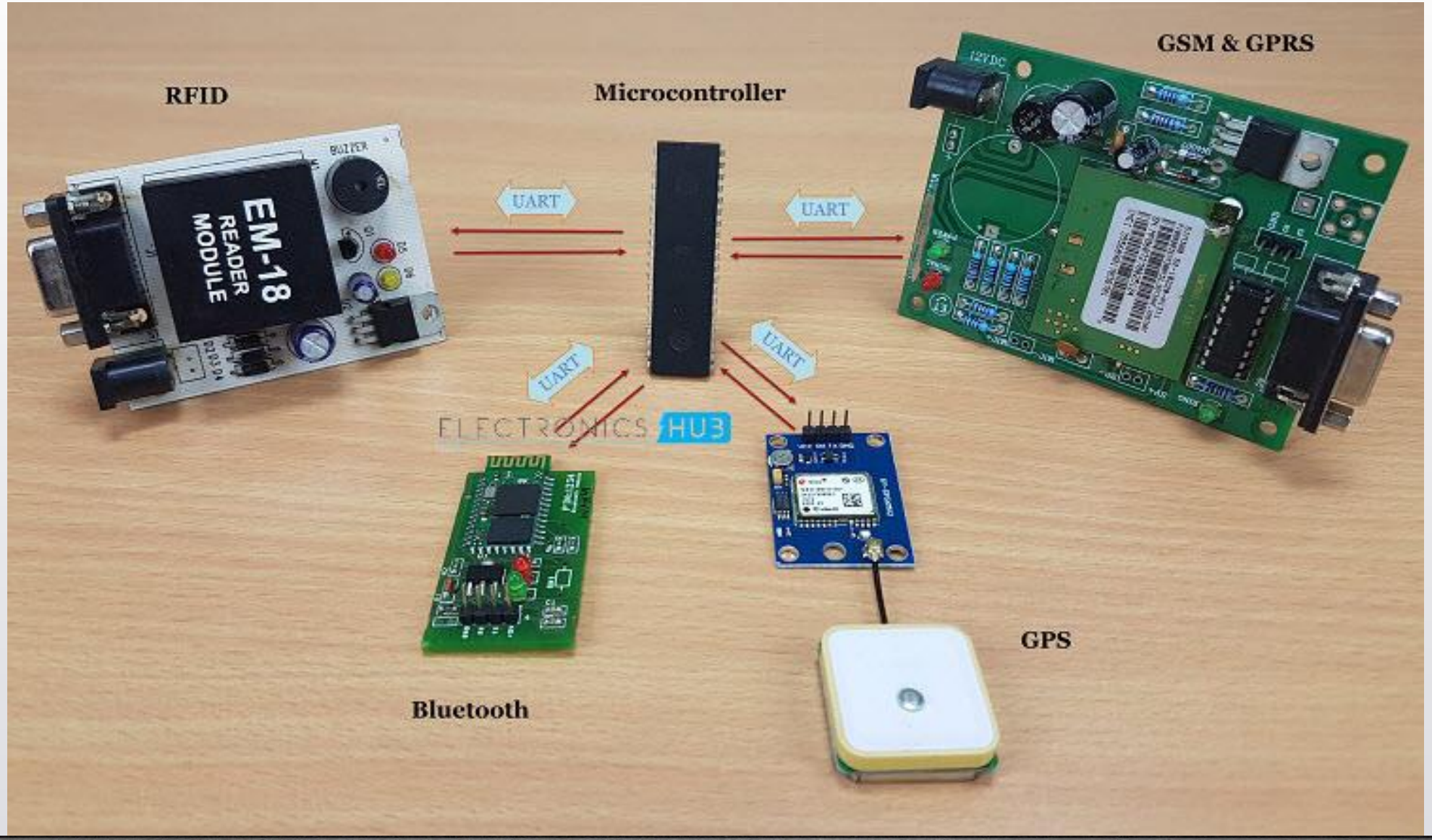
# UART

LPC2148

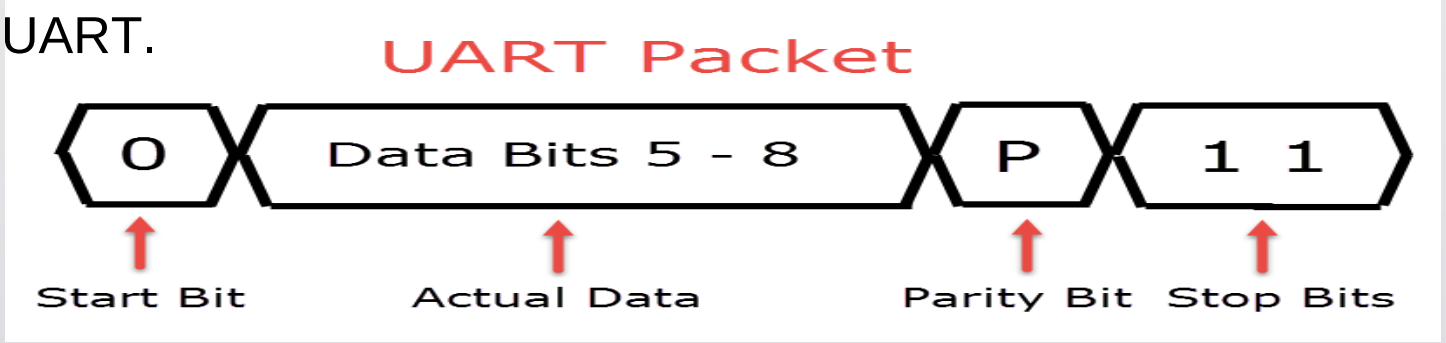
# Introduction

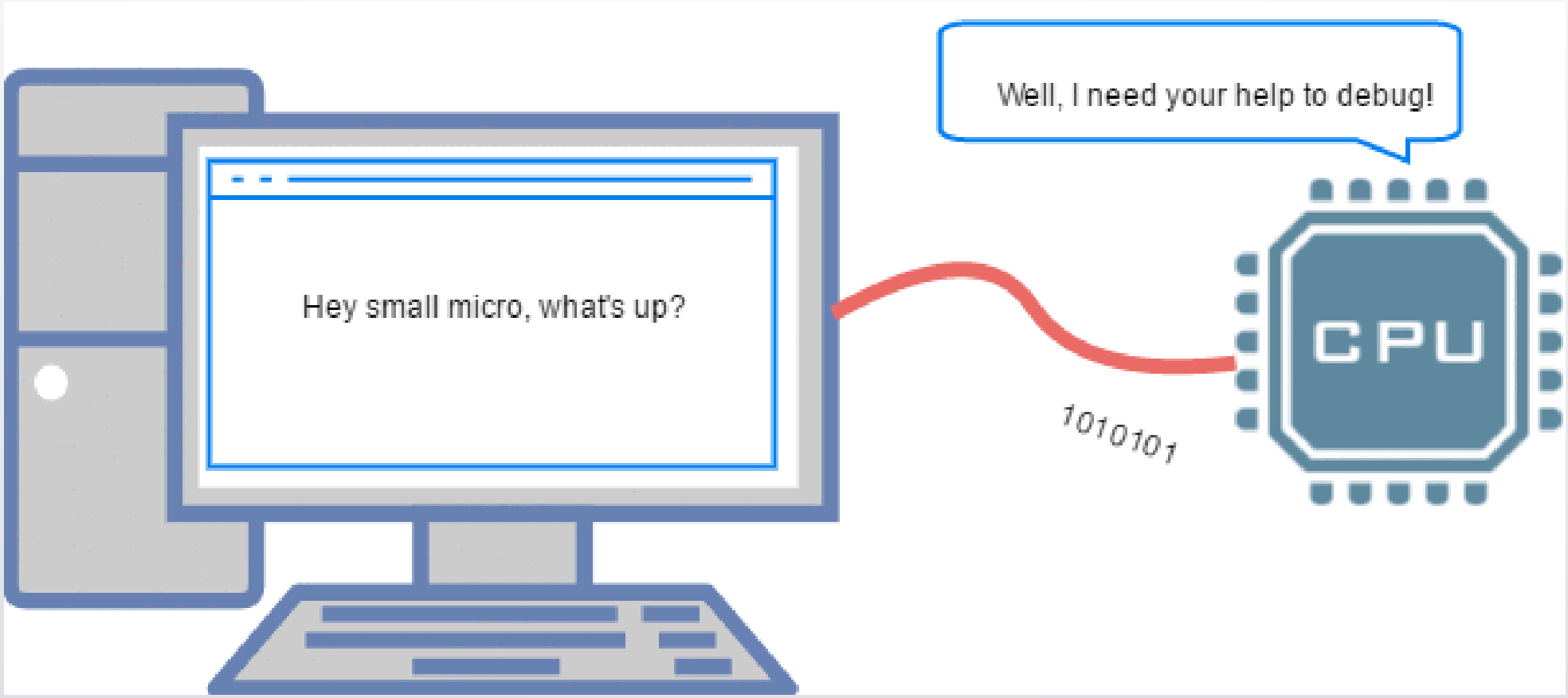
- UART or Universal Asynchronous Receiver Transmitter is a dedicated hardware associated with serial communication. The hardware for UART can be a circuit integrated on the microcontroller or a dedicated IC. This is contrast to SPI or I2C, which are just communication protocols.
- UART is one of the most simple and most commonly used Serial Communication techniques. Today, UART is being used in many applications like GPS Receivers, Bluetooth Modules, GSM and GPRS Modems, Wireless Communication Systems, RFID based applications etc.
- Almost all microcontrollers have dedicated UART hardware built in to their architecture. The main reason for integrating the UART hardware in to microcontrollers is that it is a serial communication and requires only two wires for communication.


