UART Library

The UART hardware module is available with a number of dsPIC30/33 and PIC24 MCUs. The mikroC PRO for dsPIC30/33 and PIC24 UART Library provides comfortable work with the Asynchronous (full duplex) mode.

Important :

- UART library routines require you to specify the module you want to use. To select the desired UART module, simply change the letter **x** in the routine prototype for a number from **1** to **4**.
- Switching between the UART modules in the UART library is done by the UART_Set_Active function (UART modules have to be previously initialized).
- Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

Library Routines

- UARTx_Init
- UARTx_Init_Advanced
- UARTx_Data_Ready
- UARTx_Tx_Idle
- UARTx_Read
- UARTx_Read_Text
- UARTx_Write
- UARTx_Write_Text
- UART_Set_Active

Generic Routines

- UART_Data_Ready
- UART_Tx_Idle
- UART_Read
- UART_Read_Text
- UART_Write
- UART_Write_Text

UARTx_Init

Prototype	`void UARTx_Init(unsigned long baud_rate);`
Description	Configures and initializes the UART module. The internal UART module module is set to: ▪ continue operation in IDLE mode ▪ default Tx and Rx pins ▪ loopback mode disabled ▪ 8-bit data, no parity ▪ 1 STOP bit ▪ transmitter enabled ▪ generate interrupt on transmission end ▪ interrupt on reception enabled ▪ Address Detect mode disabled
Parameters	▪ `baud_rate:` requested baud rate
Returns	Nothing.
Requires	Routine requires the UART module.
Example	`// Initialize hardware UART1 module and establish communication at 2400 bps` `UART1_Init(2400);`
Notes	Refer to the device data sheet for baud rates allowed for specific Fosc. For the dsPIC33 and PIC24 MCUs, the compiler will choose for which speed the calculation is to be performed (high or low). This does not mean that it is the best choice for desired baud rate. If the baud rate error generated in this way is too big then UARTx_Init_Advanced routine, which allows speed select be used. UART library routines require you to specify the module you want to use. To select the desired UART module, simply change the letter **x** in the routine prototype for a number from **1** to **4**. Switching between the UART modules in the UART library is done by the UART_Set_Active function (UART modules have to be previously initialized). Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

UARTx_Init_Advanced

Prototype

```
// dsPIC30 prototype
void UARTx_Init_Advanced(unsigned long baud_rate, unsigned int parity, unsigned int stop_bits);

// dsPIC33 and PIC24 prototype
void UARTx_Init_Advanced(unsigned long baud_rate, unsigned int parity, unsigned int stop_bits, unsigned int high_low_sp
```

Description

Configures and initializes the UART module with user defined settings.

Parameters

- `baud_rate`: requested baud rate
- `parity`: parity and data selection parameter.
  Valid values :

Data/Parity Mode	
Description	**Predefined library const**
8-bit data, no parity	_UART_8BIT_NOPARITY
8-bit data, even parity	_UART_8BIT_EVENPARITY
8-bit data, odd parity	_UART_8BIT_ODDPARITY
9-bit data, no parity	_UART_9BIT_NOPARITY

- `stop_bits`: stop bit selection parameter.
  Valid values :

Stop bits	
Description	**Predefined library const**
One stop bit	_UART_ONE_STOPBIT
Two stop bit	_UART_TWO_STOPBITS

- `high_low_speed`: high/low speed selection parameter. Available only for dsPIC33 and PIC24 MCUs.
  Valid values :

High/Low Speed	
Description	**Predefined library const**
Low Speed UART	_UART_LOW_SPEED
Hi Speed UART	_UART_HI_SPEED

Returns

Nothing.

Requires

Routine requires the UART module.

Example

```
// dsPIC30 family example
// Initialize hardware UART1 module and establish communication at 2400 bps, 8-bit data, even parity and 2 STOP bits
UART1_Init_Advanced(2400, 2, 1);

// dsPIC33 and PIC24 family example
// Initialize hardware UART2 module and establish communication at 2400 bps, 8-bit data, even parity, 2 STOP bits and high speed
UART2_Init_Advanced(2400, 2, 1, 1);
```

Notes

Refer to the device data sheet for baud rates allowed for specific Fosc.

UART library routines require you to specify the module you want to use. To select the desired UART module, simply change the letter **x** in th number from **1** to **4**.

Switching between the UART modules in the UART library is done by the UART_Set_Active function (UART modules have to be previously initi

Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

UARTx_Data_Ready

Prototype

```
unsigned UARTx_Data_Ready();
```

Description

The function tests if data in receive buffer is ready for reading.

Parameters

None.

Returns

- `1` if data is ready for reading
- `0` if there is no data in the receive register

Requires	Routine requires at least one UART module. Used UART module must be initialized before using this routine. See UARTx_Init and UARTx_Init_Advanced routines.

