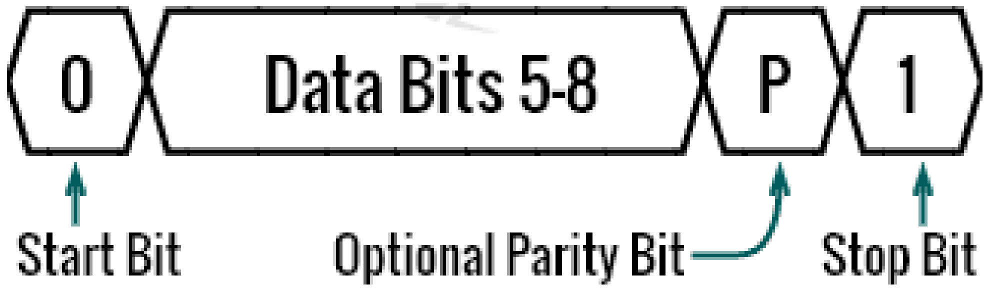
Serial Communication

UART(Universal Asynchronous Receiver/Transceiver uses TxD(Transmit) Pin for sending Data and RxD(Receive) Pin to get data. UART sends & receives data in form of chunks or packets. These chunks or packets are also referred to as 'Frames'.

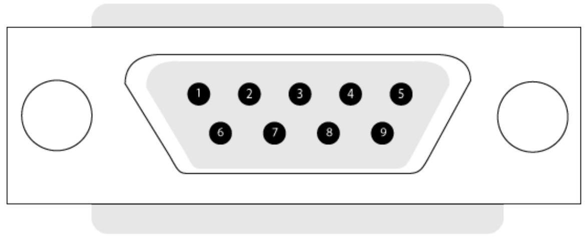


Pins:	**TxD**	**RxD**
UART0	P0.2	P0.3
UART1	P0.15/P2.0	P0.16/P2.1
UART2	P0.10/P2.8	P0.11/P2.9
UART3	P0.0/P0.25/P4.28	P0.1/P0.26/P4.29

Serial Communication

The microcontroller has an inbuilt UART for carrying out serial communication. The serial communication is done in the asynchronous mode.
IBM introduced the **DB-9 RS-232** version of serial I/O standard, which is most widely used in PCs and several devices. In RS232, high and low bits are represented by flowing voltage ranges:

Bit	**Voltage Range (in V)**	
0	+3	+25
1	-25	-3



The range -3V to +3V is undefined.

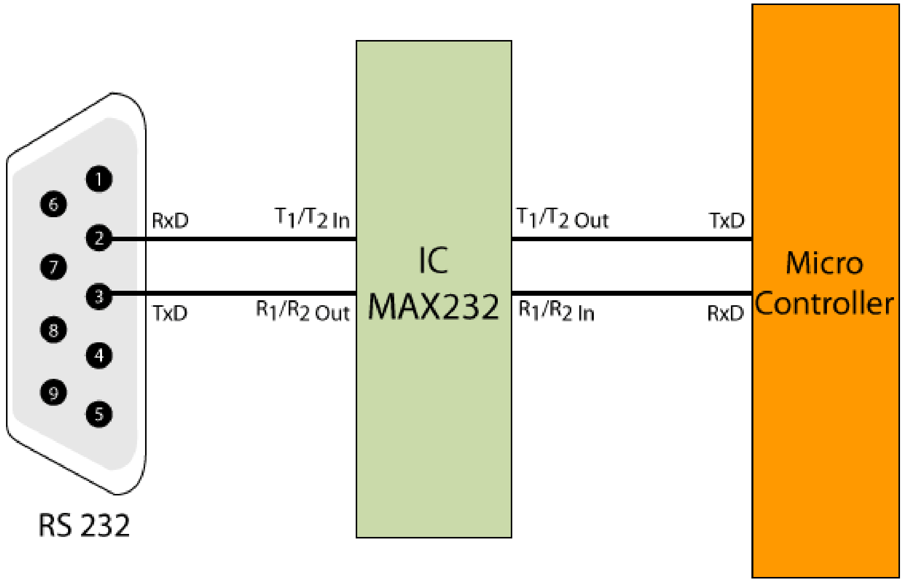
The TTL standards came a long time after the RS232 standard was set. Due to this reason RS232 voltage levels are not compatible with TTL logic.

