate

Why UART is used / Designed?

→ UART is mainly designed for asynchronous serial communication.

Note : UART can do asynchronous as well as synchronous serial communication.

It is possible to synchronous serial Communication but we need to select synchronous Communication mode

Serial Communication vs Parallel Communication

→ Serial commu is a process of sending 1 bit at a time

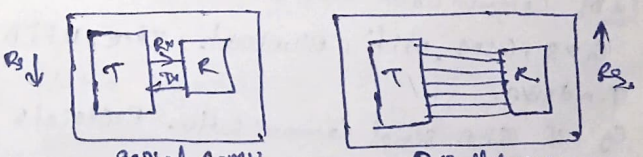
→ Parallel Commu is a process of sending 1 byte at a time

→ Parallel Communication is faster because 8 bits are transfering at a time of 1 msec

⇒ But serial only 1 bit is transfer in 1 msec so this serial commu is slow

→ By using Serial communication we can send data of transfer for longer distance & H/W Complex is less due to 2 wires,

→ By using Parallel Communication we can transfer data for smaller distances. due to H/W Complexity is high



Examples

Serial Communication (it is slower) is limited to these only

① Mouse

② Keyboard.

now a day there are several high end controllers to transfer data fastly

Parallel Communication

① Alpha LCD

② Old printers (with no. of wires)

For faster the serial communication there are high end controllers like - 1MBPS, 5MBPS, 10MBPS, 20MBPS.

using UART CAN, I2C, SPI

Different type of serial Communication

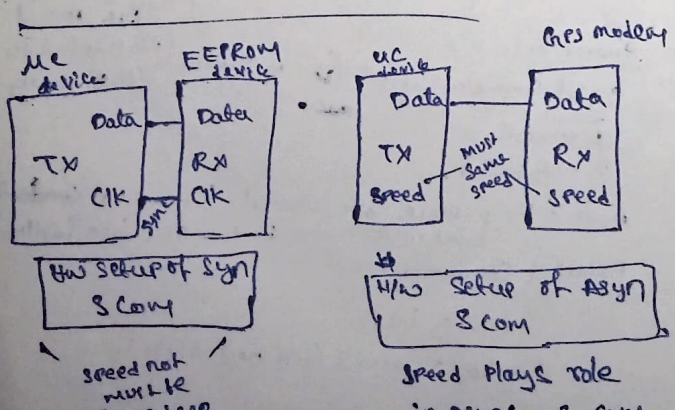
① Synchronous serial Communication
② Asynchronous serial Communication

Synchronous serial Communication

① There is a common clock line used b/w the TX & RX.
② TX & RX can synchronize with the help of clock pulses.
* If the TX & RX speed is not same, then also they can synchronize
③ example: (devices) EEPROM, RTC, Digi Accelerometer & thermometer etc...
④ eg of serial communication Protocal is: I2C, SPI etc.

Asynchronous serial Communication:

① there is no common clock line is used b/w TX & RX.
② the TX & RX speed must and should be same. otherwise TX & RX is can't synchronize. it leads to no data transfer b/w TX & RX.
③ Ex of Communication devices is: GPS, GSM, wifi, Bluetooth, xbee, RFID pendrive.
④ eg of asyn serial Communication Protocol: UART, USB, CAN, etc...



In Asynchronous serial Communication, Similar words of Speed = Boudrate = Frequency

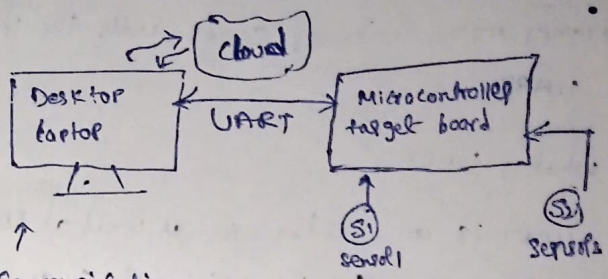
---

UART Protocal:

① Features of UART Protocal
② working of UART (Parallel-In serial out & serial-In, Parallel out).
③ UART Registers & Explanation

---

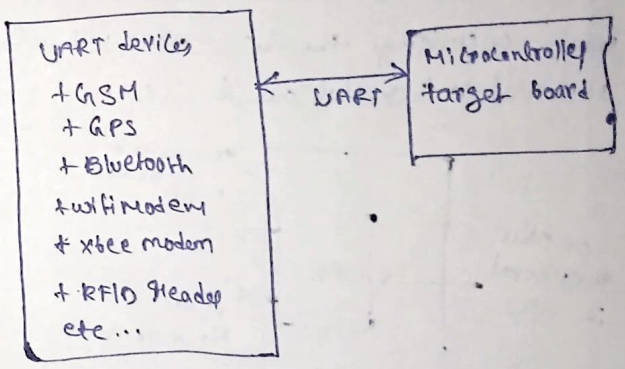
use/UART/Applications in real time.