RFID
Microcontroller
GSM & GPRS
EM-18
READER
MODULE
BUZZER
UART
UART
UART
UART
ELECTRONICS HUB
Bluetooth
GPS

# Basics of UART

- The UART Protocol uses only two wires (or pins in a device like microcontroller) to transmit the data. In that, one is for transmitting the data and the pin is called TX pin in the device. The other pin is used to receive the data and is called RX pin.
- As UART is a serial communication, the data is transmitted in a series of packets. Usually, a packet consists of 4 parts: a start bit, the actual data, a parity bit and stop bits. The following image shows a typical structure of the data packet in UART.




Hey small micro, what's up?
Well, I need your help to debug!
1010101
CPU

# LPC2148 UART communication

- LPC2148 has two inbuilt UARTs i.e. UART0&UART1.
- UART module and registers. LPC2148 has 2-UARTs numbering 0-3, similarly, the pins are also named as RXD0-RXD1 and TXD0-TXD1.

| Port Pin | Pin Number | PINSEL_FUNC_0 | PINSEL_FUNC_1 | PINSEL_FUNC_2 | PINSEL_FUNC_3 |
|---|---|---|---|---|---|
| P0.0 | 19 | GPIO | **TXD0** | PWM1 | |
| P0.1 | 21 | GPIO | **RXD0** | PWM3 | EINT0 |
| P0_8 | 33 | GPIO | **TXD1** | PWM4 | AD1.1 |
| P0.9 | 34 | GPIO | **RXD1** | PWM6 | EINT3 |