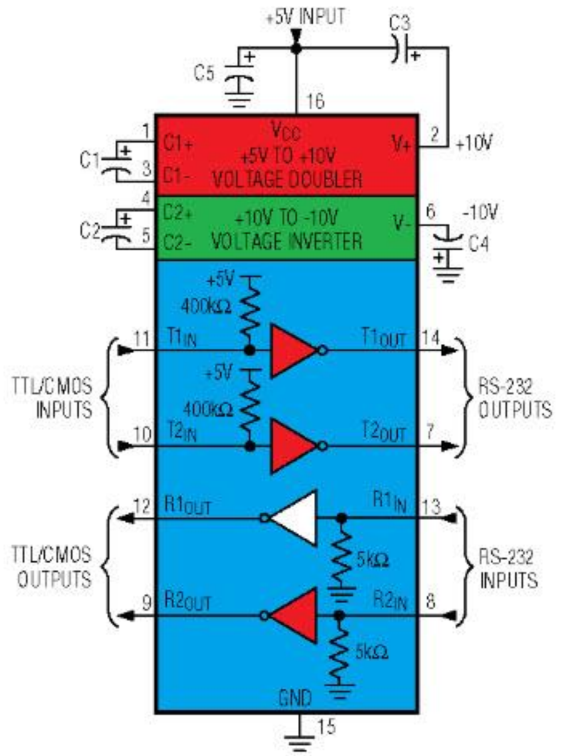
Therefore, while connecting an RS232 to microcontroller system, a voltage converter is required. This converter converts the microcontroller output level to the RS232 voltage levels, and vice versa. IC MAX232, also known as line driver, is very commonly used for this purpose.

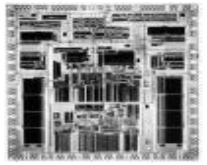
Serial Communication

TxD pin of serial port connects to RxD pin of controller via MAX232. And similarly, RxD pin of serial port connects to the TxD pin of controller through MAX232.

MAX232 has two sets of **line drivers** for transferring and receiving data. The line drivers used for transmission are called T1 and T2, where as the line drivers for receiver are designated as R1 and R2. The connection of MAX232 with computer and the controller is shown in the circuit diagram.







Serial Communication

UARTn Transmit Holding Register (U0THR)

The UnTHR is the top byte of the UARTn TX FIFO. The top byte is the newest character in the TX FIFO and can be written via the bus interface. The Divisor Latch Access Bit (DLAB) in UnLCR must be zero in order to access the UnTHR
UnTHR.

Bit	Symbol	Description
7:0	THR	Writing to the UARTn Transmit Holding Register causes the data to be stored in the UARTn transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.
31:8	-	Reserved, user software should not write ones to reserved bits.

UARTn Receiver Buffer Register (U0RBR)