① Communication b/w PC & Microcontroller board

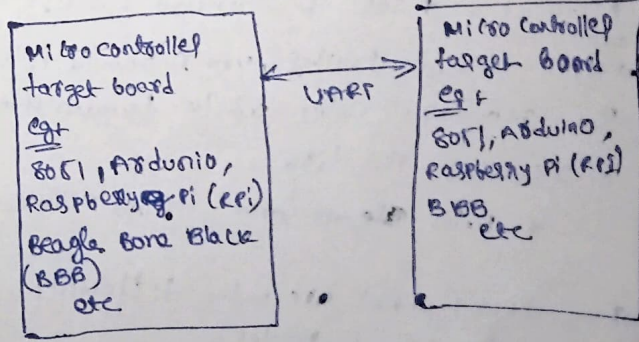
---

② Communication with external UART devices



---

③ Communication between microcontroller boards:



Can exchange the data from one to other microcontroller boards also communication is done b/w different

Eg: 8051 → 8051, 8051 → Arduino, Arduino → etc

* If we use different power supply, we should

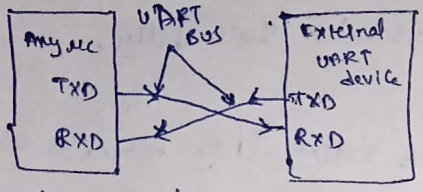
In UART we should follow fixed speed of baudrate to transfer data successfully

Features of UART Protocol :

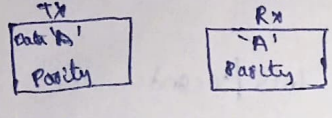
⇒ It is two wired communication protocol
ie - TXD (Transmitt data)
- RXD (Receive data).

⇒ Max speed of UART is upto 1 Mbps
this speed not for all uc. (bps = bits per second)

⇒ UART is designed for peer-to-peer communication. (only one device can communicate) one-to-one



⇒ Parity checking is the error detection mechanism used in UART



⇒ using UART we cannot communicate with multiple UART devices

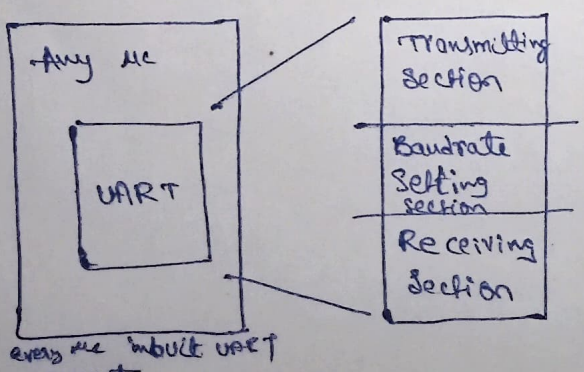
⇒ Data transfer direction is from LSB to MSB always (Protocol of one rule).

⇒ UART is Full duplex communication protocol.

⇒ UART can send only 1 byte of data at the same time
ex: 'ABCD'
↳ MSB ← LSB
0100 0001 → TXD
Bit by bit it will transmit serially.

working of UART (Parallel-in serial-out serial in-parallel out).

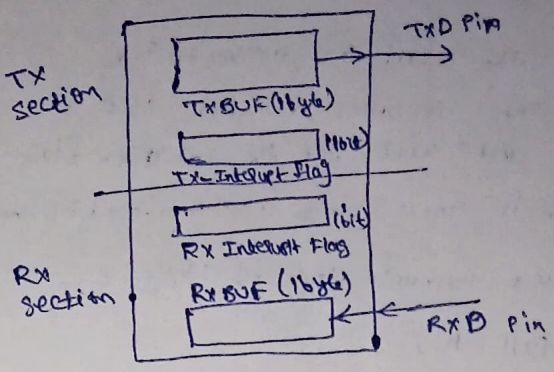
⇒ while sending a data UART will act as Parallel-In, Serial out & while receiving a data UART acts as serial-In, Parallel-out



every uc inbuilt UART presents

that three section do these own job individually.