# Features

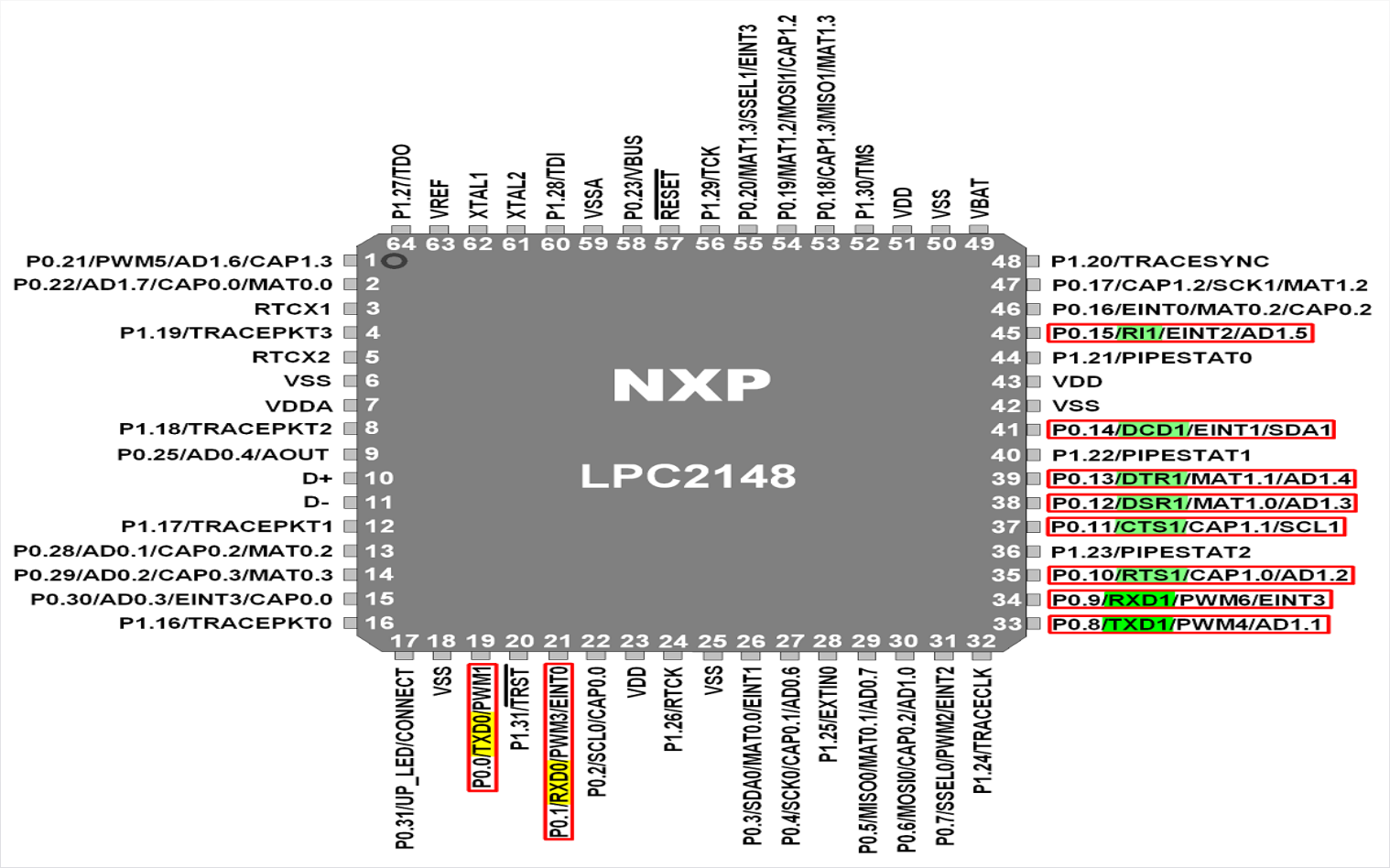
**Features of UART0**

- 16 byte Receive and Transmit FIFOs
- Built-in fractional baud rate generator with autobauding capabilities
- Software flow control through TXEN bit in Transmit Enable Register

**Features of UART1**

- 16 byte Receive and Transmit FIFOs
- Built-in fractional baud rate generator with autobauding capabilities
- Software and hardware flow control implementation possible
- Standard modem interface signals included with flow control (auto-CTS/RTS) fully supported in hardware

# LPC2148 UART Pins



# UART Pins

**UART0**

- **TXD0 (Output pin):** Serial Transmit data pin.
- **RXD0 (Input pin):** Serial Receive data pin.

**UART1 :**

- **TXD1** (Output pin): Serial Transmit data pin.
- **RXD1** (Input pin): Serial Receive data pin.

