LECTURE 20- UART

(UNIVERSAL ASYNCHRONOUS RECEIVER AND TRANSMITTER)

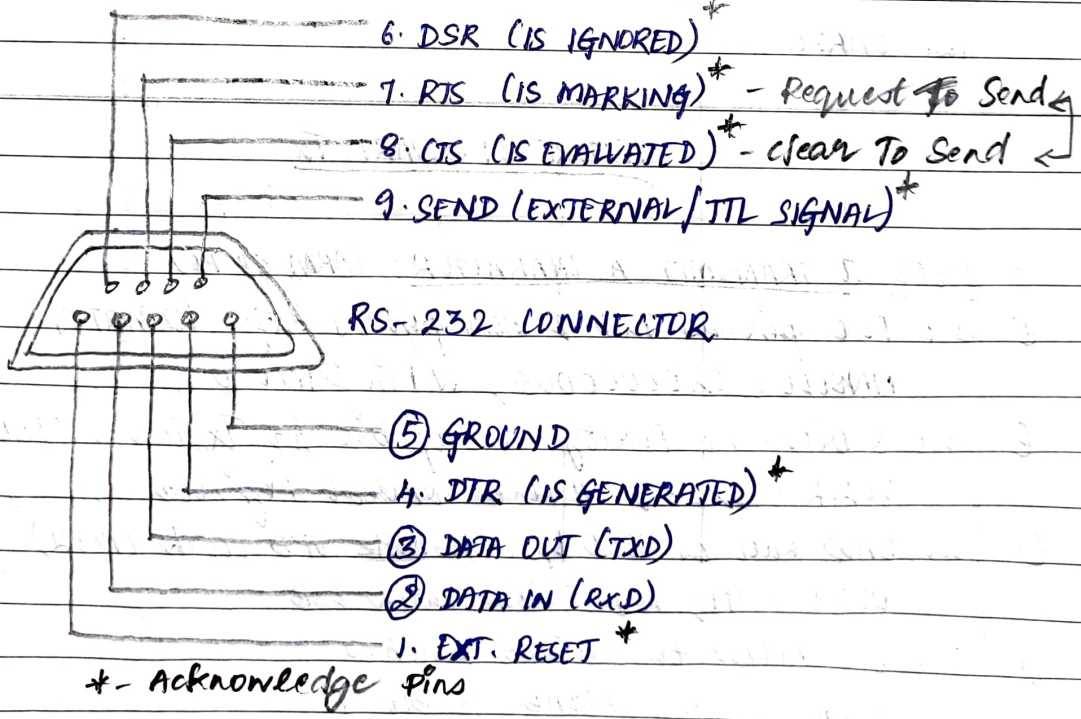
* ARM vs PC SERIAL COMMUNICATION : [speed - 9000 bits/second] → Each bit of data will be sent serially, one after another.
  ↳ PARALLEL COMMUNICATION [FASTER SPEED / MORE HARDWARE CONNECTIONS]

ARM	Max 232		PC	
Tx	T_{IN}	T_{OUT}	Rx	TRANSMISSION
Rx	R_{OUT}	R_{IN}	Tx	RECEPTION
Gnd	'Level Converter' [common ground by sharing]		Gnd	GROUND

TTL
Logic 1 - Vcc (3.3V)
Logic 0 - 0V

RS232
Logic 1 - -3V to -15V.
Logic 0 - +3V to +15V
[Error correction mechanism]

* DB9 CONNECTOR DETAILS : [Hyper Terminal Software - Virtual Terminal]



*- Acknowledge pins

Q. Why do we have a range of voltage level for RS232 as opposed to TTL?

A. The cable length between ARM and PC can be 10 m. Therefore, to reduce voltage drop in the data cable over this distance, we have such a range.

* **UART0 and UART1 :**

UART0		UART1	
P0.0/TXD0/PWM1	19	34	P0.9/RXD1/PWM6/EINT3
P1.31/TRST	20	33	P0.8/TXD1/PWM4
P0.1/RXD0/PWM3/EINT0	21		

UART0 can also be used for programming the microcontroller in boot load mode of operation. Boot loader is a small program that is already burnt in the microcontroller and instead of using an external programmer, program file is supplied through the serial port via the boot loader.

Boot loader will receive the data coming through the serial port and write to the actual flash memory of the microcontroller. For the boot loader to receive the file, we use UART0.