Example	`unsigned receive;`
	`...`
	`// read data if ready`
	`if (UART1_Data_Ready())`
	`  receive = UART1_Read();`

Notes	UART library routines require you to specify the module you want to use. To select the desired UART module, simply change the letter **x** in the routine prototype for a number from **1** to **4**. Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

UARTx_Tx_Idle

Prototype	`char UARTx_Tx_Idle();`
Description	Use the function to test if the transmit shift register is empty or not.
Parameters	None.
Returns	▪ `1` if the data has been transmitted ▪ `0` otherwise
Requires	Routine requires at least one UART module. Used UART module must be initialized before using this routine. See UARTx_Init and UARTx_Init_Advanced routines.

Example	`// If the previous data has been shifted out, send next data:`
	`if (UART1_Tx_Idle() == 1) {`
	`  UART1_Write(_data);`
	`}`

Notes	UART library routines require you to specify the module you want to use. To select the desired UART module, simply change the letter **x** in the routine prototype for a number from **1** to **4**. Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

UARTx_Read

Prototype	`unsigned UARTx_Read();`
Description	The function receives a byte via UART. Use the UARTx_Data_Ready function to test if data is ready first.
Parameters	None.
Returns	Received byte.
Requires	Routine requires at least one UART module. Used UART module must be initialized before using this routine. See UARTx_Init and UARTx_Init_Advanced routines.

Example	`unsigned receive;`
	`...`
	`// read data if ready`
	`if (UART1_Data_Ready())`
	`  receive = UART1_Read();`

Notes	UART library routines require you to specify the module you want to use. To select the desired UART module, simply change the letter **x** in the routine prototype for a number from **1** to **4**. Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

UARTx_Read_Text

Prototype	`void UARTx_Read_Text(char *Output, char *Delimiter, char Attempts);`
Description	Reads characters received via UART until the delimiter sequence is detected. The read sequence is stored in the parameter `output`; delimiter sequence is stored in the parameter `delimiter`.

	This is a blocking call: the delimiter sequence is expected, otherwise the procedure exits (if the delimiter is not found).
Parameters	▪ `Output:` received text ▪ `Delimiter:` sequence of characters that identifies the end of a received string ▪ `Attempts:` defines number of received characters in which `Delimiter` sequence is expected. If `Attempts` is set to 255, this routine will continuously try to detect the `Delimiter` sequence.
Returns	Nothing.
Requires	Routine requires at least one UART module. Used UART module must be initialized before using this routine. See UARTx_Init and UARTx_Init_Advanced routines.

Example	Read text until the sequence "OK" is received, and send back what's been received:
	`UART1_Init(4800);  // initialize UART1 module`
	`Delay_ms(100);`
	`while (1) {`
	`  if (UART1_Data_Ready() == 1) {  // if data is received`
	`  UART1_Read_Text(output, "OK", 10);  // reads text until 'OK' is found`
	`  UART1_Write_Text(output);  // sends back text`
	`  }`
	`}`

Notes	UART library routines require you to specify the module you want to use. To select the desired UART module, simply change the letter **x** in the routine prototype for a number from **1** to **4**. Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

UARTx_Write

Prototype	`void UARTx_Write(unsigned char data);`
Description	The function transmits a byte via the UART module.
Parameters	▪ `data:` data to be sent
Returns	Nothing.
Requires	Routine requires at least one UART module. Used UART module must be initialized before using this routine. See UARTx_Init and UARTx_Init_Advanced routines.

Example	`unsigned char data = 0x1E;`
	`...`
	`UART1_Write(data);`

Notes	UART library routines require you to specify the module you want to use. To select the desired UART module, simply change the letter **x** in the routine prototype for a number from **1** to **4**. Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

UARTx_Write_Text

Prototype	`void UARTx_Write_Text(char * UART_text);`
Description	Sends text via UART. Text should be zero terminated.
Parameters	▪ `UART_text:` text to be sent
Returns	Nothing.
Requires	Routine requires at least one UART module. Used UART module must be initialized before using this routine. See UARTx_Init and UARTx_Init_Advanced routines.
Example	Read text until the sequence "OK" is received, and send back what's been received:

```
UART1_Init(4800);                       // initialize UART1 module
Delay_ms(100);

while (1) {
  if (UART1_Data_Ready() == 1) {        // if data is received
    UART1_Read_Text(output, "OK", 10);  // reads text until 'OK' is found
    UART1_Write_Text(output);           // sends back text
  }
}
```

Notes	UART library routines require you to specify the module you want to use. To select the desired UART module, simply change the letter **x** in the routine prototype for a number from **1** to **4**. Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