# UART1 Pins

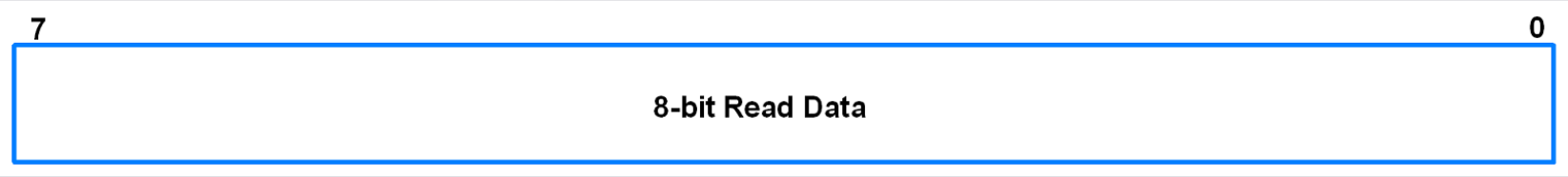
- **RTS1 (Output pin):** Request To Send signal pin. Active low signal indicates that the UART1 would like to transmit data to the external modem.
- **CTS1 (Input pin):** Clear To Send signal pin. Active low signal indicates if the external modem is ready to accept transmitted data via TXD1 from the UART1.
- **DSR1 (Input pin):** Data Set Ready signal pin. Active low signal indicates if the external modem is ready to establish a communication link with the UART1.
- **DTR1 (Output pin):** Data Terminal Ready signal pin. Active low signal indicates that the UART1 is ready to establish connection with external modem.
- **DCD1 (Input pin):** Data Carrier Detect signal pin. Active low signal indicates if the external modem has established a communication link with the UART1 and data may be exchanged.
- **RI1 (Input pin):** Ring Indicator signal pin. Active low signal indicates that a telephone ringing signal has been detected by the modem.

# UART Registers

| Register | Description |
|---|---|
| UxRBR | Contains the recently received Data |
| UxTHR | Contains the data to be transmitted |
| UxFCR | FIFO Control Register |
| UxLCR | Controls the UART frame formatting(Number of Data Bits, Stop bits) |
| UxDLL | Least Significant Byte of the UART baud rate generator value. |
| UxDLM | Most Significant Byte of the UART baud rate generator value. |

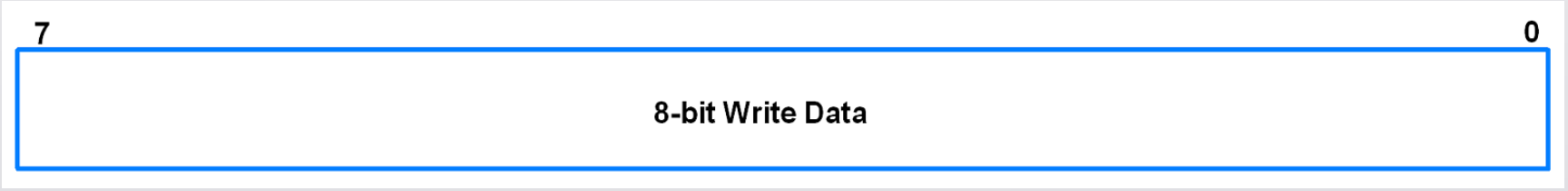
# U0RBR (UART0 Receive Buffer Register)

- It is an 8-bit read only register.
- This register contains the received data.
- It contains the “oldest” received byte in the receive FIFO.
- If the character received is less than 8 bits, the unused MSBs are padded with zeroes.
- The Divisor Latch Access Bit (DLAB) in U0LCR must be zero in order to access the U0RBR. (DLAB = 0)



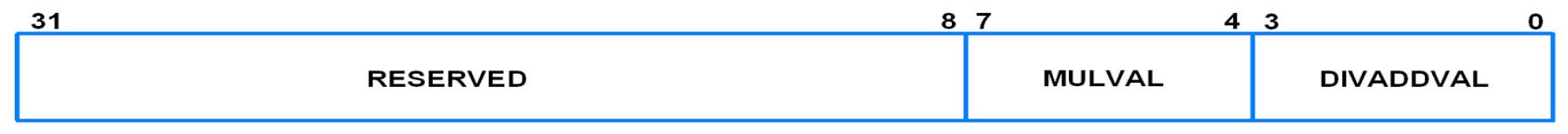
# U0THR (UART0 Transmit Holding Register)