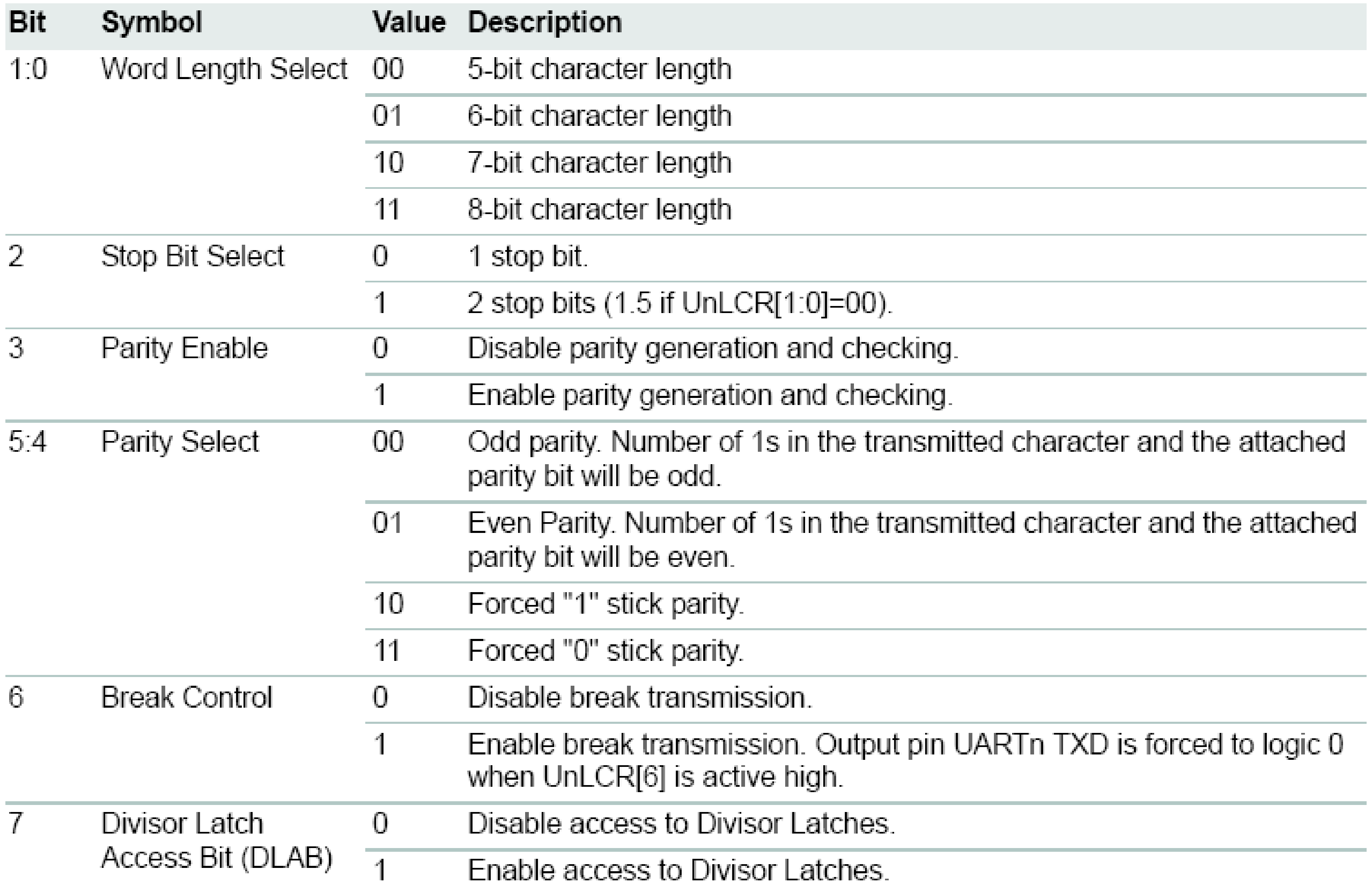
The UnRBR is the top byte of the UARTn Rx FIFO. The top byte of the Rx FIFO contains the oldest character received and can be read via the bus interface. The Divisor Latch Access Bit (DLAB) in LCR must be zero in order to access the UnRBR

Bit	Symbol	Description
7:0	RBR	The UARTn Receiver Buffer Register contains the oldest received byte in the UARTn Rx FIFO.
31:8	-	Reserved, the value read from a reserved bit is not defined.

Serial Communication

UARTn Line Control Register (U0LCR)

The UnLCR determines the format of the data character that is to be transmitted or received.



Bit	Symbol	Value	Description
1:0	Word Length Select	00	5-bit character length
		01	6-bit character length
		10	7-bit character length
		11	8-bit character length
2	Stop Bit Select	0	1 stop bit.
		1	2 stop bits (1.5 if UnLCR[1:0]=00).
3	Parity Enable	0	Disable parity generation and checking.
		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.
		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.
		1	Enable break transmission. Output pin UARTn TXD is forced to logic 0 when UnLCR[6] is active high.
7	Divisor Latch Access Bit (DLAB)	0	Disable access to Divisor Latches.
		1	Enable access to Divisor Latches.

