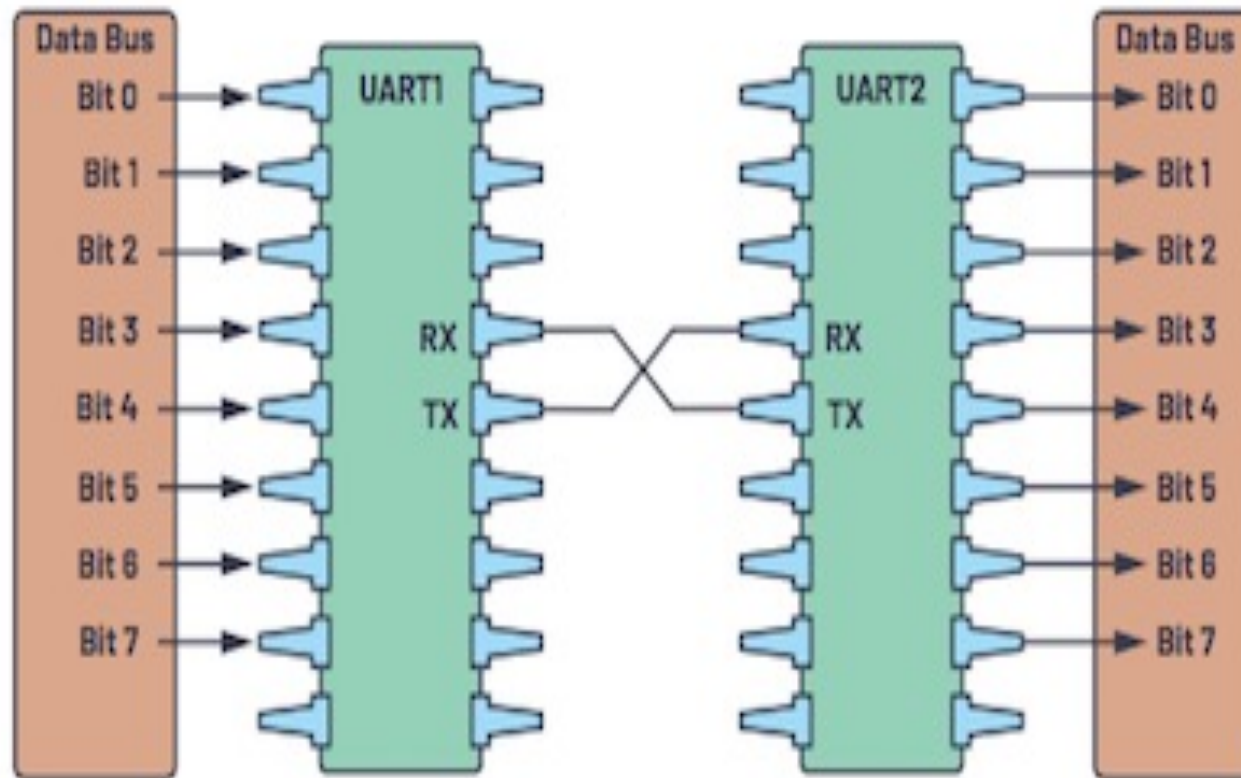


UART PROTOCOL

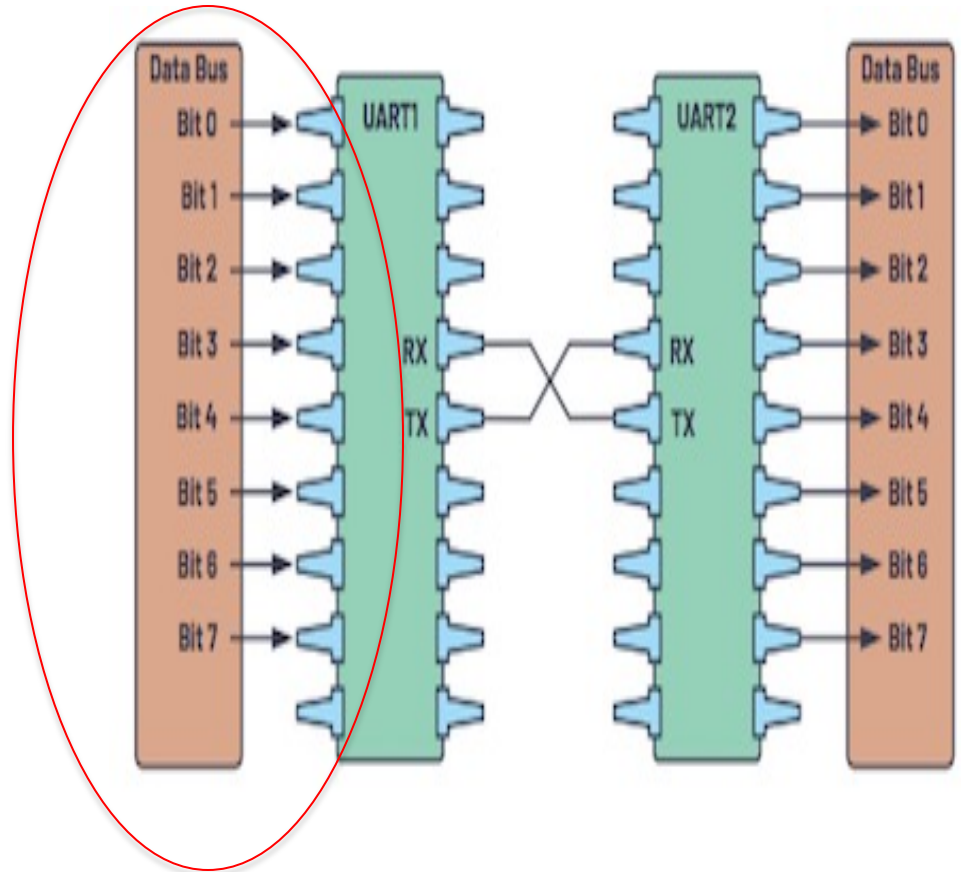
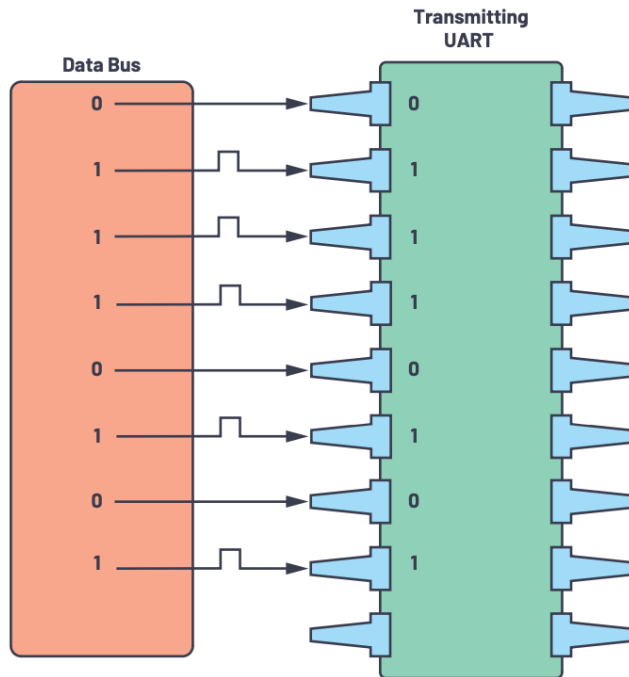
Transmission and reception of Data



Data to be transmitted is the following 8 bits: 0 1 1 1 0 1 0 1

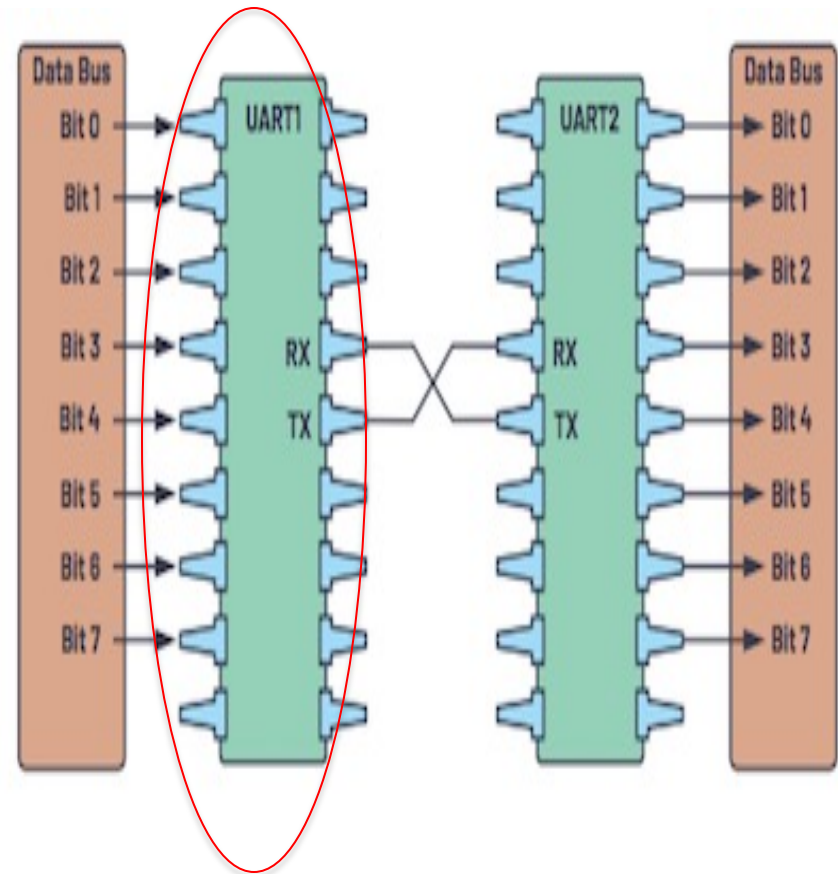
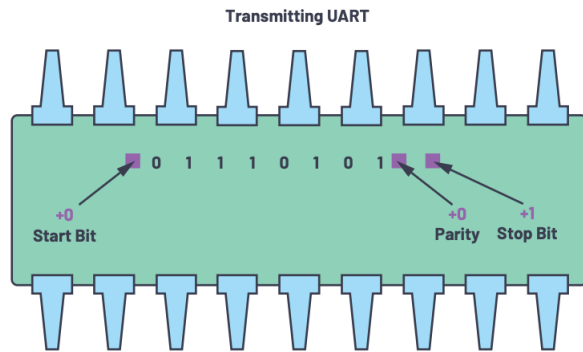
Step 1: UART1 transfers the parallel data and puts it in THR

First: The transmitting UART receives data in parallel from the data bus.