LECTURE 21 - UART Tx

* **STEPS TO TRANSMIT A CHARACTER: (ARM TO PC)**

① Set P0.0 pin as TxD & P0.1 pin as RxD - (USART0)
```
PINSEL0 = 0x00000005; // Pin Select 0
```
② Set 8 bits - no Parity - 1 Stop bit for Txsion & DLAB = 1
```
U0LCR = 0x83; // Line Control Register
```
③ Set BAUD Rate as 9600 bps - 15 MHz VPB clock (PCLK)
```
U0DLL = 97; // Division Latch LSB
```
④ Disable Access to Divisor Latches
```
U0LCR = 0x03; // DLAB = 0 Bit-7
```
⑤ Wait until UART0 ready to send character
```
while (!(U0LSR & 0x20)); // Line status Register - THRE
```
⑥ Sent a character to UART Transmit Register
```
U0THR = 'a'; // Transmit Holding Register
```

* STEP 1 - PINSEL0 = 0x00000005;

→ PINSEL0 Register -

Bit	Symbol	Value	Function	Reset Value
1:0	P0.0	00	GPIO Port 0.0	0
		01	TXD (UART0)	
		10	PWM1	
		11	Reserved	
3:2	P0.1	00	GPIO Port 0.1	0
		01	RxD (UART0)	
		10	PWM3	
17:16	P0.8	00	GPIO Port 0.8	0
		01	TXD UART1	
		10	PWM4	
		11	Reserved or AD1.1	
19:18	P0.9	00	GPIO Port 0.9	0
		01	RxD (UART1)	
		10	PWM6	
		11	EINT3	

```
PINSEL0 = 0x00000005; //UART0
PINSEL0 = 0x00050000; //UART1
```

* STEP 2 - U0LCR = 0x83;

→ Parameters - Serial Communication -

① Data length - Bits per Tx or Rx [8 bits]

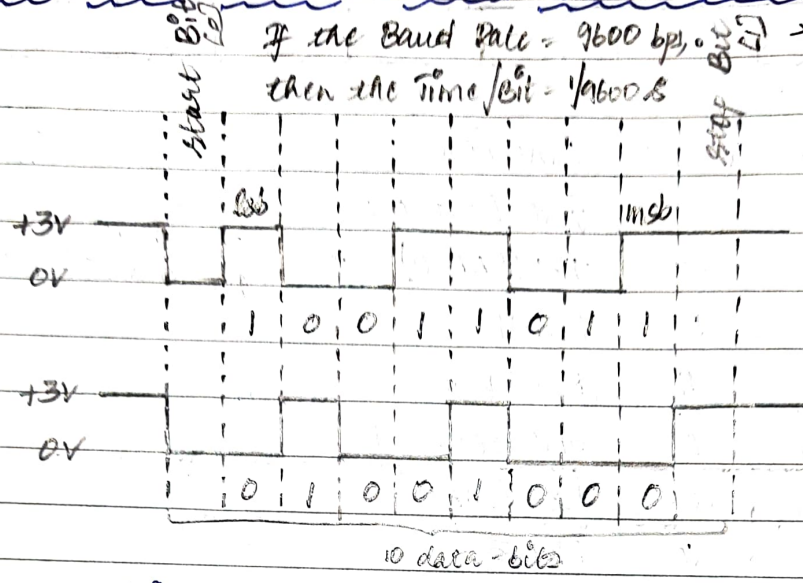
② Parity Bit - [Eg. 1110 - odd, 0101 - even] [error correction]

③ Stop Bits

④ Baud Rate - Number of bits Tx/Rx per second (bps) [9600 bps]

→ PTO

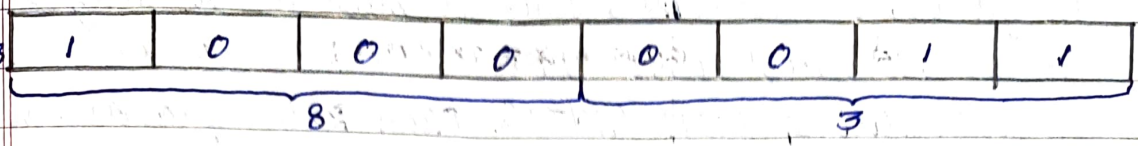
→ Data Frame Format - Serial Communication -



→ VOLER (Line Control Register) - (32-bit)

Bit	Symbol	Value	Description	Reset value
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 VOLER[1:0] = 00).	
3	Parity Enable	0	Disable parity generation & checking.	0
		1	Enable parity generation & 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 VOLER[6] is active high.	
7	Divisor Latch Access Bit (DLAB)	0	Disable access to Divisor Latches.	0
		1	Enable access to Divisor Latches.	