Serial Communication

UARTn Line Status Register (U0LSR)

The UnLSR is a read-only register that provides status information on the UARTn TX and RX blocks

LSR								
31:8	7	6	5	4	3	2	1	0
Reserved	RXFE	TEMT	THRE	BI	FE	PE	OE	RDR

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

Serial Communication

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

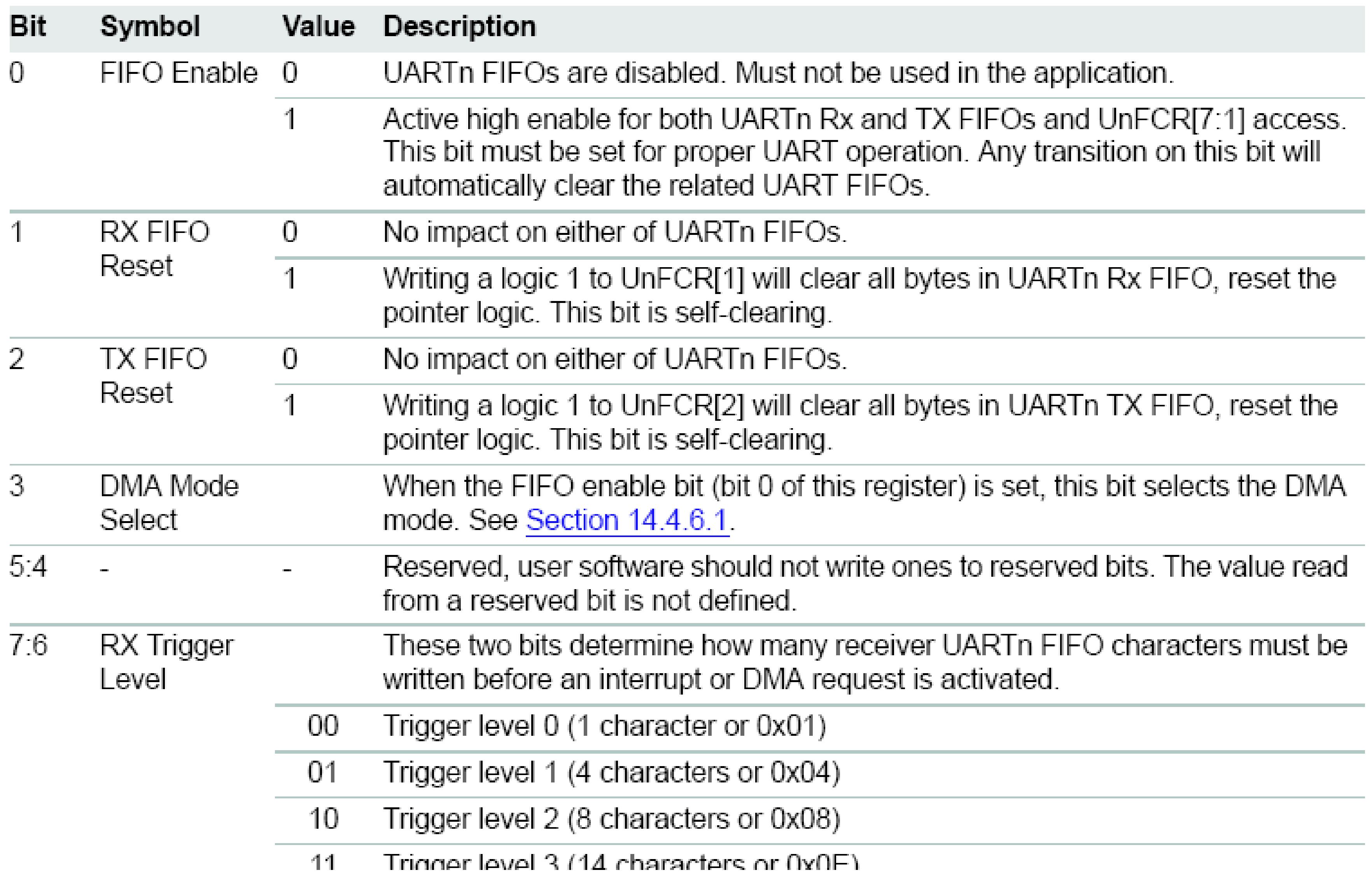
This bit is set when the received data is affected by Framing Error/Parity Error/Break Error.

0-- RBR contains no UARTn RX errors.

1-- RBR contains at least one RX error.