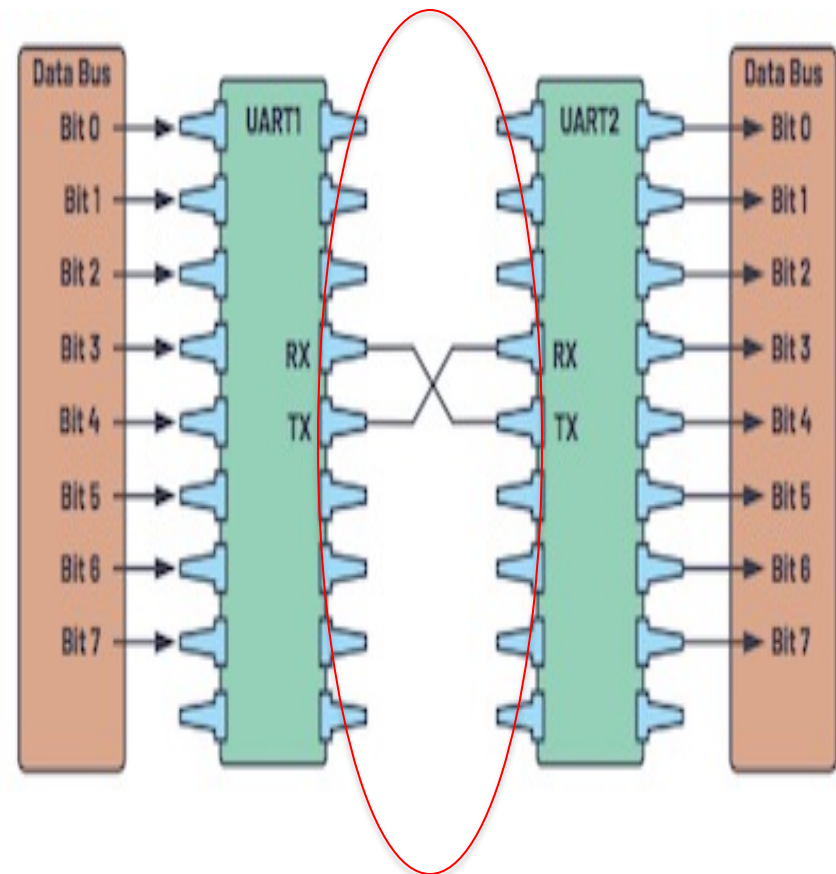
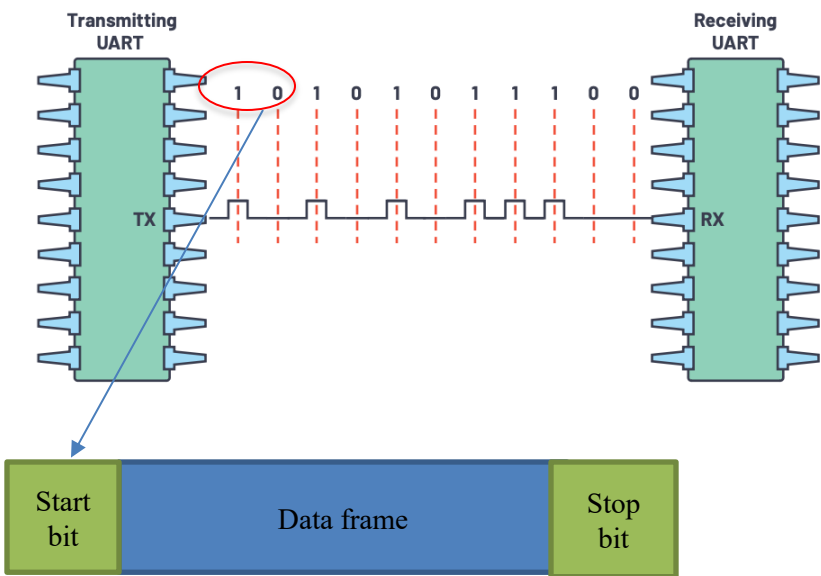
Step 2: UART1 adds start, parity and stop bit to the data

Second: The transmitting UART adds the start bit, parity bit, and the stop bit(s) to the data frame.



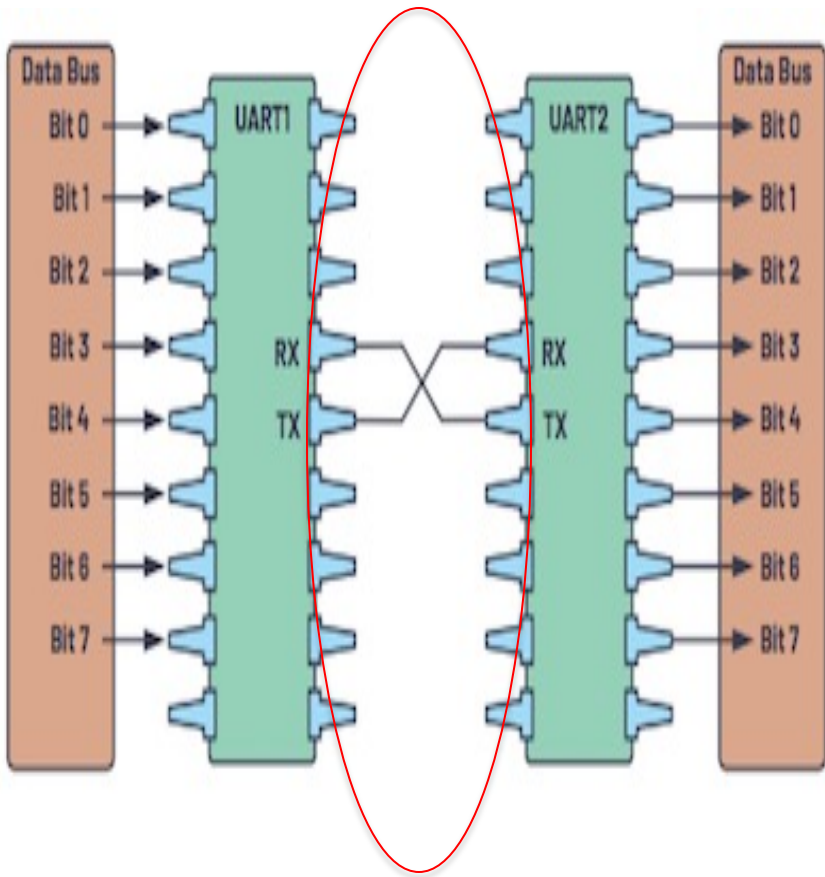
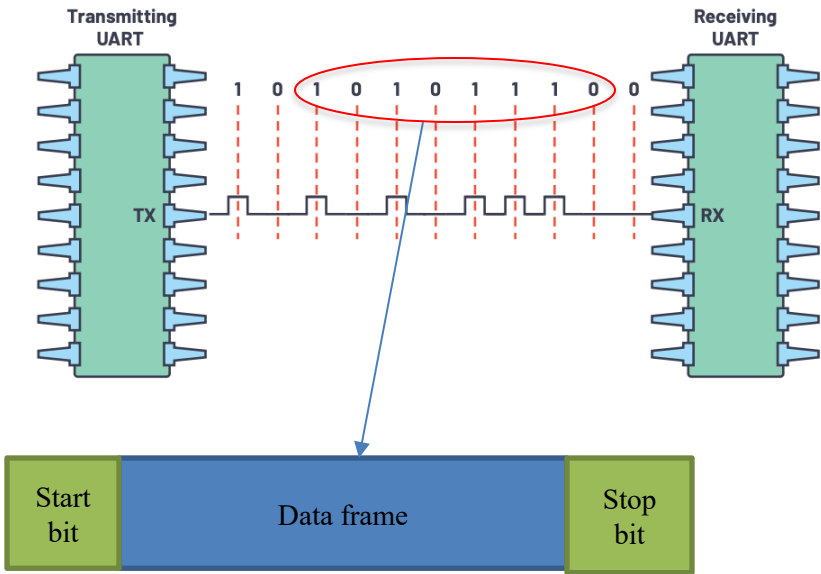
Step 3: UART1 transmitting data serially bit by bit.

Third: The entire packet is sent serially starting from start bit to stop bit from the transmitting UART to the receiving UART. The receiving UART samples the data line at the preconfigured baud rate.