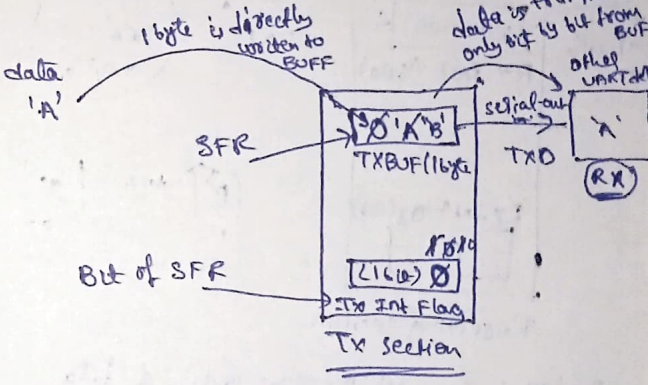
Transmitting section & Receiving section



upon reset all are zero.

all internal design is same for all uc.

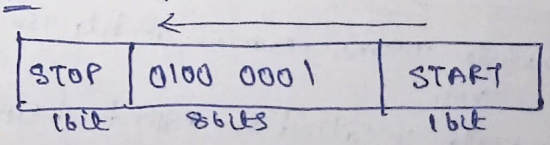
Transmiting section



TXBUF = 'A'; // this is the instruction to write the data UART.
(Parallel-IN)

Before transmiting the data the UART will create the data frame

ie : data = 'A'



standard data frame of UART
frame is also called as packet

Stall & stop bits are used for Synchronization purpose

By this Recievier know what is the start bit & stop last bit.

⇒ whenever start bit is transmitted Rx known

SUART has started transmitting the data.

⇒ But Rx don't Consider start bit as data bit, it's a just indication

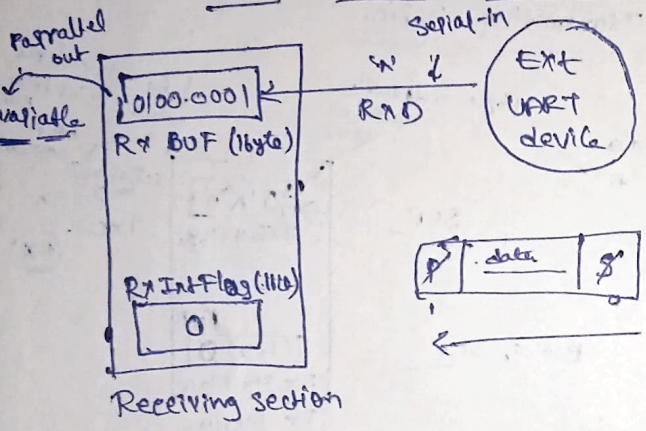
⇒ When the last bit arrives to Rx. next the UART transmit stop bit then UART will set the interrupt flag

⇒ then it know data is transfered successful.

Syntax : transmit code for 1 byte :

```
TXBUF = 'A';
while (TX Intr Flag == 0);
TX Intr Flag = 0;
```

Understanding receiving section of UART. (Serial-in & parallel-out logic)



Syntax : Code for Recieve 1byte of data

```
while (RX INTR Flag == 0);
Var = RXBUF;
Rx Intr Flag = 0; // to receive the next data byte
```

Baudrate setting section :

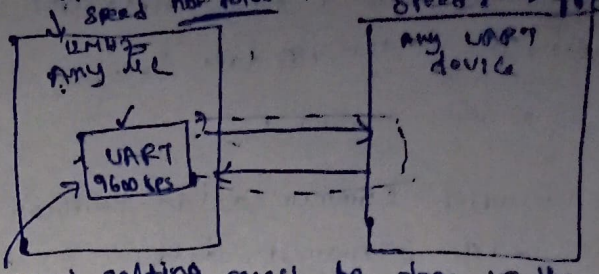
⇒ Baudrate means, number of bits transmited per second.

⇒ Baudrate is called as a speed of UART Communication.

Similar words

Baudrate == speed == Frequency == Rate of data transmission

setting the speed of UART is



speed setting must be done in the UART

⇒ defaut speed of every UART maximum is 9600 bps. units

if Both UART's speed then only the data transfer successfully.