Serial Communication

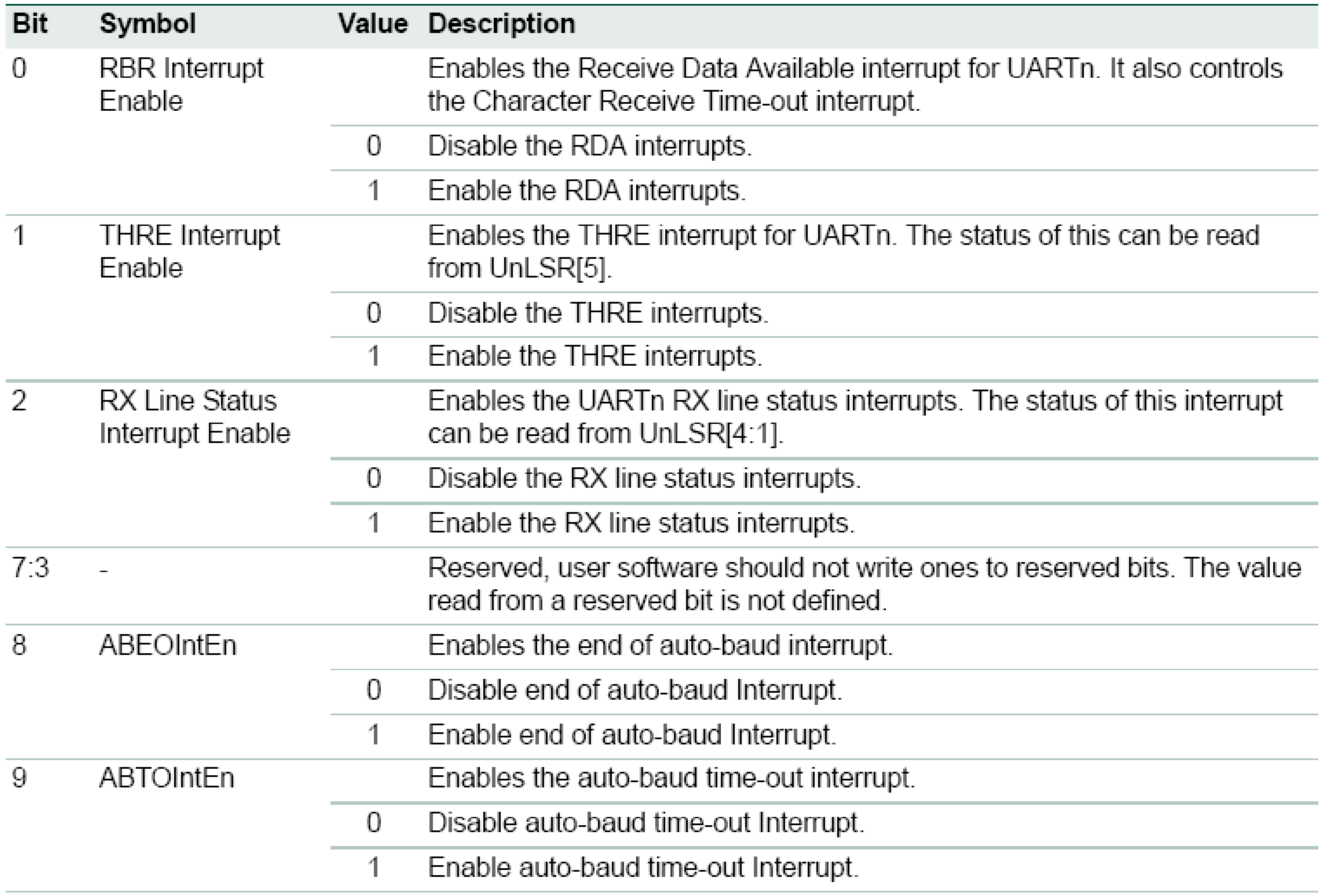
UARTn FIFO Control Register (U0FCR)