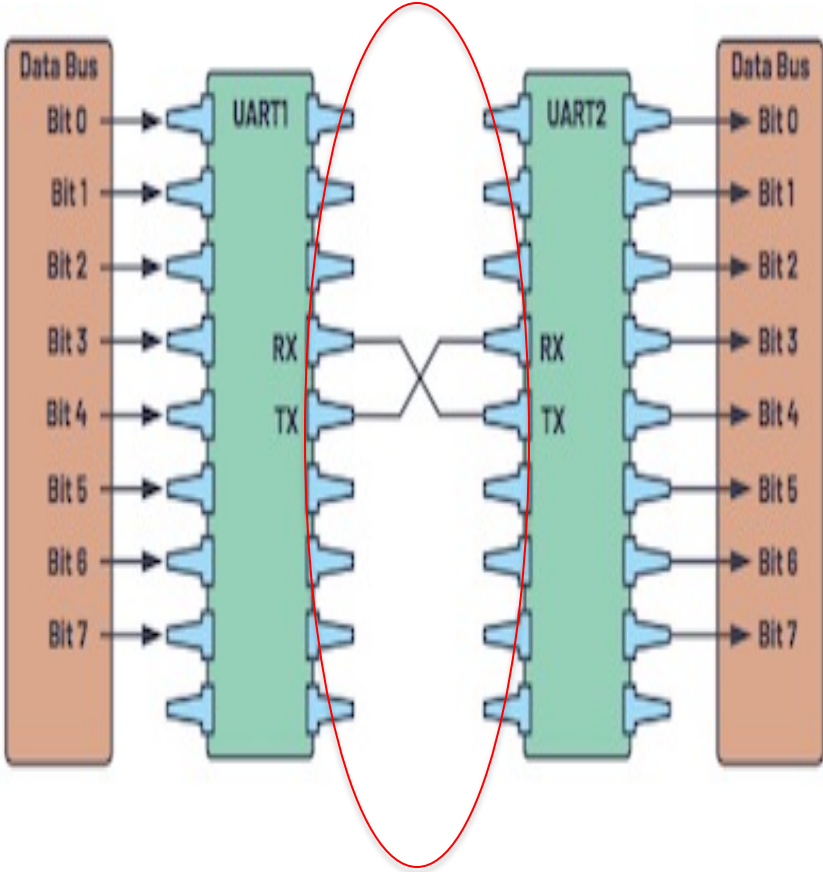
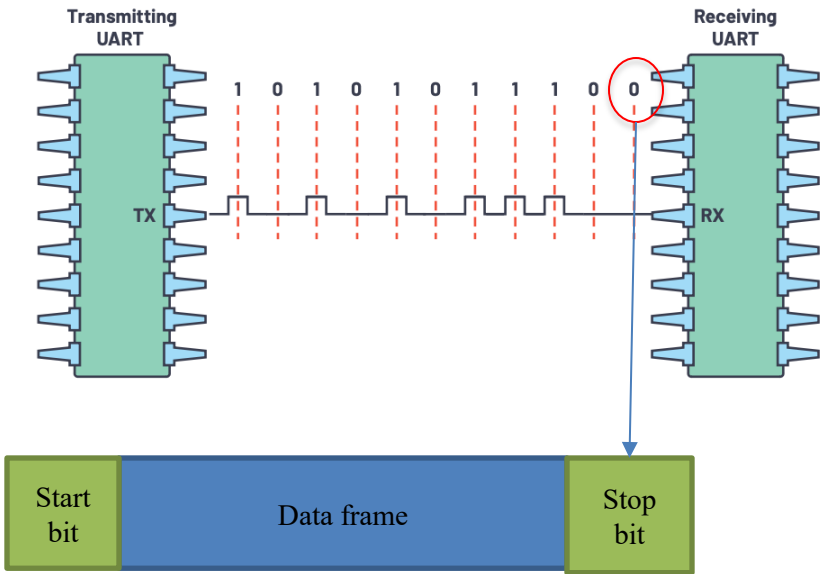
Step 3: UART1 transmitting data serially bit by bit.

Third: The entire packet is sent serially starting from start bit to stop bit from the transmitting UART to the receiving UART. The receiving UART samples the data line at the preconfigured baud rate.



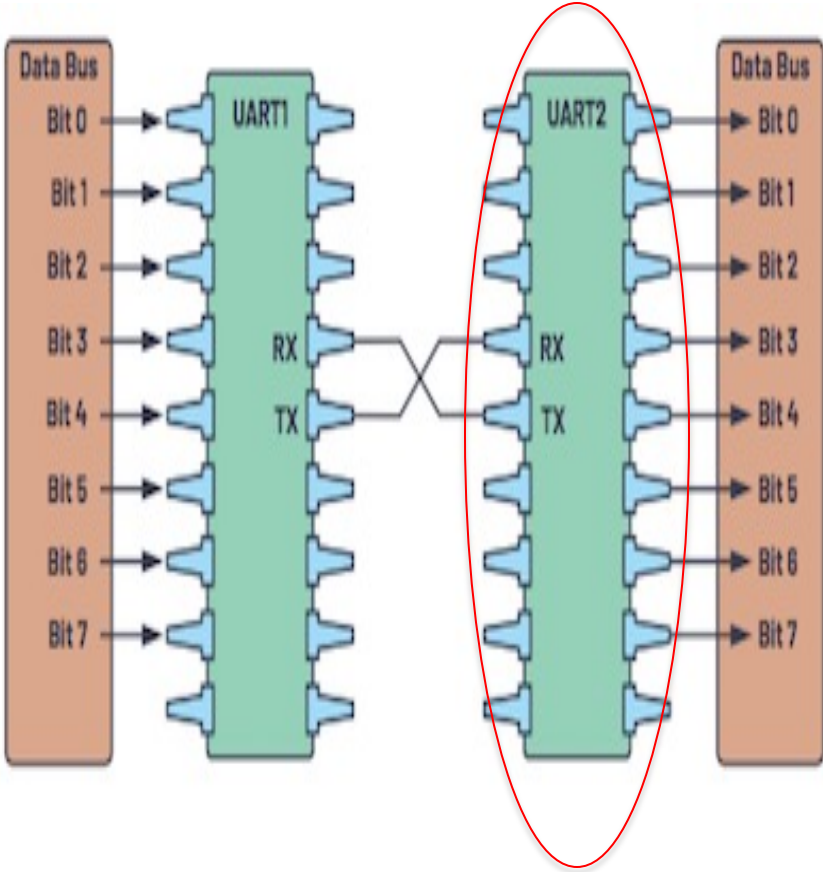
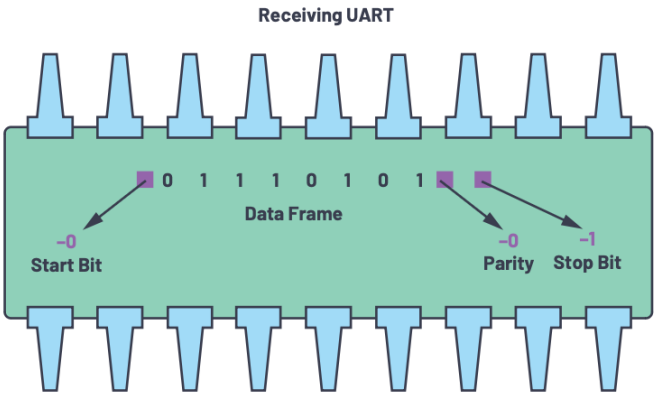
Step 3: UART1 transmitting data serially bit by bit.

Third: The entire packet is sent serially starting from start bit to stop bit from the transmitting UART to the receiving UART. The receiving UART samples the data line at the preconfigured baud rate.



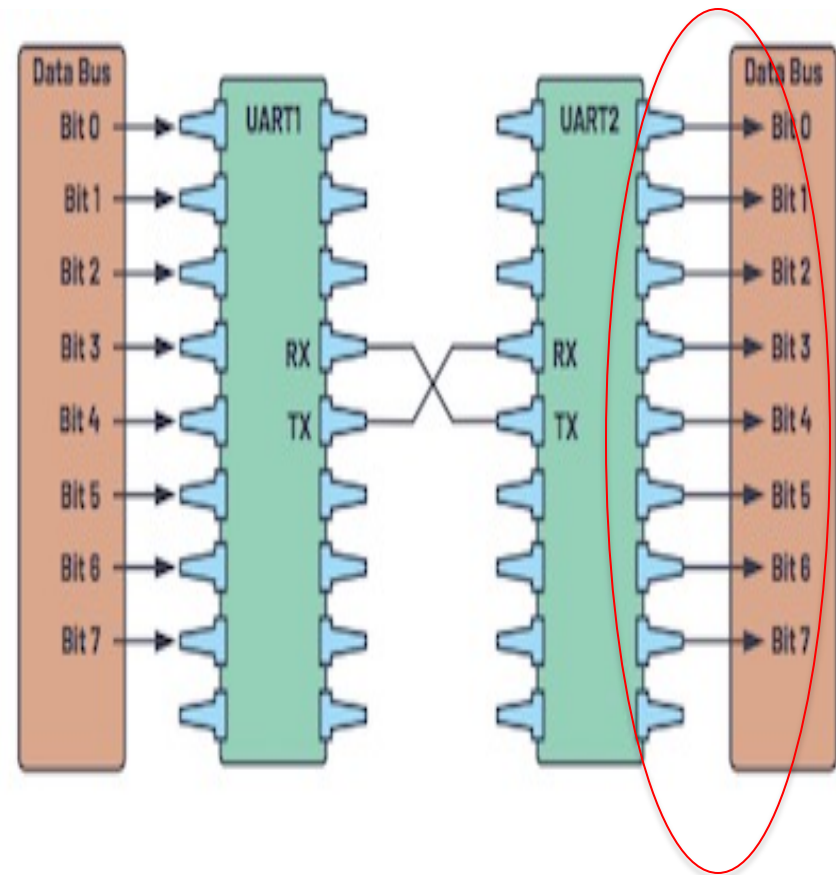
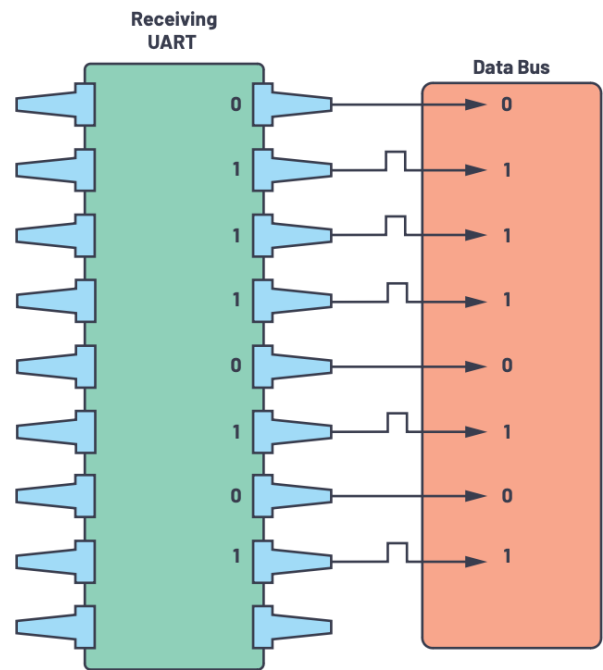
Step 4: UART2 receives the serial data in RBR

Fourth: The receiving UART discards the start bit, parity bit, and stop bit from the data frame.



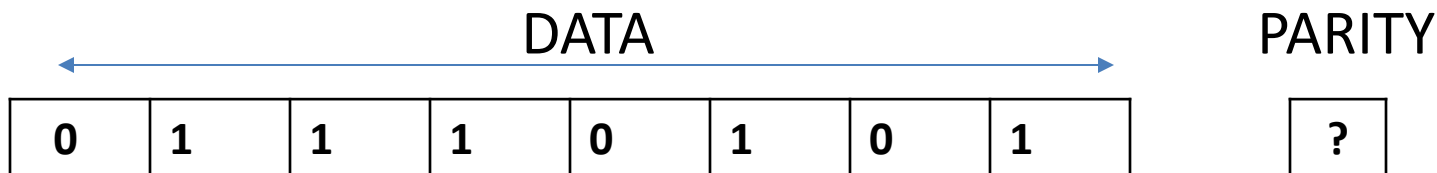
Step 5: UART2 transfers the data parallelly onto data bus.