---

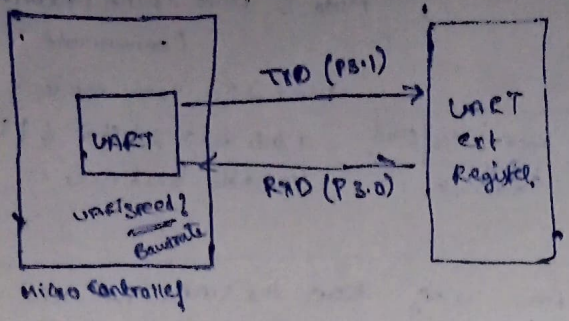
baud_setting_Reg = formula_to_set_Speed

this formula is not same for all MC

Baudrate = bits/second.

Bit rate = time taken to transmit a single bit

$f = \frac{1}{9600\,Hz}$ ← time taken to Tx single bit



TxBuf & RxBuf is generic name. that is actually UART registers.

But for 8051 :

→ UART registers
→ Basic Program to send & receive a data serially using UART
→ Baudrate setting & baudrate setting formula.

⟹

List of UART Registers :

	Register Name	Register Address.
UART	SBUF	0x99
	SCON	0x98
Timer registers	TMOD	0x89
	TCON	0x88
	TH1	0x8D

For Baudrate setting

TxBUF & RxBUF name Same in 8051 ie = SBUF

SBUF (8bits = 1byte).

→ SBUF → Serial Buffer.
→ SBUF is Transmit & Receive.
→ To transmit data serially, write into this reg. (SBUF)
→ Collected received data from the reg
→ Physically 2 registers are available in 8051 UART . actually 2 SBUF's available Physically in UART

That's we called as Full duplex. ie we can perform both operations ie Transmit & Recive. Like.

SBUF = SBUF. ← This is possible
TX RX

But Practically not possible to read Both

Both SBUF's address is same but the instruction opcode is different for Tx & Rx.

write
SBUF = data
TxBUF TX
write OPcode

Read
temp = SBUF
RxBUF
Read OPcode

write instruction opcode & Read opcode is different, Depends on instruction of Opcode CPU knows the operation to be done ie Reading or Writing

Read & write Can perform at same time.

ie SBUF = SBUF
Tx Rx
Syntax.

This instruct has different opcode.

Sending & Receiving data

```
SBUF = data ;
while (TxintFlg==0);
TxintFlag=0;
```

TX Using UART

```
unsigned char temp;
while (RxINTFlag==0);
RxINTFlag=0;
temp = SBUF
```

RX using UART

→ SCON register

• Serial Control register (SCON) 8bits

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

upon Reset value is 0

RI : Receive interrupt Flag
TI : Transmitter Interrupt Flag
RB8 : Received bit number 8
TB8 : Transmitted bit number 8
REN : Receive Enable
SMx : Serial mode select Bits

x means 0 → 2 : 0, 1, 2

The below 3 bits are used in multi-processor Communication mode ie : SM2, RB8, TB8.
only one multiprocessor mode is given in UART

But practically we can't do multi processor communication mode operation.

RI → 0 (data is not yet received into RxSBUF)

→ 1 (data is received into Rx SBUF)

name of Rx Interrupt Flag in 8051 is RI

the syntax is

```
while(RI==0);
RI=0;
```

TI → 0 (data is not transmitted)

→ 1 (data is transmitted from tx SBUF)

Transmit Interrupt Flag

Syntax:

```
SBUF = data
while (TI==0);
TI=0;
```

By default RxREN is disabled & vice versa.

REN → 0 (UART receiving is disabled)

→ 1 (UART receiving is enabled).

Upon reset REN=0.

⇒ By default transmitting section is enabled

⇒ If we want to receive any data from external world REN must be 1. Then only the UART receives data.

SM0	SM1	Mode	Name & Description
0	0	Mode 0	Name: Shift reg mode
		No frame for this mode. This mode is half-duplex.	Desc: It is Syn serial Comm mode),
0	1	Mode 1	Name: Universal mode
this mode is widely used		Desc: Asyn Scom mode	
It creates standard frame		8 bits data & 1 stop & 1 start variable baudrate mode	
1	0	Mode 2	Name: Multi-processor Comm mode.
			Desc: Asy Synchronous Scom mode
		9 bits, 1 start bit & 1 stop bit	
		Fixed baudrate mode	

STOP	X XXXX XXXX	START

9 data bits

1	1	Mode 3	Name: multi processor Comm mode
		Desc: Asyn Scom mode.	
		9 bits data & 1 Stop & 1 Start	
		variable baudrate.	