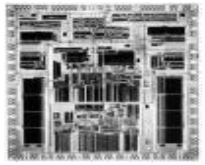
Bit	Symbol	Value	Description
0	FIFO Enable	0	UARTn FIFOs are disabled. Must not be used in the application.
		1	Active high enable for both UARTn Rx and TX FIFOs and UnFCR[7:1] access. This bit must be set for proper UART operation. Any transition on this bit will automatically clear the related UART FIFOs.
1	RX FIFO Reset	0	No impact on either of UARTn FIFOs.
		1	Writing a logic 1 to UnFCR[1] will clear all bytes in UARTn Rx FIFO, reset the pointer logic. This bit is self-clearing.
2	TX FIFO Reset	0	No impact on either of UARTn FIFOs.
		1	Writing a logic 1 to UnFCR[2] will clear all bytes in UARTn TX FIFO, reset the pointer logic. This bit is self-clearing.
3	DMA Mode Select		When the FIFO enable bit (bit 0 of this register) is set, this bit selects the DMA mode. See Section 14.4.6.1.
5:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
7:6	RX Trigger Level		These two bits determine how many receiver UARTn FIFO characters must be written before an interrupt or DMA request is activated.
		00	Trigger level 0 (1 character or 0x01)
		01	Trigger level 1 (4 characters or 0x04)
		10	Trigger level 2 (8 characters or 0x08)
		11	Trigger level 3 (14 characters or 0x0E)

Serial Communication

UARTn Interrupt Enable Register (U0IER)