Fifth: The receiving UART converts the serial data back into parallel and transfers it to the data bus on the receiving end.



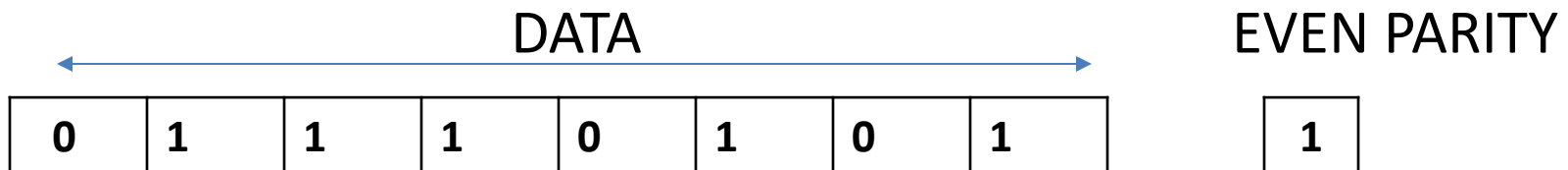
UART protocol

- No clock signal used, so how will the receiver know when one unit of data (data packet) is received?
 - using the same baud rate
 - using start and stop bit
- How about corruption of data?
 - Use of parity bit.
- How does Parity help in detecting data corruption?



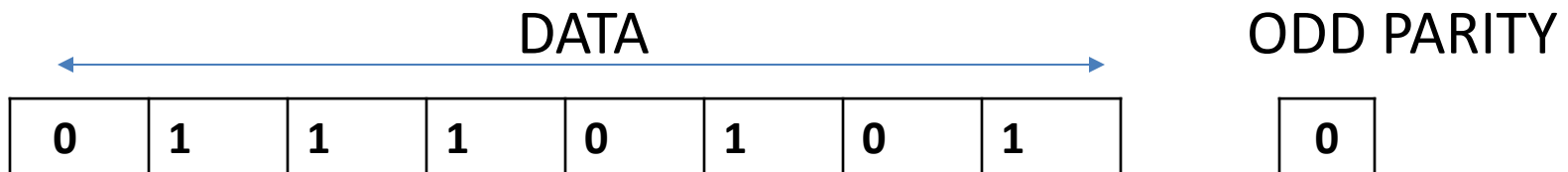
UART protocol

- No clock signal used, so how will the receiver know when one unit of data (data packet) is received?
 - using the same baud rate
 - using start and stop bit
- How about corruption of data?
 - Use of parity bit.
- How does Parity help in detecting data corruption?



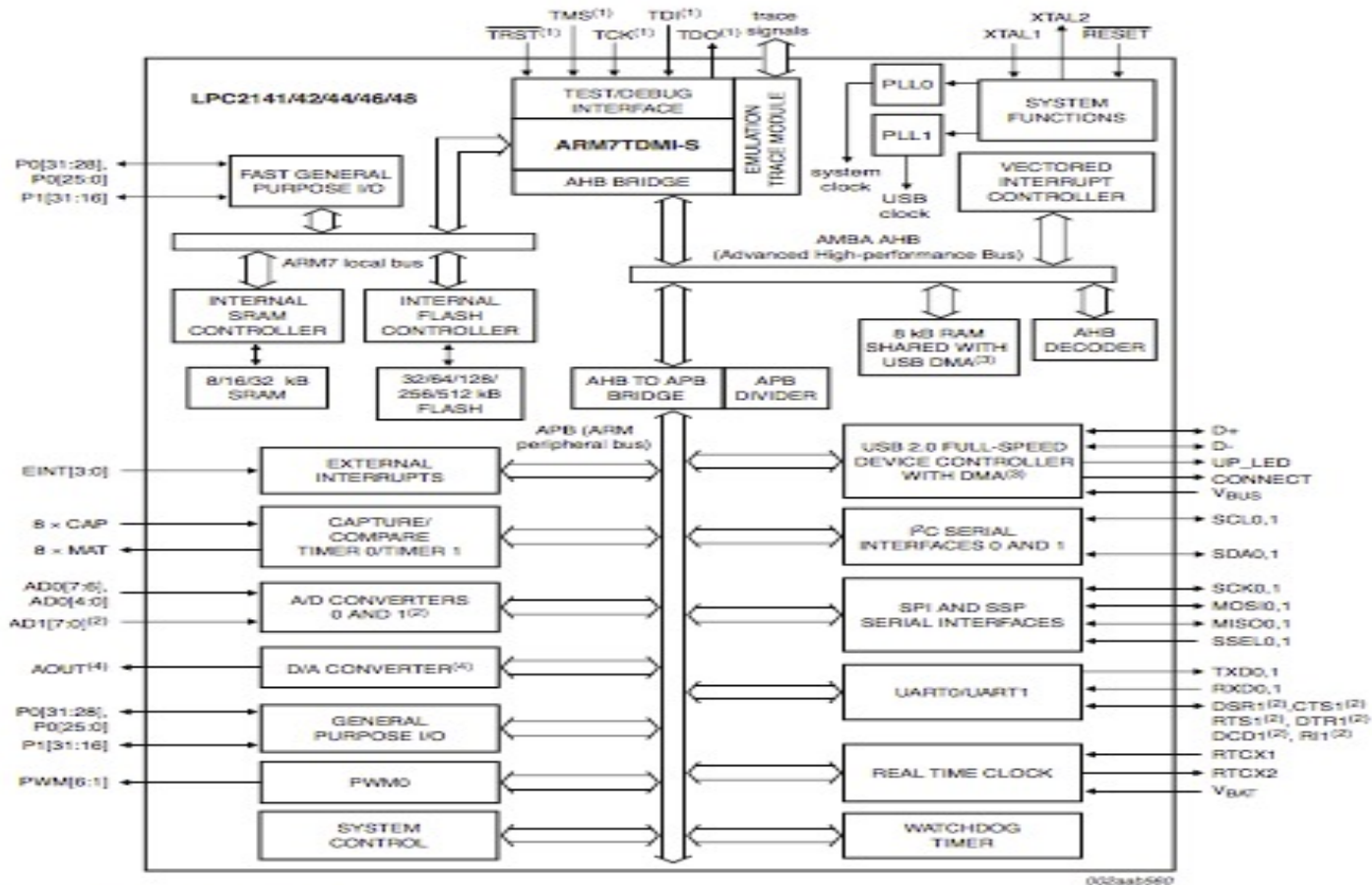
UART protocol

- No clock signal used, so how will the receiver know when one unit of data (data packet) is received?
 - using the same baud rate
 - using start and stop bit
- How about corruption of data?
 - Use of parity bit.
- How does Parity help in detecting data corruption?



UART DEVICE

UART in LPC2148 : block diagram



(1) Pins shared with GPIO.

(2) LPCC2144/6/8 only.

(3) USB DMA controller with 8 kB of RAM accessible as general purpose RAM and/or DMA is available in LPC2148/8 only.

(4) LPC2142/4/6/8 only.

Fig 1. LPC2141/2/4/6/8 block diagram

UART in LPC2148 : block diagram

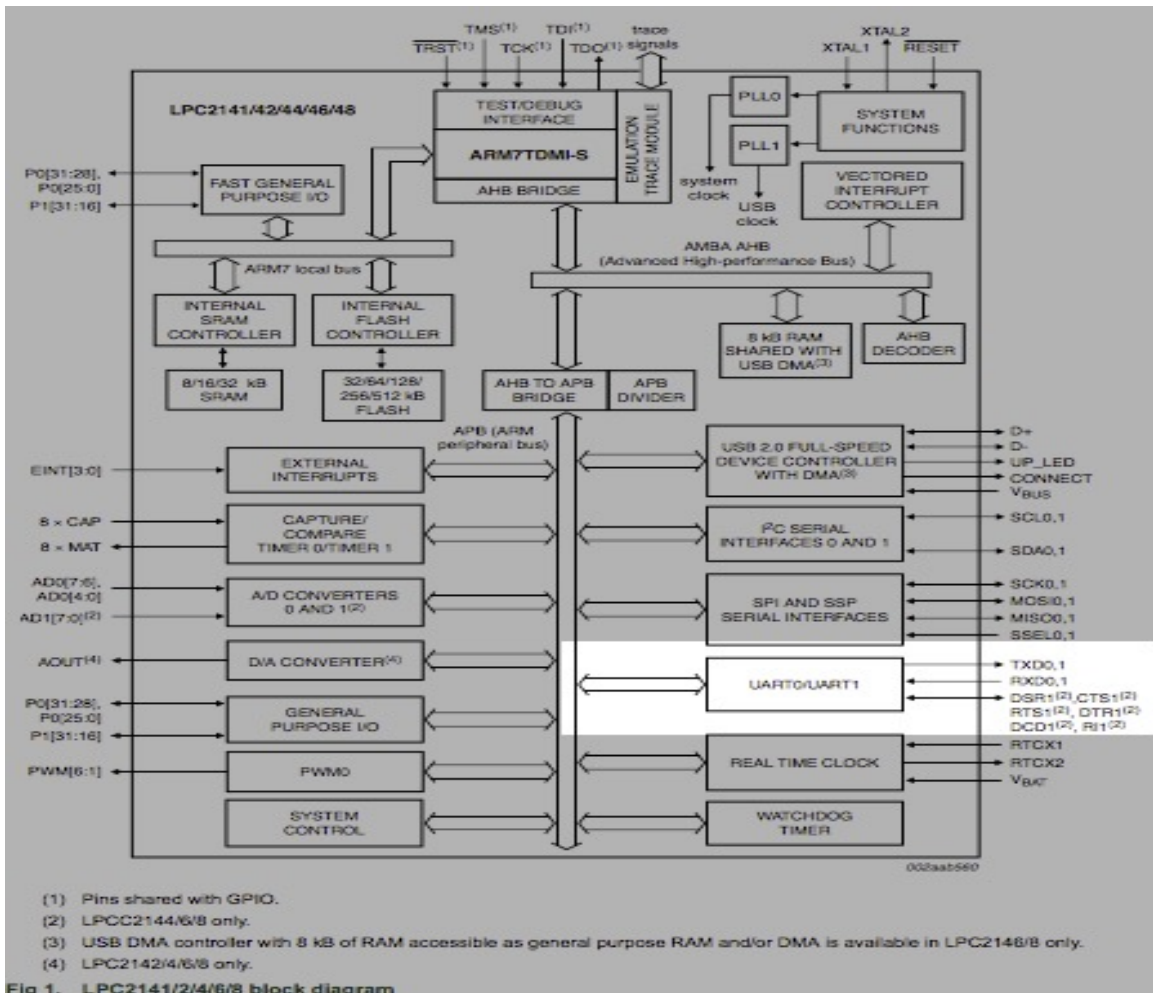
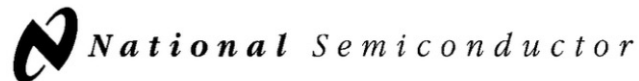