- It is an 8-bit write only register.
- Data to be transmitted is written to this register.
- It contains the "newest" received byte in the transmit FIFO.
- The Divisor Latch Access Bit (DLAB) in U0LCR must be zero in order to access theU0THR. (DLAB = 0)



# U0FDR (UART0 Fractional Divider Register)

- It is a 32-bit read write register.
- It decides the clock pre-scalar for baud rate generation.
- If fractional divider is active (i.e. DIVADDVAL>0) and DLM = 0, DLL must be greater than 3.




- If DIVADDVAL is 0, the fractional baudrate generator will not impact the UART0 baudrate.
- Reset value of DIVADDVAL is 0.
- MULVAL must be greater than or equal to 1 for UART0 to operate properly, regardless of whether the fractional baudrate generator is used or not.
- Reset value of MULVAL is 1.

# U0FDR (UART0 Fractional Divider Register)

- The formula for UART0 baudrate is given below
- UART0 Baudrate = \frac{Pclk}{16 * (256 * U0DLM + U0DLL) * (1 + \frac{DIVADDVAL}{MULVAL})}

- MULVAL and DIVADDVAL should have values in the range of 0 to 15. If this is not ensured, the output of the fractional divider is undefined.
- The value of the U0FDR should not be modified while transmitting/receiving data. This may result in corruption of data.

# U0LCR (UART0 Line Control Register)

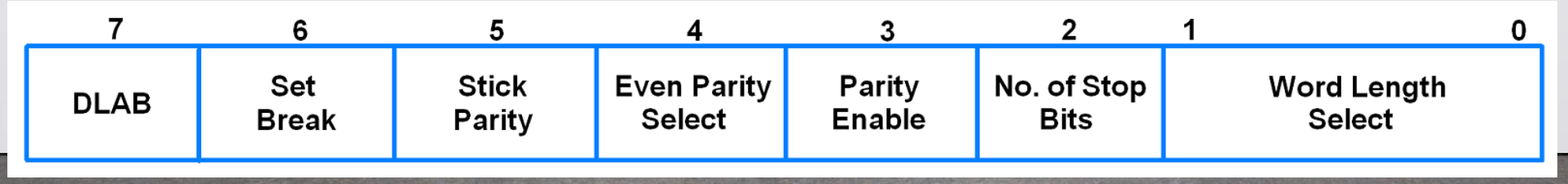
- It is an 8-bit read-write register.
- It determines the format of the data character that is to be transmitted or received.
- **Bit 1:0 - Word Length Select**