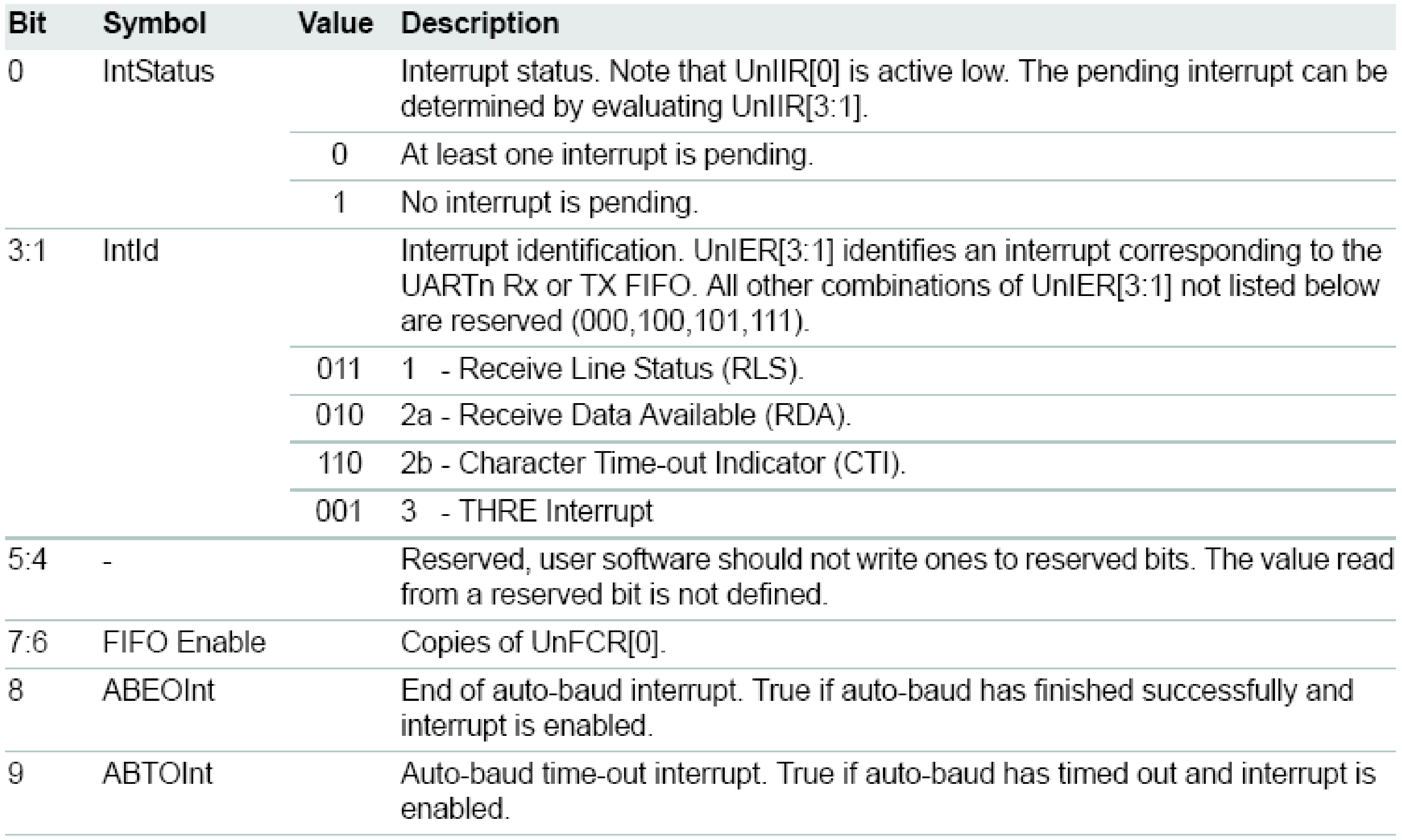


Bit	Symbol	Value	Description
0	RBR Interrupt Enable		Enables the Receive Data Available interrupt for UARTn. It also controls the Character Receive Time-out interrupt.
		0	Disable the RDA interrupts.
		1	Enable the RDA interrupts.
1	THRE Interrupt Enable		Enables the THRE interrupt for UARTn. The status of this can be read from UnLSR[5].
		0	Disable the THRE interrupts.
		1	Enable the THRE interrupts.
2	RX Line Status Interrupt Enable		Enables the UARTn RX line status interrupts. The status of this interrupt can be read from UnLSR[4:1].
		0	Disable the RX line status interrupts.
		1	Enable the RX line status interrupts.
7:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
8	ABEOIntEn		Enables the end of auto-baud interrupt.
		0	Disable end of auto-baud Interrupt.
		1	Enable end of auto-baud Interrupt.
9	ABTOIntEn		Enables the auto-baud time-out interrupt.
		0	Disable auto-baud time-out Interrupt.
		1	Enable auto-baud time-out Interrupt.

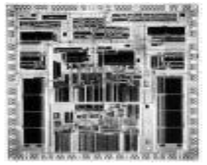


Serial Communication

UARTn Interrupt Identification Register (U0IIR)



Bit	Symbol	Value	Description
0	IntStatus		Interrupt status. Note that UnIIR[0] is active low. The pending interrupt can be determined by evaluating UnIIR[3:1].
		0	At least one interrupt is pending.
		1	No interrupt is pending.
3:1	IntId		Interrupt identification. UnIER[3:1] identifies an interrupt corresponding to the UARTn Rx or TX FIFO. All other combinations of UnIER[3:1] not listed below are reserved (000,100,101,111).
		011	1 - Receive Line Status (RLS).
		010	2a - Receive Data Available (RDA).
		110	2b - Character Time-out Indicator (CTI).
		001	3 - THRE Interrupt
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
7:6	FIFO Enable		Copies of UnFCR[0].
8	ABEOInt		End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.
9	ABTOInt		Auto-baud time-out interrupt. True if auto-baud has timed out and interrupt is enabled.