Table 161: UART0 register map

Name	Description	Bit functions and addresses								Access	Reset value ^[1]	Address
		MSB				LSB						
		BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0			
U0RBR	Receiver Buffer Register	8-bit Read Data								RO	NA	0xE000 C000 (DLAB=0)
U0THR	Transmit Holding Register	8-bit Write Data								WO	NA	0xE000 C000 (DLAB=0)
U0DLL	Divisor Latch LSB	8-bit Data								R/W	0x01	0xE000 C000 (DLAB=1)
U0DLM	Divisor Latch MSB	8-bit Data								R/W	0x00	0xE000 C004 (DLAB=1)
U0IER	Interrupt Enable Register	-	-	-	-	-	-	En.ABTO	En.ABEO	R/W	0x00	0xE000 C004 (DLAB=0)
		-	-	-	-	-	En.RX Lin.St.Int	Enable THRE Int	En. RX Dat.Av.Int			
U0IIR	Interrupt ID Reg.	-	-	-	-	-	-	ABTO Int	ABEO Int	RO	0x01	0xE000 C008
		FIFOs Enabled		-	-	IIR3	IIR2	IIR1	IIR0			
U0FCR	FIFO Control Register	RX Trigger		-	-	-	TX FIFO Reset	RX FIFO Reset	FIFO Enable	WO	0x00	0xE000 C008
U0LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Par.Selct.	Parity Enable	No. of Stop Bits	Word Length Select		R/W	0x00	0xE000 C00C
U0LSR	Line Status Register	RX FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	0xE000 C014
U0SCR	Scratch Pad Reg.	8-bit Data								R/W	0x00	0xE000 C01C
U0ACR	Auto-baud Control Register	-	-	-	-	-	-	ABTO Int.Clr	ABEO Int.Clr	R/W	0x00	0xE000 C020
		-	-	-	-	-	Aut.Rstrtl.	Mode	Start			
U0FDR	Fractional Divider Register	Reserved[31:8]									0x10	0xE000 C028
		MulVal				DivAddVal						
U0TER	TX. Enable Reg.	TXEN	-	-	-	-	-	-	-	R/W	0x80	0xE000 C030



June 1995

PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs†

General Description

The PC16550D is an improved version of the original 16450 Universal Asynchronous Receiver/Transmitter (UART). Functionally identical to the 16450 on powerup (CHARACTER mode)* the PC16550D can be put into an alternate mode (FIFO mode) to relieve the CPU of excessive software overhead.

In this mode internal FIFOs are activated allowing 16 bytes (plus 3 bits of error data per byte in the RCVR FIFO) to be stored in both receive and transmit modes. All the logic is on chip to minimize system overhead and maximize system efficiency. Two pin functions have been changed to allow signalling of DMA transfers.

The UART performs serial-to-parallel conversion on data characters received from a peripheral device or a MODEM, and parallel-to-serial conversion on data characters received from the CPU. The CPU can read the complete status of the UART at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions (parity, overrun, framing, or break interrupt).

The UART includes a programmable baud rate generator that is capable of dividing the timing reference clock input by divisors of 1 to $(2^{16} - 1)$, and producing a $16 \times$ clock for driving the internal transmitter logic. Provisions are also included to use this $16 \times$ clock to drive the receiver logic. The UART has complete MODEM-control capability, and a processor-interrupt system. Interrupts can be programmed to the user's requirements, minimizing the computing required to handle the communications link.

The UART is fabricated using National Semiconductor's advanced M²C²MOS process.

*Can also be reset to 16450 Mode under software control.

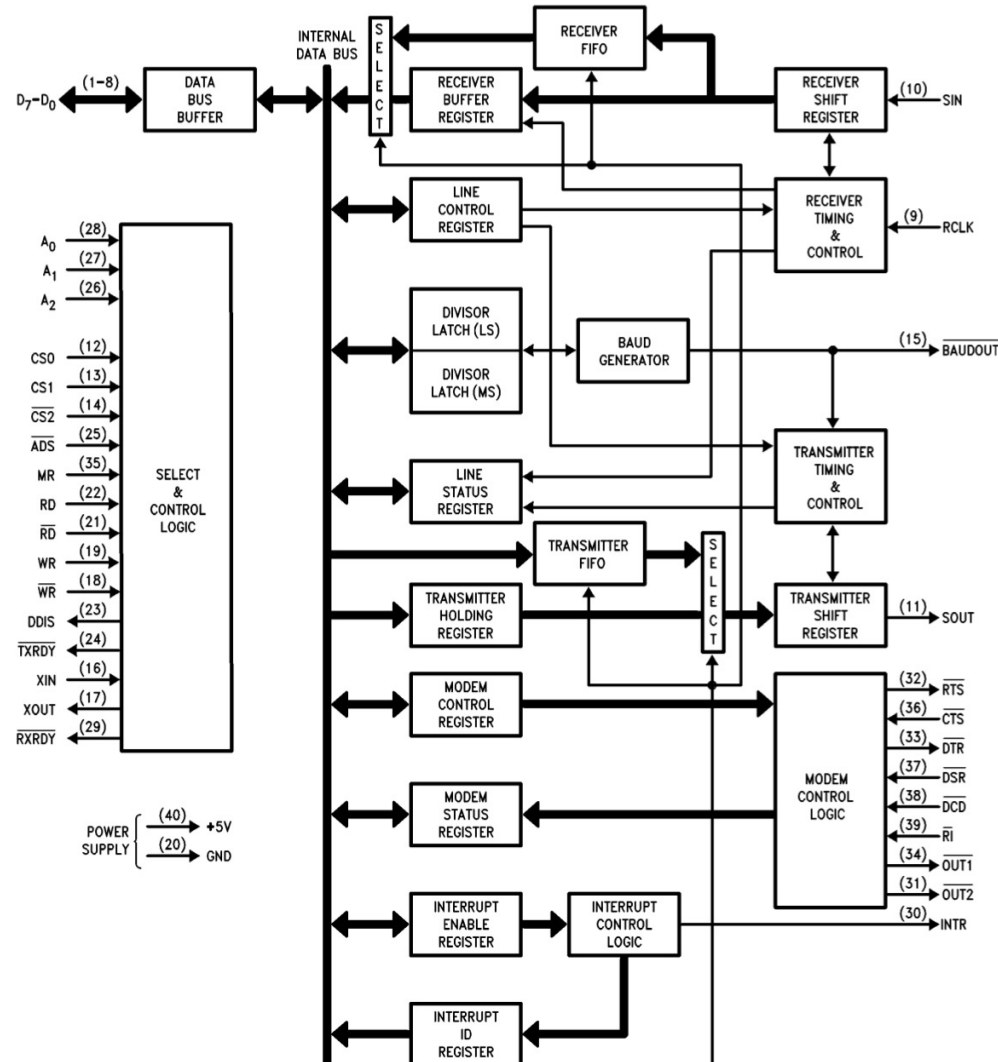
†Note: This part is patented.

Features

- Capable of running all existing 16450 software.
- Pin for pin compatible with the existing 16450 except for CSOUT (24) and NC (29). The former CSOUT and NC pins are TXRDY and RXRDY, respectively.
- After reset, all registers are identical to the 16450 register set.
- In the FIFO mode transmitter and receiver are each buffered with 16 byte FIFO's to reduce the number of interrupts presented to the CPU.
- Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from the serial data.
- Holding and shift registers in the 16450 Mode eliminate the need for precise synchronization between the CPU and serial data.
- Independently controlled transmit, receive, line status, and data set interrupts.
- Programmable baud generator divides any input clock by 1 to $(2^{16} - 1)$ and generates the $16 \times$ clock.
- Independent receiver clock input.
- MODEM control functions (CTS, RTS, DSR, DTR, RI, and DCD).
- Fully programmable serial-interface characteristics:
 - 5-, 6-, 7-, or 8-bit characters
 - Even, odd, or no-parity bit generation and detection
 - 1-, 1½-, or 2-stop bit generation
 - Baud generation (DC to 1.5M baud).
- False start bit detection.
- Complete status reporting capabilities.
- TRI-STATE® TTL drive for the data and control buses.
- Line break generation and detection.
- Internal diagnostic capabilities:
 - Loopback controls for communications link fault isolation
 - Break, parity, overrun, framing error simulation.
- Full prioritized interrupt system controls.

PC16550D: Block diagram

5.0 Block Diagram



PC16550D: Registers

TABLE II. Summary of Registers												
Bit No.	Register Address											
	0 DLAB=0	0 DLAB=0	1 DLAB=0	2	2	3	4	5	6	7	0 DLAB=1	1 DLAB=1
	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Ident. Register (Read Only)	FIFO Control Register (Write Only)	Line Control Register	MODEM Control Register	Line Status Register	MODEM Status Register	Scratch Register	Divisor Latch (LS)	Divisor Latch (MS)
	RBR	THR	IER	IIR	FCR	LCR	MCR	LSR	MSR	SCR	DLL	DLM
0	Data Bit 0 (Note 1)	Data Bit 0	Enable Received Data Available Interrupt (ERBFI)	"0" if Interrupt Pending	FIFO Enable	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (DCTS)	Bit 0	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Enable Transmitter Holding Register Empty Interrupt (ETBEI)	Interrupt ID Bit (0)	RCVR FIFO Reset	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready (DDSR)	Bit 1	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Enable Receiver Line Status Interrupt (ELSI)	Interrupt ID Bit (1)	XMIT FIFO Reset	Number of Stop Bits (STB)	Out 1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	Enable MODEM Status Interrupt (EDSSI)	Interrupt ID Bit (2) (Note 2)	DMA Mode Select	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Data Carrier Detect (DDCD)	Bit 3	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Reserved	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Reserved	Stick Parity	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	FIFOs Enabled (Note 2)	RCVR Trigger (LSB)	Set Break	0	Transmitter Empty (TEMT)	Ring Indicator (RI)	Bit 6	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	FIFOs Enabled (Note 2)	RCVR Trigger (MSB)	Divisor Latch Access Bit (DLAB)	0	Error in RCVR FIFO (Note 2)	Data Carrier Detect (DCD)	Bit 7	Bit 7	Bit 15
Note 1: Bit 0 is the least significant bit. It is the first bit serially transmitted or received. Note 2: These bits are always 0 in the 16450 Mode.												