↓
Enable access to write in DLL/DLM



* STEP 3 - U0DLL = 97;

U0DLL → U0RTO Divisor latch LSB Bit

Table - UART0 Divisor Latch LSB register (U0DLL - address 0xE000 C000, when DLAB=1) bit description

Bit	Symbol	Description	Reset Value
7:0	DLL	The UART0 Divisor Latch LSB Register, along with the U0DLM register, determines the baud rate of the UART0.	0x01

U0DLM → UART0 Divisor latch MSB Bit

Table - UART0 Divisor Latch MSB register (U0DLM - address 0xE000 C004, when DLAB=1) bit description

Bit	Symbol	Description	Reset Value
7:0	DLM	The UART0 Divisor Latch MSB Register, along with the U0DLL register, determines the baud rate of the UART0	0x00

peripheral for generating the clock for Baud Rate

→ Baud Rate Calculation - Fractional Baud Rate Generator -

$$UART0_{baudrate} = \frac{PCLK}{16 \times (16 \times U0DLM + U0DLL)} \times \frac{MulVal}{(MulVal + DivAddVal)}$$

(Default value of U0DLL = 0x01)

→ Example 1 -

PCLK = 30 MHz and Required Baud Rate is 9600 bauds.

start with U0DLM = 0, DIVADDVAL = 0 and MULVAL = 1

We have PCLK = 30 MHz = 30×10^6 Hz

We get U0DLL = 195.3125, since it must be an integer, we use 195, i.e., U0DLL = 195.
On substituting U0DLL = 195 in Baud Rate equation, we get -
Baud Rate = 9615.38, Error = +15.98 from 9600.
Error - as low as possible by adjusting MULVAL and DIVADDVAL. Maximum value of MULVAL and DIVADDVAL is 15.

Substituting MULVAL = 15 and DIVADDVAL = 1, we get U0DLL = 183
Substituting U0DLL = 183 in Baud Rate equation, we get -
Baud Rate = 9605.53, i.e ~ 9605!

The final configuration is as follows -
PCLK = 30×10^6 Hz
U0DLL = 183
U0DLM = 0
MULVAL = 15
DIVADDVAL = 1

→ Example 2 -
PCLK = 60 MHz and Required Baud Rate is 9600 bauds.
start with U0DLM = 0, DIVADDVAL = 0 & MULVAL = 1 with PCLK = 60 $\times 10^6$ Hz.

We get U0DLL = 390.625 (i.e. ~390). But U0DLL is 8-bit!
So, we set U0DLM = 1 and find U0DLL. New U0DLL = 135
Substituting U0DLM and DLL in Baud Rate equation, we get -
Baud Rate = 9590.19, Error = -9.21

Substituting MULVAL = 15, we find that New U0DLL = 110
Substituting New U0DLL in Baud Rate equation, we find that -
Baud Rate = 9605.53 which is ~ 9605!

Hence, the final configuration is as follows-

PCLK = 60×10^6 Hz

UODLL = 110

UODLM = 1

MULVAL = 15

DIVADDVAL = 0

→ Example 3 -

$USART0_{baudrate}$ = 9600 bps

PCLK = 15 MHz

UODLM = 0

MulVal = 1

DivAddVal = 0

UODLL = ?

UODLL = 97;

* STEP 4 - U0LCR = 0x03; - Disable the access for Divisor Latch

U0LCR=0x03

0	0	0	0	0	0	1	1

0 3

* STEP 5 - while ((U0LSR & 0x20) == 0); checks whether transmit register is empty or not

It can also be written as while (!(U0LSR & 0x20));

→ U0LSR (Line status Register) -

Bit	Symbol	Value	Description	Reset Value
0	Receiver Data Ready		U0LSR0 is set when the U0RBR holds an unread character and is cleared when the UART0 RBR FIFO is empty.	0
	U0RBR	0	U0RBR is empty.	
		1	U0RBR contains valid data.	
1	Overrun		The overrun error condition is set	0

	Error (OE)		as soon as it occurs. An UOLSR read clears UOLSR1. UOLSR1 is set when UART0 RSR has a new character assembled and the UART0 RBR FIFO is full. In this case, the UART0 RBR FIFO will not be overwritten and the character in the UART0 RSR will be lost.
		0	overrun error status is inactive.
		1	overrun error status is active.
2	Parity Error (PE)		When the parity bit of a received character is in the wrong state, a parity error occurs. An UOLSR read clears UOLSR[2]. Time of parity error detection is dependent on UOFCR[0]. Note - A parity error is associated with the character at the top of the UART0 RBR FIFO.
		0	Parity error status is inactive.
		1	Parity error status is active.
5	Transmitter Holding Register Empty (THRE)		THRE is set immediately upon detection of an empty UART0 THR and is cleared on a U0THR write.
		0	U0THR contains valid data
		1	U0THR is empty.

UOLSR @ 0x20	0	0	THRE	0	0	0	0	0
			2/0			0		

* STEP 6 - U0THR = 'a';

shadow register

ASCII value of character will be sent to parallel-to-serial shift register. Based on clock given, character will be sent out serially through the Tx pin.