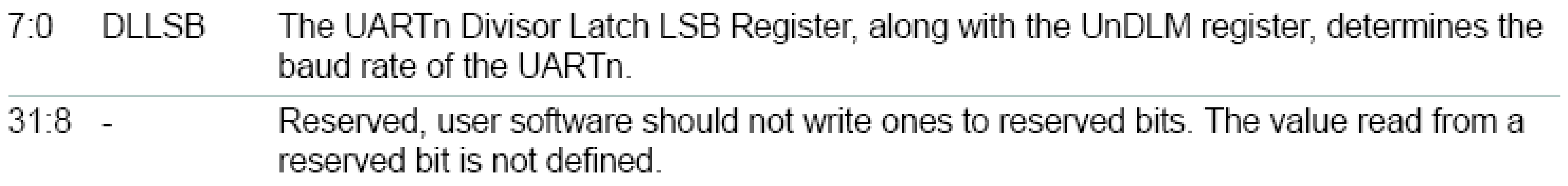


Serial Communication

UARTn Divisor Latch

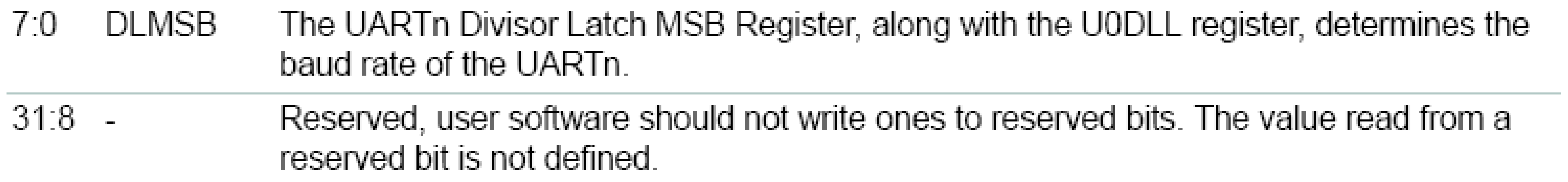
The UARTn Divisor Latch is part of the UARTn Baud Rate Generator and holds the value used, along with the Fractional Divider, to divide the PCLK in order to produce the baud rate clock.
The UnDLL and UnDLM registers together form a 16-bit divisor where UnDLL contains the lower 8 bits of the divisor and UnDLM contains the higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed.
The Divisor Latch Access Bit (DLAB) in UnLCR must be 1 in order to access the UARTn Divisor Latches.

UARTn Divisor Latch LSB register (U0DLL)



7:0	DLLSB	The UARTn Divisor Latch LSB Register, along with the UnDLM register, determines the baud rate of the UARTn.
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

UARTn Divisor Latch MSB register (U0DLM)



7:0	DLMSB	The UARTn Divisor Latch MSB Register, along with the U0DLL register, determines the baud rate of the UARTn.
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

Serial Communication

UARTn Fractional Divider Register (U0FDR)

Fractional Divider Register (U0/2/3FDR) controls the clock pre-scaler for the baud rate generation

Bit	Function	Value	Description
3:0	DIVADDVAL	0	Baud-rate generation pre-scaler divisor value. If this field is 0, fractional baud-rate generator will not impact the UARTn baudrate.
7:4	MULVAL	1	Baud-rate pre-scaler multiplier value. This field must be greater or equal 1 for UARTn to operate properly, regardless of whether the fractional baud-rate generator is used or not.
31:8	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

$$UARTn_{baudrate} = \frac{PCLK}{16 \times (256 \times UnDLM + UnDLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

Serial Communication

Program to receive a character and send it back

```
#include<LPC17xx.h>
unsigned char rx0_flag=0, tx0_flag=0;
int main(void)
{
        UART0_Init();
        while(1)
        {
                while(rx0_flag == 0x00);
                rx0_flag = 0x00;
                LPC_UART0->THR = recv_data;
                while(tx0_flag == 0x00);
                tx0_flag = 0x00;
         }
}
void UART0_Init(void)
{
        LPC_PINCON->PINSEL0 |= 0x00000050; //P0.2, TxD0(Fn 01) and P0.3 , RxD0 (Fn 01)
        LPC_UART0->LCR = 0x00000083;      //enable divisor latch, parity disable, 1 stop bit, 8bit word length
        LPC_UART0->DLM = 0X00;
        LPC_UART0->DLL = 0x13;            //select baud rate 9600 bps
        LPC_UART0->LCR = 0X00000003;
        LPC_UART0->FCR = 0x07; //FIFO enable, Reset
        LPC_UART0->IER = 0X03;                    //select Transmit and receive interrupt
        NVIC_EnableIRQ(UART0_IRQn);               //Assigning channel
}
```

Serial Communication

Program to receive a character and send it back

```
void UART0_IRQHandler(void)
{
        unsigned long Int_Stat;
        Int_Stat = LPC_UART0->IIR;              //reading the data from interrupt identification register
        Int_Stat = Int_Stat & 0x06;             //masking other than Transmit& Receive data indicator

        if((Int_Stat & 0x02)== 0x02)   //transmit interrupt
                tx0_flag = 0xff;

        else if( (Int_Stat & 0x04) == 0x04)  //receive  data available
        {
                recv_data = LPC_UART0->RBR;
                rx0_flag = 0xff;
        }
}
```