STOP	X XXXX XXXX	START

9 data bits.

Before using the UART, we need to write below values in SCON register

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
0	1	0	1	0	0	0	0

universal mode operation — Receiving section enabled — It is ready for receiving data

i.e SCON value = 0X50;

Baudrate Setting:

- Timer Mode Select (TMOD).

7	6	5	4	3	2	1	0
G	C/T	M1	M0	G	C/T	M1	M0

⇒ For baudrate setting, Timer 1 must be used in Mode 2 (TMOD |= 0X20;);

Formula to set the baudrate.

$$TH1 = 256 - \frac{2^{SMOD} * XTAL}{12 * 32 * baud}$$

TH1: This register is used to store integer value, which will set the UART baudrate.

⇒ In place baud we need to change the required baud value in above formula.

⇒ In place XTAL we need to change the required crystal osc frequency max: 11.0592 MHz

⇒ if SMOD == 1 : Double the baudrate

if SMOD == 0 : Do not double the baudrate

SMOD is the 7th bit of PCON register

If we want set the SMOD bit, we need to write like this

Mostly 8051 required as below

$$TH1 = 256 - \left[\frac{2^0 \times 11.0592\,MHZ}{12 \times 32 \times 9600.}\right]$$

$TH1 = 256 - 3$; ← this always in decimal value

We cannot store float value TH1 register

↓

If float value comes in solving baudrate then we say error in baud rate setting.

Baudrate Setting

Baudrate	TH1 Value
7200	252 (FA) -4
9600	253 (FD) -3
14400	254 (FE) -2
28800	255 (FF) -1
57600	255 (FF) -1 with PCON |= 0x80)

* 28800 is max value of baudrate setting in UART because TH1 register size is 8bits. So its range is upto 255 only.
* And is there another option increase the baudrate setting in UART. Yes.

57600. for this we need write 255 value TH1 & Initilize the 0x80 in PCON register by this SMOD bit is set.

ie 28800 * 2 = 57,600BPS ← this is the max baudrate in UART can set.

Note :→

UART initialization lastly if Syntax

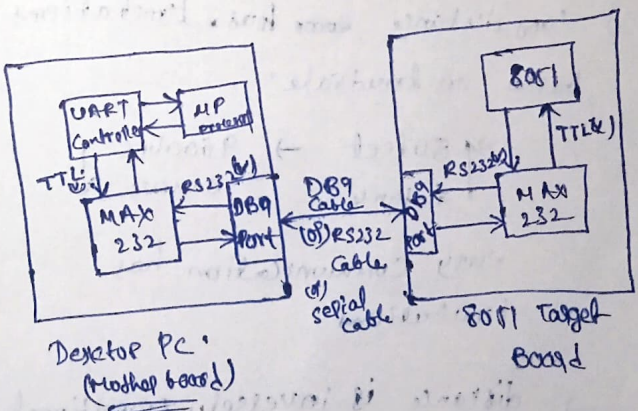
```
SCON = 0x50 // universal mode is selected.
       // REN = 1
TMOD |= 0x20 ; // timer 1 & mode 2
TH1 = 253 ; // for Baudrate 9600.
TR1 = 1; // Start baudrate setting in the
         6' UART.
```

* Basic Program to send & receive a data serially using UART :
* H/w Connection :
* Loop back Program.

Basic Program :

```
/* main_uart.c */
#include <reg51.h>
main(') {
 /* UART initialization */
SCON = 0x50;
TMOD |= 0x20;
TH1 = 253 ; // for B 9600
TR1 = 1; // Start Baudrate Setting.
 /* send a 1byte of data */
SBUF = 'A'; // 65 / 0x41.
while (TI == 0); // waiting for data T/F
 TI = 0;
 while(1); // This for one time transmison
              of 'A'. otherwise it will transmit
              continously 'A'.
}
```

UART hardware setup (for Long distance Communication)



Max 232 also inbuilt in every Micro Controller board.