UART_Set_Active

Prototype	`void UART_Set_Active(unsigned (*read_ptr)(), void (*write_ptr)(unsigned char _data), unsigned (*ready_ptr)(), unsigned (*tx_idle_ptr)());`
Description	Sets active UART module which will be used by UARTx_Data_Ready, UARTx_Read and UARTx_Write routines.
Parameters	Parameters : ▪ `read_ptr`: UARTx_Read handler ▪ `write_ptr`: UARTx_Write handler ▪ `ready_ptr`: UARTx_Data_Ready handler ▪ `tx_idle_ptr`: UARTx_Tx_Idle handler
Returns	Nothing.
Requires	Routine is available only for MCUs with multiple UART modules. Used UART module must be initialized before using this routine. See UARTx_Init and UARTx_Init_Advanced routines.

Example	`UART1_Init(9600);  // initialize UART1 module`
	`UART2_Init(9600);  // initialize UART2 module`
	
	`RS485Master_Init();  // initialize MCU as Master`
	
	`UART_Set_Active(&UART1_Read, &UART1_Write, &UART1_Data_Ready, &UART1_Tx_Idle); // set UART1 active`
	`RS485Master_Send(dat,1,160);  // send message through UART1`
	
	`UART_Set_Active(&UART2_Read, &UART2_Write, &UART2_Data_Ready, &UART2_Tx_Idle); // set UART2 active`
	`RS485Master_Send(dat,1,160);  // send through UART2`

Notes	None.

UART_Data_Ready

Prototype	`unsigned UART_Data_Ready();`
Description	The function tests if data in receive buffer is ready for reading. This is a generic routine which uses the active UART module previously activated by the UART_Set_Active routine.
Parameters	None.
Returns	▪ `1` if data is ready for reading ▪ `0` if there is no data in the receive register
Requires	Routine requires at least one UART module. Used UART module must be initialized before using this routine. See UARTx_Init and UARTx_Init_Advanced routines.

Example	`unsigned receive;`
	`...`
	`// read data if ready`
	`if (UART_Data_Ready())`
	`  receive = UART_Read();`

Notes	Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

UART_Tx_Idle

Prototype	`char UART_Tx_Idle();`
Description	Use the function to test if the transmit shift register is empty or not. This is a generic routine which uses the active UART module previously activated by the UART_Set_Active routine.
Parameters	None.
Returns	▪ `1` if the data has been transmitted ▪ `0` otherwise
Requires	Routine requires at least one UART module. Used UART module must be initialized before using this routine. See UARTx_Init and UARTx_Init_Advanced routines.

Example	`// If the previous data has been shifted out, send next data:`
	`if (UART_Tx_Idle() == 1) {`
	`  UART_Write(_data);`
	`}`

Notes	Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

UART_Read

Prototype	`unsigned UART_Read();`
Description	The function receives a byte via UART. Use the UART_Data_Ready function to test if data is ready first. This is a generic routine which uses the active UART module previously activated by the UART_Set_Active routine.
Parameters	None.
Returns	Received byte.
Requires	Routine requires at least one UART module. Used UART module must be initialized before using this routine. See UARTx_Init and UARTx_Init_Advanced routines.

Example	`unsigned receive;`
	`...`
	`// read data if ready`
	`if (UART_Data_Ready())`
	`  receive = UART_Read();`

Notes	Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

UART_Read_Text

Prototype	`void UART_Read_Text(char *Output, char *Delimiter, char Attempts);`
Description	Reads characters received via UART until the delimiter sequence is detected. The read sequence is stored in the parameter `output`; delimiter sequence is stored in the parameter `delimiter`. This is a blocking call: the delimiter sequence is expected, otherwise the procedure exits (if the delimiter is not found). This is a generic routine which uses the active UART module previously activated by the UART_Set_Active routine.
Parameters	▪ `Output:` received text ▪ `Delimiter:` sequence of characters that identifies the end of a received string ▪ `Attempts:` defines number of received characters in which `Delimiter` sequence is expected. If `Attempts` is set to 255, this routine will continuously try to detect the `Delimiter` sequence.
Returns	Nothing.
Requires	Routine requires at least one UART module. Used UART module must be initialized before using this routine. See UARTx_Init and UARTx_Init_Advanced routines.

Example	Read text until the sequence "OK" is received, and send back what's been received:
	`UART1_Init(4800);  // initialize UART1 module`
	`Delay_ms(100);`
	
	`while (1) {`
	`  if (UART_Data_Ready() == 1) {  // if data is received`
	`  UART_Read_Text(output, "OK", 10);  // reads text until 'OK' is found`
	`  UART_Write_Text(output);  // sends back text`
	`  }`
	`}`

Notes	Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

UART_Write

Prototype	`void UART_Write(unsigned char data);`
Description	The function transmits a byte via the UART module. This is a generic routine which uses the active UART module previously activated by the UART_Set_Active routine.
Parameters	▪ `data:` data to be sent
Returns	Nothing.
Requires	Routine requires at least one UART module. Used UART module must be initialized before using this routine. See UARTx_Init and UARTx_Init_Advanced routines.