  00 = 5-bit character length

  01 = 6-bit character length

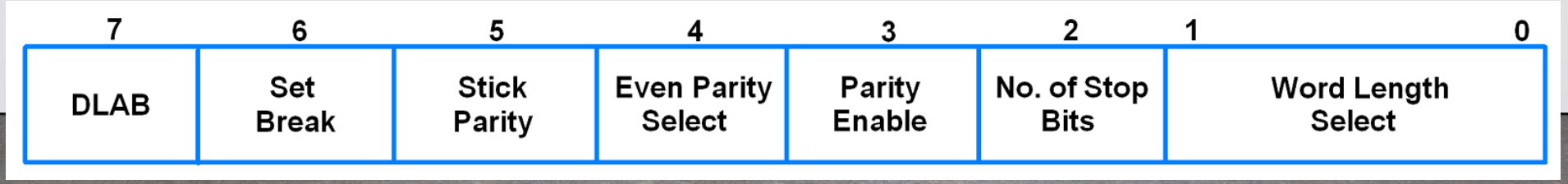
  10 = 7-bit character length

  11 = 8-bit character length



# U0LCR (UART0 Line Control Register)

- **Bit 2 - Number of Stop Bits**
  0 = 1 stop bit
  1 = 2 stop bits
- **Bit 3 - Parity Enable**
  0 = Disable parity generation and checking
  1 = Enable parity generation and checking
- **Bit 5:4 - Parity Select**
  00 = Odd Parity
  01 = Even Parity
  10 = Forced "1" Stick Parity
  11 = Forced "0" Stick Parity
- **Bit 6 - Break Control**
  0= Disable break transmission
  1 = Enable break transmission
- **Bit 7 - Divisor Latch Access Bit (DLAB)**
  0 = Disable access to Divisor Latches
  1 = Enable access to Divisor Latches



# U0DLL and U0DLM (UART0 Divisor Latch Registers)

- U0DLL is the Divisor Latch LSB.
- U0DLM is the Divisor Latch MSB.
- These are 8-bit read-write registers.
- UART0 Divisor Latch holds the value by which the PCLK(Peripheral Clock) will be divided. This value must be 1/16 times the desired baud rate.
- A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed.
- The Divisor Latch Access Bit (DLAB) in U0LCR must be one in order to access the UART0 Divisor Latches. (DLAB = 1)

# Programming of UART0

**1. Initialization of UART0**

- Configure P0.0 and P0.1 as TXD0 and RXD0 by writing 01 to the corresponding bits in PINSEL0.
- Using U0LCR register, make DLAB = 1. Also, select 8-bit character length and 1 stop bit.
- Set appropriate values in U0DLL and U0DLM depending on the PCLK value and the baud rate desired. Fractional divider can also be used to get different values of baudrate.
- Example,
- PCLK = 15MHz. For baud rate 9600, without using fractional divider register, from the baud rate formula, we have,9600 = \frac{15000000}{16 * (256 * U0DLM + U0DLL)} * \frac{MulVal}{MulVal + DivAddVal}
- On reset, MulVal = 1 and DivAddVal = 0 in the Fractional Divider Register.
- Hence, ( 256 x U0DLM + U0DLL ) = \frac{15000000}{16 * 9600}
- ( 256 x U0DLM + U0DLL ) = 97.65
- We can consider it to be 98 or 97. It will make the baud rate slightly less or more than 9600. This small change is tolerable. We will consider 97.
- Since 97 is less than 256 and register values cannot contain fractions, we will take U0DLM = 0. This will give U0DLM = 97.
- Make DLA = 0 using U0LCR register.

# Programming of UART0

- void UART0_init(void) { PINSEL0 = PINSEL0 | 0x00000005; /* Enable UART0 Rx0 and Tx0 pins of UART0 */
- U0LCR = 0x83; /* DLAB = 1, 1 stop bit, 8-bit character length */
- U0DLM = 0x00; /* For baud rate of 9600 with Pclk = 15MHz */
- U0DLL = 0x61; /* We get these values of U0DLL and U0DLM from formula */
- U0LCR = 0x03; /* DLAB = 0 */ }

# Programming of UART0

**2. Receiving character**

- Monitor the RDR bit in U0LSR register to see if valid data is available in U0RBR register.

```
unsigned char UART0_RxChar(void) /*A function to receive a byte on UART0 */
{
    while( (U0LSR & 0x01) == 0); /*Wait till RDR bit becomes 1 which tells that receiver contains
    valid data */
     return U0RBR;
 }
```

# Programming of UART0

3. **Transmitting character**

- Monitor the THRE bit in U0LSR register. When this bit becomes 1, it indicates that U0THR register is empty and the transmission is completed.

```
void UART0_TxChar(char ch) /*A function to send a byte on UART0 */
{
        U0THR = ch; while( (U0LSR & 0x40) == 0 ); /* Wait till THRE bit becomes 1 which tells that
       transmission is completed */
 }
```

# References


- https://www.exploreembedded.com/wiki/LPC2148_UART_Programming
- https://www.electronicwings.com/arm7/lpc2148-uart0
- https://www.electronicshub.org/basics-uart-communication/

# Thank You… ☺