PC16550D: Registers to be considered

TABLE II. Summary of Registers												
Bit No.	Register Address											
	0 DLAB = 0	0 DLAB = 0	1 DLAB = 0	2	2	3	4	5	6	7	0 DLAB = 1	1 DLAB = 1
	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Ident. Register (Read Only)	FIFO Control Register (Write Only)	Line Control Register	MODEM Control Register	Line Status Register	MODEM Status Register	Scratch Register	Divisor Latch (LS)	Divisor Latch (MS)
	RBR	THR	IER	IIR	FCR	LCR	MCR	LSR	MSR	SCR	DLL	DLM
0	Data Bit 0 (Note 1)	Data Bit 0	Enable Received Data Available Interrupt (ERBF)	"0" if Interrupt Pending	FIFO Enable	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (DCTS)	Bit 0	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Enable Transmitter Holding Register Empty Interrupt (ETBEI)	Interrupt ID Bit (0)	RCVR FIFO Reset	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready (DDSR)	Bit 1	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Enable Receiver Line Status Interrupt (ELSI)	Interrupt ID Bit (1)	XMIT FIFO Reset	Number of Stop Bits (STB)	Out 1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	Enable MODEM Status Interrupt (EDSSI)	Interrupt ID Bit (2) (Note 2)	DMA Mode Select	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Data Carrier Detect (DDCD)	Bit 3	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Reserved	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Reserved	Stick Parity	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	FIFOs Enabled (Note 2)	RCVR Trigger (LSB)	Set Break	0	Transmitter Empty (TEMT)	Ring Indicator (RI)	Bit 6	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	FIFOs Enabled (Note 2)	RCVR Trigger (MSB)	Divisor Latch Access Bit (DLAB)	0	Error in RCVR FIFO (Note 2)	Data Carrier Detect (DCD)	Bit 7	Bit 7	Bit 15
Note 1: Bit 0 is the least significant bit. It is the first bit serially transmitted or received. Note 2: These bits are always 0 in the 16450 Mode.												

Understanding Register information

Takeaways:

- Each register is made of 8 bits.

Bit position	7	6	5	4	3	2	1	0
RBR/THR/DLL (0)	← Data →							
DLM(1)								
LCR(3)	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Stop Bit (STB)	Word Length Select (WLS)	
LSR(5)	Error in Rcvr FIFO	Transmitter Empty	Transmitter Holding Register	Break Interrupt (BI)	Framing error (FE)	Parity error (PE)	Overrun error (OE)	Data Ready (DR)

Hardware – Electronic circuitry

Understanding Register information

Takeaways:

- Each register is made of 8 bits.
- Each register is mapped to a memory location. In LPC2128 data sheet the actual memory address is given, and in pc16550D the offsets are given. Can you guess why?

Bit position	7	6	5	4	3	2	1	0
RBR/THR/DLL (0)	← Data →							
DLM(1)								
LCR(3)	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Stop Bit (STB)	Word Length Select (WLS)	
LSR(5)	Error in Rcvr FIFO	Transmitter Empty	Transmitter Holding Register	Break Interrupt (BI)	Framing error (FE)	Parity error (PE)	Overrun error (OE)	Data Ready (DR)

Hardware – Electronic circuitry

Information in the datasheets

Register table from LPC2148

Name	Description	Bit functions and addresses								Access	Reset value ^[1]	Address
		MSB				LSB						
		BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0			
U0RBR	Receiver Buffer Register	8-bit Read Data								RO	NA	0xE000 C000 (DLAB=0)
U0THR	Transmit Holding Register	8-bit Write Data								WO	NA	0xE000 C000 (DLAB=0)
U0DLL	Divisor Latch LSB	8-bit Data								R/W	0x01	0xE000 C000 (DLAB=1)
U0DLM	Divisor Latch MSB	8-bit Data								R/W	0x00	0xE000 C004 (DLAB=1)
U0IER	Interrupt Enable	-	-	-	-	-	-	En.ABTO	En.ABEO	R/W	0x00	0xE000 C004


Register table from PC16550D

Bit No.	Register Address											
	0 DLAB=0	0 DLAB=0	1 DLAB=0	2	2	3	4	5	6	7	0 DLAB=1	1 DLAB=1
	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Ident. Register (Read Only)	FIFO Control Register (Write Only)	Line Control Register	MODEM Control Register	Line Status Register	MODEM Status Register	Scratch Register	Divisor Latch (LS)	Divisor Latch (MS)
	RBR	THR	IER	IIR	FCR	LCR	MCR	LSR	MSR	SCR	DLL	DLM

About RBR, THR and DLL

Takeaways:


- 3 registers RBR, THR and DLL are mapped to 0 offset, how do we know that the information at the location is to which register?

Bit position	7	6	5	4	3	2	1	0
RBR/THR/DLL (0)								
DLM(1)								
LCR(3)	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Stop Bit (STB)	Word Length Select (WLS)	
LSR(5)	Error in Rcvr FIFO	Transmitter Empty	Transmitter Holding Register	Break Interrupt (BI)	Framing error (FE)	Parity error (PE)	Overrun error (OE)	Data Ready (DR)

About RBR, THR and DLL

Takeaways:

- 3 registers RBR, THR and DLL are mapped to 0 offset, how do we know that the information at the location is to which register?
 - Understand the use of DLAB bit in the register LCR to access registers at offset 0 and 1.

Bit position	7	6	5	4	3	2	1	0
RBR/THR/DLL (0)								
DLM(1)								
LCR(3)	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Stop Bit (STB)	Word Length Select (WLS)	
LSR(5)	Error in Rcvr FIFO	Transmitter Empty	Transmitter Holding Register	Break Interrupt (BI)	Framing error (FE)	Parity error (PE)	Overrun error (OE)	Data Ready (DR)

About RBR, THR and DLL

Takeaways:

- 3 registers RBR, THR and DLL are mapped to 0 offset, how do we know that the information at the location is to which register?
 - Understand the use of DLAB bit in the register LCR to access registers at offset 0 and 1.
 - Access type of RBR and THR are complimentary. One is read only, while the other is write only.

Bit position	7	6	5	4	3	2	1	0
RBR/THR /DLL (0)	← Data →							
DLM(1)								
LCR(3)	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Stop Bit (STB)	Word Length Select (WLS)	
LSR(5)	Error in Rcvr FIFO	Transmitter Empty	Transmitter Holding Register	Break Interrupt (BI)	Framing error (FE)	Parity error (PE)	Overrun error (OE)	Data Ready (DR)

About RBR, THR and DLL

Takeaways:

- 3 registers RBR, THR and DLL are mapped to 0 offset, how do we know that the information at the location is to which register?
 - Understand the use of DLAB bit in the register LCR to access registers at offset 0 and 1.
 - Access type of RBR and THR are complimentary. One is read only, while the other is write only.
 - What this means that:
 - when the pin WR is high, the s/w running in the CPU is accessing THR
 - when the pin RD is high, the s/w running in the CPU is accessing RBR

Bit position	7	6	5	4	3	2	1	0
RBR/THR /DLL (0)	← Data →							
DLM(1)								
LCR(3)	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Stop Bit (STB)	Word Length Select (WLS)	
LSR(5)	Error in Rcvr FIFO	Transmitter Empty	Transmitter Holding Register	Break Interrupt (BI)	Framing error (FE)	Parity error (PE)	Overrun error (OE)	Data Ready (DR)

About RBR, THR, DLL and DLM

Takeaways:

- RBR, THR, DLL and DLM registers are accessed as 8 bit values.
- RBR:
 - Receives the serial data from pin **sin** and stores it in the 8 bit register
 - Is “read only”(RO) register, in the sense that you cannot write into it.
- THR :
 - Transmits the contents of the register serially on pin **sout**
 - Is “write only” (WO) register, in the sense that you cannot read from it.
- DLL and DLM: are referred to as the divisor latch registers and are used to configure the baud rate.

Bit position	7	6	5	4	3	2	1	0
RBR/THR/DLL (0)								
DLM(1)								
LCR(3)	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Stop Bit (STB)	Word Length Select (WLS)	
LSR(5)	Error in Rcvr FIFO	Transmitter Empty	Transmitter Holding Register	Break Interrupt (BI)	Framing error (FE)	Parity error (PE)	Overrun error (OE)	Data Ready (DR)

Bits 0 & 1 of LCR Register

Takeaways:

- Offset is 3
- Access type is Read and Write (RW)
- This register can be considered to be divided into different sub sections:
 - Bit 0 and 1 deals with configuring the word length (**WLS**). The values are:

Bit 0	Bit 1	Data/word length
0	0	5 bit data
0	1	6 bit data
1	0	7 bit data
1	1	8 bit data

Bit position	7	6	5	4	3	2	1	0
RBR/THR/DLL (0)								
DLM(1)								
LCR(3)	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Stop Bit (STB)	Word Length Select (WLS)	
LSR(5)	Error in Rcvr FIFO	Transmitter Empty	Transmitter Holding Register	Break Interrupt (BI)	Framing error (FE)	Parity error (PE)	Overrun error (OE)	Data Ready (DR)

Bit 2 of LCR Register

Takeaways:

- Offset is 3
- Access type is Read and Write (RW)
- This register can be considered to be divided into different sub sections
 - Bit 2 deals with configuring the stop bit(**STB**). This is just 1 bit, but has 3 values: 1 stop bit, 1.5 stop bit and 2 stop bit!! This is how it is done:

Bit 0	Bit 1	Bit 2	Behaviour
x	x	0	1 stop bit
0	0	1	1.5 stop bit
0	1	1	2 stop bit
1	0	1	2 stop bit
1	1	1	2 stop bit

Bit position	7	6	5	4	3	2	1	0
RBR/THR/DLL (0)								
DLM(1)								
LCR(3)	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Stop Bit (STB)	Word Length Select (WLS)	
LSR(5)	Error in Rcvr FIFO	Transmitter Empty	Transmitter Holding Register	Break Interrupt (BI)	Framing error (FE)	Parity error (PE)	Overrun error (OE)	Data Ready (DR)