Example	`unsigned char data = 0x1E;`
	`...`
	`UART_Write(data);`

Notes	Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

UART_Write_Text

Prototype	`void UART_Write_Text(char * UART_text);`
Description	Sends text via UART. Text should be zero terminated. This is a generic routine which uses the active UART module previously activated by the UART_Set_Active routine.
Parameters	▪ `UART_text:` text to be sent
Returns	Nothing.
Requires	Routine requires at least one UART module. Used UART module must be initialized before using this routine. See UARTx_Init and UARTx_Init_Advanced routines.

Example	Read text until the sequence "OK" is received, and send back what's been received:
	`UART1_Init(4800);  // initialize UART1 module`
	`Delay_ms(100);`
	
	`while (1) {`
	`  if (UART_Data_Ready() == 1) {  // if data is received`
	`  UART_Read_Text(output, "OK", 10);  // reads text until 'OK' is found`
	`  UART_Write_Text(output);  // sends back text`
	`  }`
	`}`

Notes	Number of UART modules per MCU differs from chip to chip. Please, read the appropriate datasheet before utilizing this library.

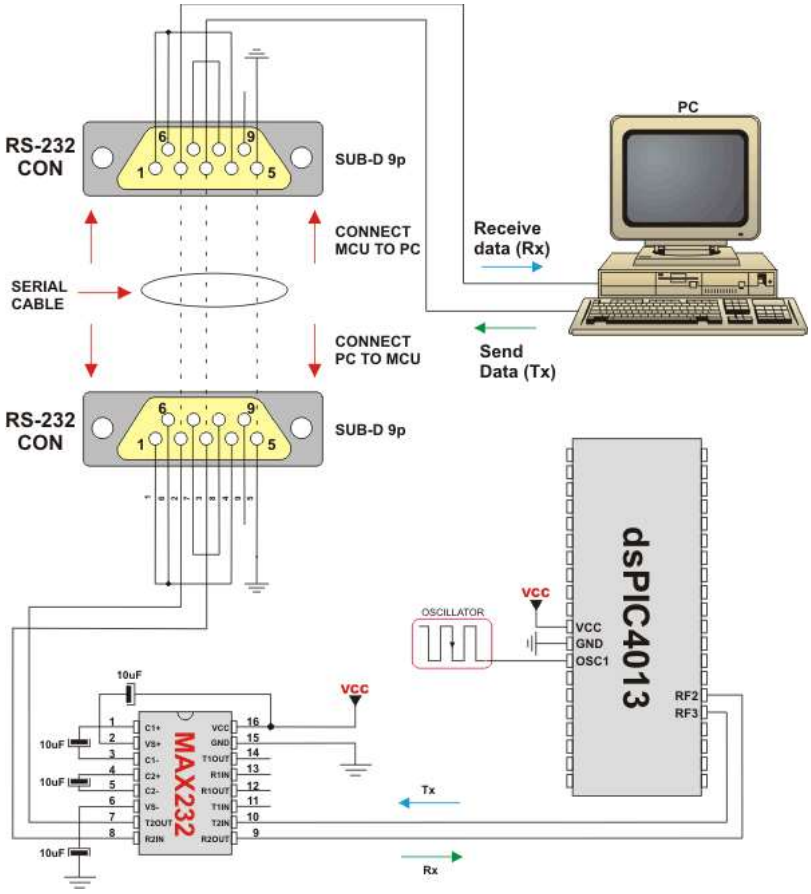
Library Example

This example demonstrates simple data exchange via UART. If MCU is connected to the PC, you can test the example from the mikroC PRO for dsPIC30/33 and PIC24 USART communication terminal, launch it from the drop-down menu **Tools › USART Terminal** or simply click the USART Terminal Icon .

Copy Code To Clipboard

```
char uart_rd;

void main() {

  UART1_Init(9600);               // Initialize UART module at 9600 bps
  Delay_ms(100);                  // Wait for UART module to stabilize

//  U1MODEbits.ALTIO = 1; // un-comment this line to have Rx and Tx pins on their alternate
                          // locations. This is used to free the pins for other module, namely the SPI.

  UART_Write_Text("Start");
  UART_Write(10);
  UART_Write(13);

  while (1) {                     // Endless loop
    if (UART_Data_Ready()) {      // If data is received,
      uart_rd = UART_Read();      // read the received data,
      UART_Write(uart_rd);        // and send data via UART
    }
  }
}
```

HW Connection



RS232 HW connection