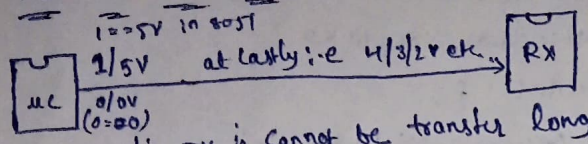
Process to see output of UART program :

select view → open-serial window → UART 1 :

DB9 port is used for easy connections only. Because no need to remember the pins of IC.

⇒ Max 232 is IC.
DB9 is the Port

Role of this MAX 232 is



This 5V is cannot be transfer long distance it will drop the voltage level for this reason we need to convert voltage into required manner.

i.e. microcontroller Voltage levels / or TTL Voltage levels

to

TTL Voltage levels.

⇒ For this reason 8051 provides MAX 232 IC to convert TTL voltages into RS232 Voltage levels & vice-versa.

* RS 232 voltage can transfer long distances. thats why we use MAX 232.

⇒ long distance also * Limitations based on baudrate

⇒ 50 feet → 9600 BPS
distance ↑ Baudrate ↓

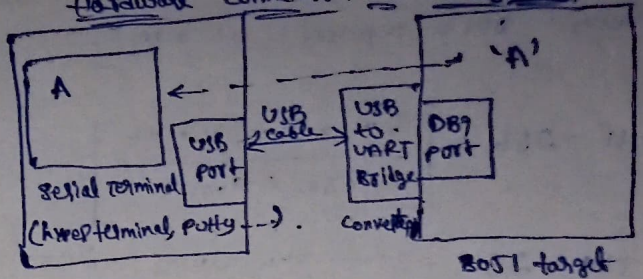
every Communication has Limitations

⇒ distance is inversely proportional to speed

Lastly : RS232 Voltage levels.

for Logic 0: +3V to +10V / +25V
Logic 1: -3V to -10V / -25V

This range is depends on µC. Board



⇒ Serial terminal :

→ It is the software which reads the serial port of the laptop or Desktop PC & it will print whatever received on the Console.

Serial terminals in windows OS :

* Hyper terminal (exclusively used for windows xp).
* Putty (Software Terminal) (626kb / 326kb).
↑
It is open sources S.W. used for windows
* Hercules serial terminal
* Flash magic --> Tools --> terminal
* etc ---

To know the comport Pin No. in system

Right click on My Computer
↳ select mange
↳ Device manger
↳ Ports.
See these port No. by

```
/* Uart_driver.c */
#include <reg51.h>
#include "header.h"
void uart_init (uint baud)
{
  SCON = 0x50;
  TMOD = 0x20;
  switch (baud)
  {
    case 2400 : TH1 = 0x52; break;
    case 9600 : TH1 = 0xFD; break;
    case 14400 : TH1 = 0x98; break;
    case 28800 : TH1 = 0xCC; break;
    case 57600 : TH1 = 0xFF; SCON = 0x52;
                 break;
    default : TH1 = 0xFD;
  }
  TR1 = 1; // start baudrate setting
}
void uart_tx (uint d)
{
  SBUF = d;
  while (TI == 0); // waiting for data transfer
  TI = 0;
}
```

← add all the header functions here i.e Rx & string function

```
/* main_uart.c */
#include "header.h"
main ()
{
  uint d;
  uart_init (9600);
  while (1)
  {
    for (i=0; i<256; i++)
    {
      uart_tx ('A' + i);
      delay (50ms);
    }
  }
}
```

add all the function prototypes in header.h file

← add at last of uart_driver.c

```
uint uart_rx (void)
{
  while (RI == 0); // wait for data to receive
  RI = 0;
  return SBUF;
}
void uart_rx_string (char *ptr)
{
  while (*ptr != '\0')
  {
    SBUF = *ptr;
    while (TI == 0);
    ptr++;
  }
}
```

Program for runtime test using uart.

```
/* main_uart.c */
#include "header.h"
main ()
{
  uint temp;
  uart_init (9600);
  while (1)
  {
    temp = uart_rx ();
    uart_tx (temp); // send back
  }
}
```

→ /* add last at driver.c for receive string function */

```
void uart_tx_string (char *buf, uint max_len)
{
  uint i;
  for (i=0; i<max_len; i++)
  {
    while (RI == 0);
    buf[i] = SBUF;
    if (buf[i] == '\0') // terminating character received
      break;
  }
  buf[i] = '\0'; // add null at last
}
```