Bits 3, 4 & 5 of LCR Register

Takeaways:

- Offset is 3
- Access type is Read and Write (RW)
- This register can be considered to be divided into different sub sections
 - Bit 3, Bit 4 and Bit 5 deal with configuring the parity.
 - Bit 3 : Parity Enable (**PEN**) enables and disables parity.
 - Bit 4: Even Parity (**EPS**) enables even and odd parity.
 - Bit 5: is called **Stick Parity** which works as follows:

Bit 3	Bit 4	Bit 5	Behavior
1	1	1	Parity enabled to even, the parity check as logic 0
1	0	1	Parity enabled to odd, the parity check as logic 1
x	x	0	Stick parity disabled

Bit position	7	6	5	4	3	2	1	0
RBR/THR/DLL (0)								
DLM(1)								
LCR(3)	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Stop Bit (STB)	Word Length Select (WLS)	
LSR(5)	Error in Rcvr FIFO	Transmitter Empty	Transmitter Holding Register	Break Interrupt (BI)	Framing error (FE)	Parity error (PE)	Overrun error (OE)	Data Ready (DR)

Bits 6 & 7 of LCR Register

Takeaways:

- Offset is 3
- Access type is Read and Write (RW)
- This register can be considered to be divided into different sub sections
 - Bit 6 **sets the Break** condition. A 1 in this bit position forces pin **sout** to a spacing state (logic 0). This is used to prevent extraneous characters to be sent.
 - Bit 7 (**DLAB**) A 1 in this bit position enables access to Divisor latch registers (DLL and DLM).

Bit position	7	6	5	4	3	2	1	0
RBR/THR/DLL (0)								
DLM(1)								
LCR(3)	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Stop Bit (STB)	Word Length Select (WLS)	
LSR(5)	Error in Rcvr FIFO	Transmitter Empty	Transmitter Holding Register	Break Interrupt (BI)	Framing error (FE)	Parity error (PE)	Overrun error (OE)	Data Ready (DR)

Bits 0, 1, 2, 3 & 4 of LCR Register

Takeaways:

- Offset is 4
- Access type is Read and Write (RW)
- This register can be considered to be divided into different sub sections
 - Bit 0 (**DR**): A 1 in this bit, flags that data is available in RBR register.
 - Bit 1 (**OE**): A 1 in this bit, flags that data has overrun error
 - Bit 2 (**PE**): A 1 in this bit, flags that data has parity error
 - Bit 3 (**FE**): A 1 in this bit, flags that data has framing error
 - Bit 4 (**BI**): A 1 in this bit, flags that there was a break in the transmission indicated by a spacing state for more than 1 word transmission.

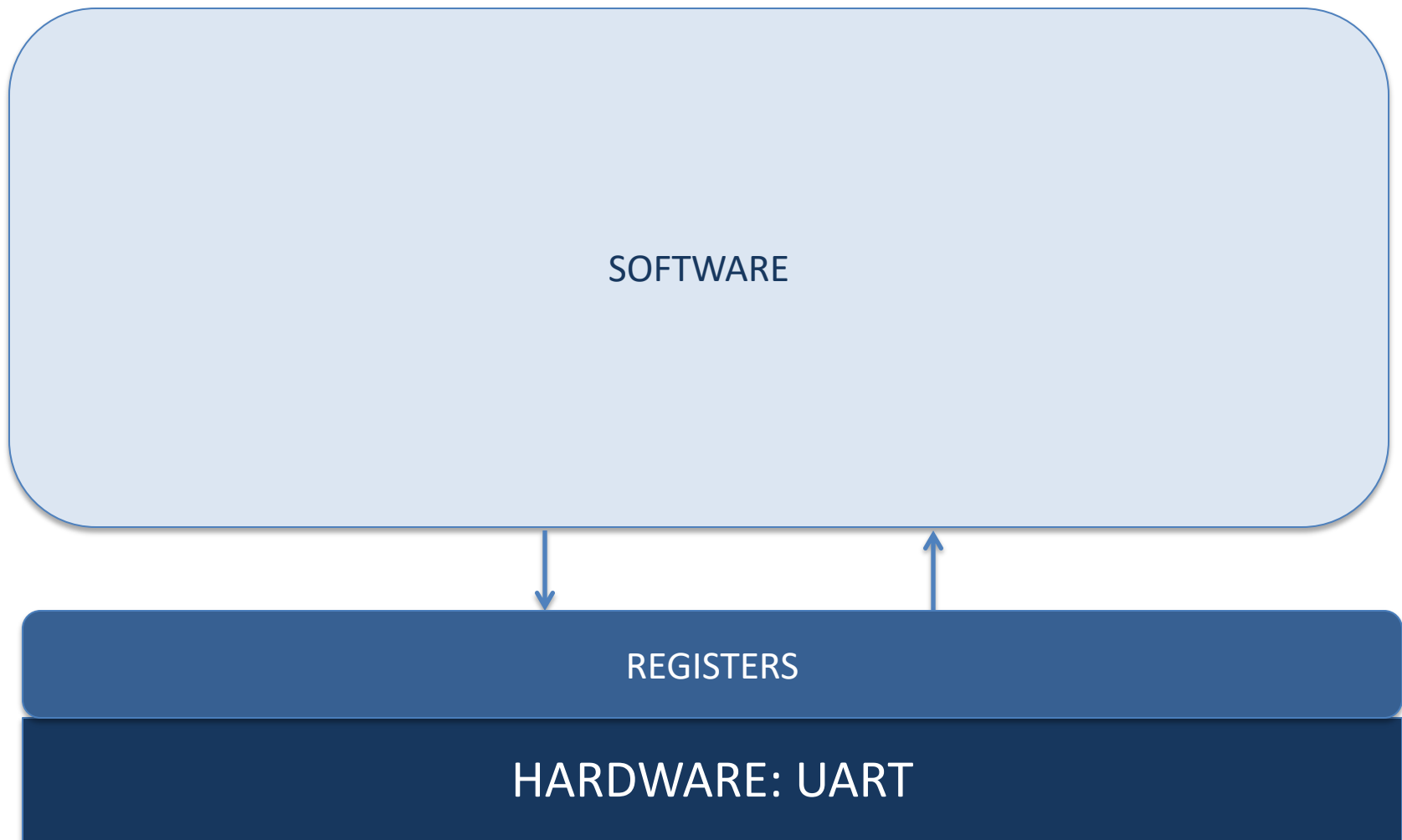
Bit position	7	6	5	4	3	2	1	0
RBR/THR/DLL (0)								
DLM(1)								
LCR(3)	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Stop Bit (STB)	Word Length Select (WLS)	
LSR(5)	Error in Rcvr FIFO	Transmitter Empty	Transmitter Holding Register	Break Interrupt (BI)	Framing error (FE)	Parity error (PE)	Overrun error (OE)	Data Ready (DR)

Bits 0, 1, 2, 3 & 4 of LCR Register

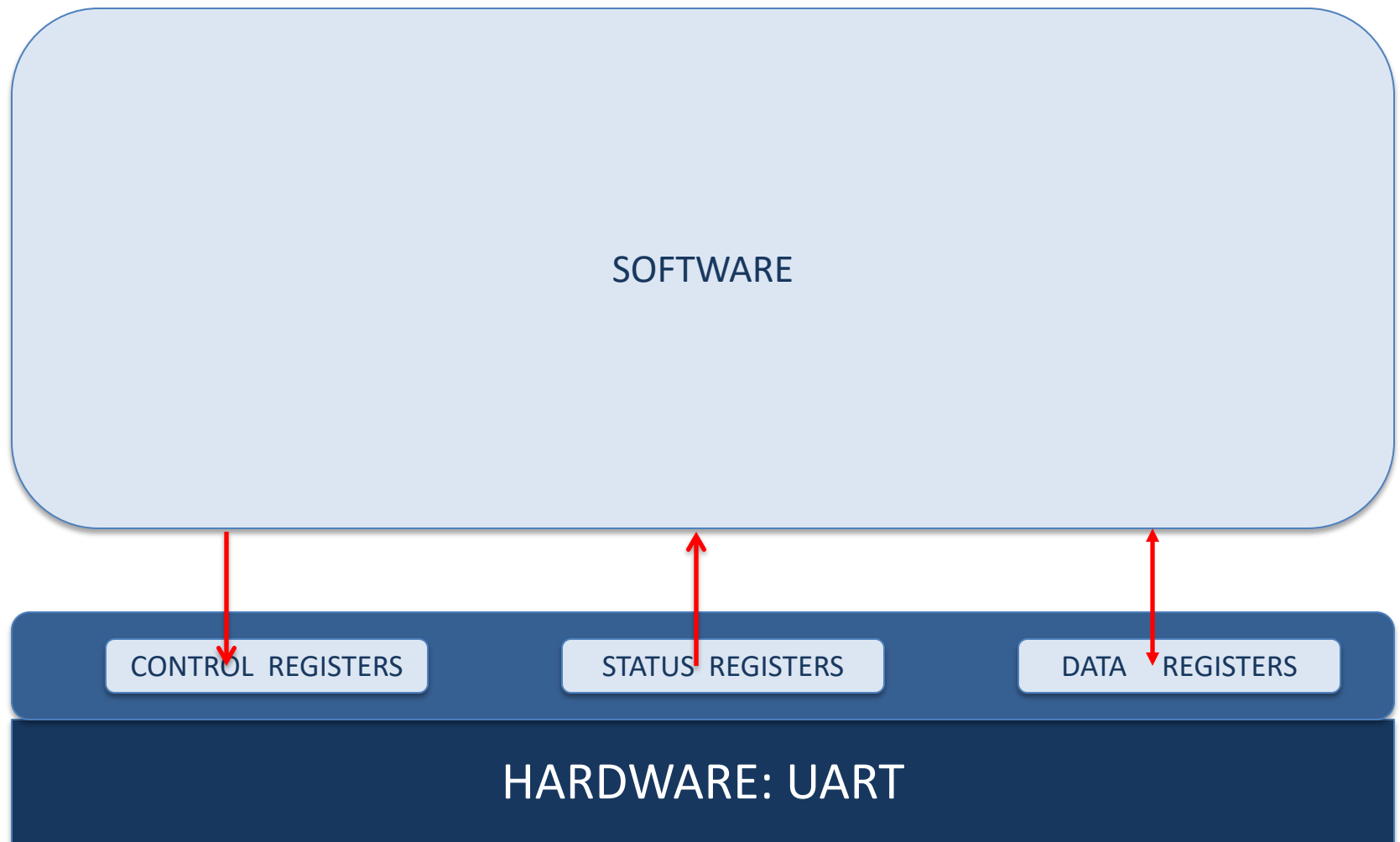
Takeaways:

- Offset is 4
- Access type is Read and Write (RW)
- This register can be considered to be divided into different sub sections
 - Bit 5 (**THRE**): A 1 in this bit, flags that the UART is ready to take new data for transmission. When data is transferred from Transmit Holding register to Transmit Shift register.
 - Bit 6 (**TEMT**): A 1 in this bit, flags that both Transmit Holding register to Transmit Shift register are empty.
 - Bit 7 (**Error in Rceiver FIFO**): A 1 in this bit, flags that there has been a parity error, frame error or break indication for the received data in the FIFO.

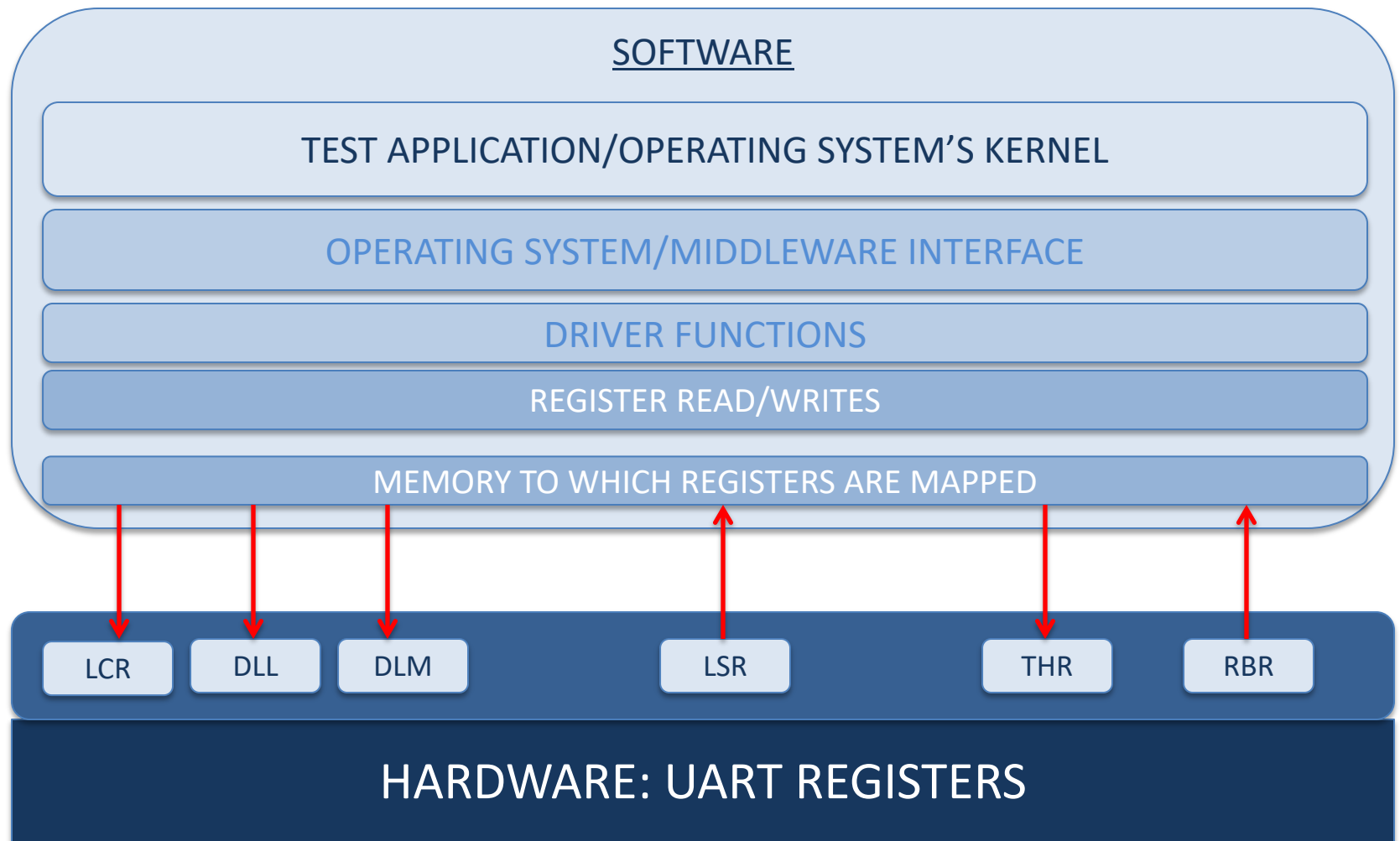
Bit position	7	6	5	4	3	2	1	0
RBR/THR/DLL (0)								
DLM(1)								
LCR(3)	Divisor Latch Access Bit (DLAB)	Set Break	Stick Parity	Even Parity Select (EPS)	Parity Enable (PEN)	Stop Bit (STB)	Word Length Select (WLS)	
LSR(5)	Error in Rcvr FIFO	Transmitter Empty	Transmitter Holding Register	Break Interrupt (BI)	Framing error (FE)	Parity error (PE)	Overrun error (OE)	Data Ready (DR)



Software – Hardware Interface



Software abstraction



Software abstraction: UART example

```
int main()
{
    configure_uart(); transmit_data(); receive_data();
}
```

```
receive_data()
{
    register_read(RBR);
}
```

```
transmit_data(data)
{
    register_write (THR, data)
}
```

```
register_read(?)
{ ...
}
```

```
register_write(?, int value)
{ ...
}
```

offset	name	offset	name	offset	name	offset	name
0	RBR/THR/DLM	2	IIIR/FCR	4	MCR	6	MSR
1	IER/DLL	3	LCR	5	LSR		

HARDWARE: UART

UART DRIVER DESIGN

UART Driver: Features

The driver functions will be based on the following UART features:

UART feature	comments
Configure the baud rate	Configuration values:
Configure the data length	Configuration values:
Configure the stop bit	Configuration values:
Configure the parity	Configuration values:
Receive data	
Transmit data	

UART Driver: Features

The driver functions will be based on the following UART features:

UART feature	comments
Configure the baud rate	Configuration values: 50, ...9600, ... 56000
Configure the data length	Configuration values: 5, 6, 7, 8
Configure the stop bit	Configuration values: 1, 1.5, 2
Configure the parity	Configuration values: odd, even
Receive data	
Transmit data	

UART Driver: Features

The driver functions will be based on the following UART features:

UART feature	comments
Configure the baud rate	Configuration values: 50, ...9600, ... 56000
Configure the data packet	Configuration values for: <ul style="list-style-type: none">• data length: 5, 6, 7, 8• stop bit: 1, 1.5, 2• parity: odd, even
Receive data	
Transmit data	

UART: Configure baud rate - datasheet

8.3 PROGRAMMABLE BAUD GENERATOR

The UART contains a programmable Baud Generator that is capable of taking any clock input from DC to 24 MHz and dividing it by any divisor from 2 to $2^{16}-1$. The output frequency of the Baud Generator is $16 \times$ the Baud [divisor # = (frequency input) \div (baud rate \times 16)]. Two 8-bit latches store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization to ensure proper operation of the Baud Generator. Upon loading either of the Divisor Latches, a 16-bit Baud counter is immediately loaded.

Table III provides decimal divisors to use with crystal frequencies of 1.8432 MHz, 3.072 MHz and 18.432 MHz, respectively. For baud rates of 38400 and below, the error obtained is minimal. The accuracy of the desired baud rate is dependent on the crystal frequency chosen. Using a divisor of zero is **not** recommended.

Takeaways:

- Generated baud rate depends on the input clock frequency and a 16 bit divisor value.
- The 16 bit divisor value has to be calculated using the formula:
$$16\text{-bit divisor} = \text{input-clock-frequency} / (\text{baud-rate} * 16)$$
- The 16 bit divisor to be loaded into the Divisor Latches register.

UART: Configure baud rate - Algorithm

Questions to be asked:

- What should be the values of the *input-clock-frequency* and the *baud-rate*?
 - *input-clock-frequency*: depend on the clocks available on the board
 - *baud-rate*: depend on what the user wants to set the TX and RX UART to.
- Can we assume these values in the function, or not? If not, how do we get these values?
 - hardcoding the values inside the function makes it non-scalable.
 - Take the values as input parameters to the function.
- What are the Divisor Latches mentioned in the datasheet? How to access these?
 - Page 14 in the datasheet refers to DLL and DLM as Divisor Latch registers
 - DLAB bit of LCR register has to be set to 1 in order to access DLL and DLM

UART: Configure baud rate - Algorithm

The Algorithm to configure baud rate would be:

- Take baud rate to be generated and clock frequency as input parameters
- Calculate the 16 bit divisor value using the formula
- $\text{divisor} = \text{clock_frequency} / (\text{baud} * 16)$.
- Set DLAB bit of LCR register to 1 to access DLL and DLM
- Write the most significant 8 bits of the 16-bit divisor into DLM
- Write the least significant 8 bits of the 16-bit divisor into DLL
- Reset DLAB bit of LCR register to 0 so the registers THR and RBR are accessed by default.

UART: Configure baud rate - Declaration

```
/**
 * @brief Configures the baud rate of the UART
 *
 * @param baud : the baud rate of the UART data to be transmitted
 * valid values: 50 ...128000
 * @param clock: the input clock frequency in MHz is from the crystal clock.
 * valid values: 1.8432MHz, 3.072MHz, 18.432MHz
 *
 * @return none
 */
void configure_baud_rate (unsigned int baud, ?? clock);
```

UART: driver design assignment

Come up with function declarations/signatures for the following functions:

`configure_data_width`

`configure_stop_bit`

`configure_parity`

`configure_data_packet`

`transmit_data`

`receive_data`

in the same manner as given for **`configure_baud_rate`** in the previous slide.

Questions to be asked before you start on the design:

1. Should I pass parameters to the function? If so:
 - what should they represent?
 - what values can they take? Based on which what will be the data type
2. Should there be a return value? What values should it take and how should the calling function interpret this?

Thank You

www.vayavyalabs.com

