# Usage

**Usage**

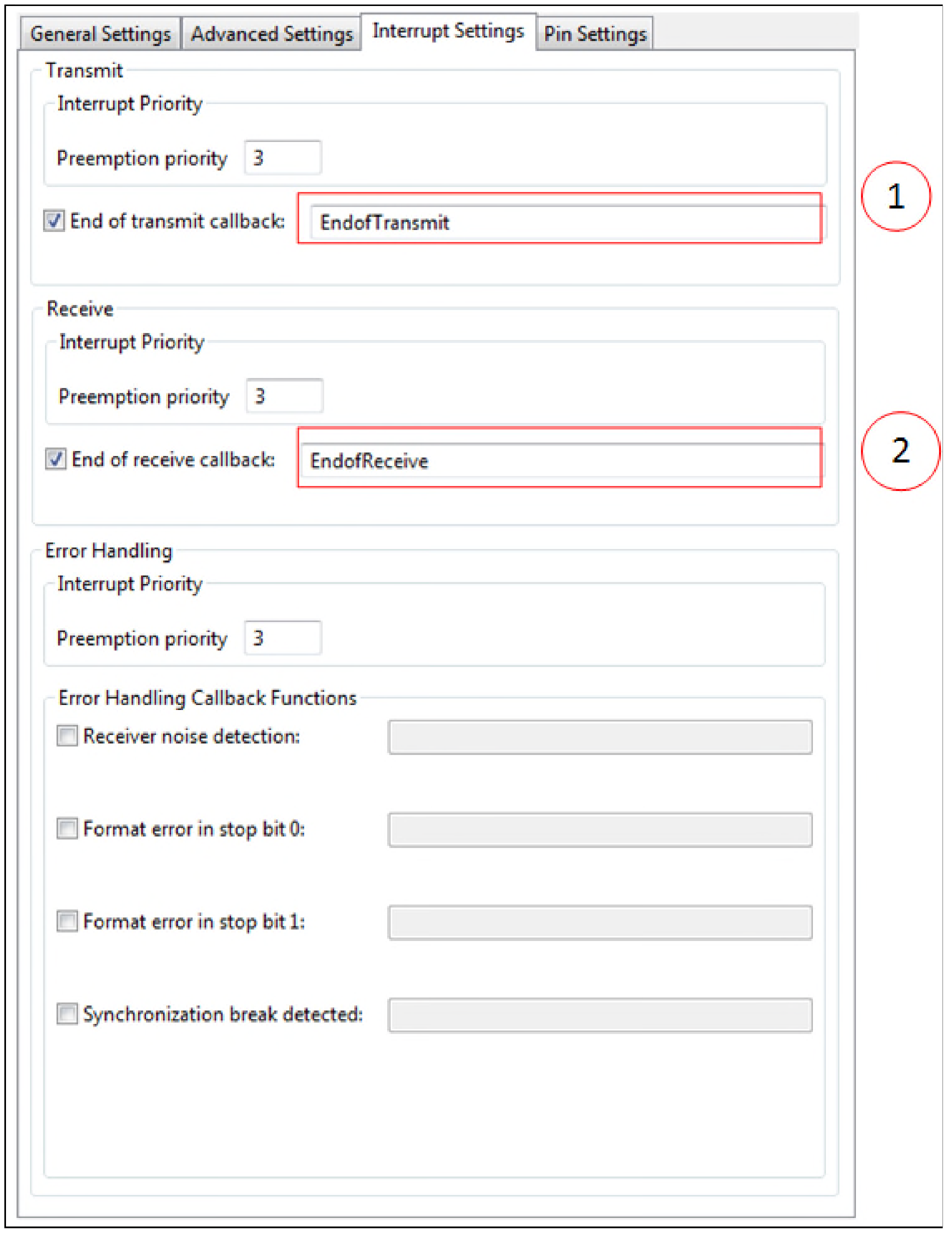
*This example demonstrates the serial communication using **UART** APP. Hyperterminal application can be used for communicating data between the example application on device and the user. The application transmits a text string on start up. It then waits for 10 bytes of data input. At this point, the user should transmit 10 bytes of data to the device. After receiving the user input data of 10 bytes, the application will re-transmit the received data. For every 10 bytes of data transmitted by the user, the same data will be sent back. This example application does not have any blocking code. Callback functions are configured to be executed on end of transmission and on end of reception.*

The following description gives a stepwise guide to implement this example.

**Instantiate the required APP**
Drag an instance of **UART** APP.

**Configure the APP**
**UART** APP:

- Default configuration for the tabs "General Settings", "Advanced Settings" and "Pin Settings".
- Define callback function names in the "Interrupt Settings" tab as shown below.



Transmit

Interrupt Priority

Preemption priority 3

☑ End of transmit callback: EndofTransmit

Receive

Interrupt Priority

Preemption priority 3

☑ End of receive callback: EndofReceive

Error Handling

Interrupt Priority

Preemption priority 3

Error Handling Callback Functions

☐ Receiver noise detection:

☐ Format error in stop bit 0:

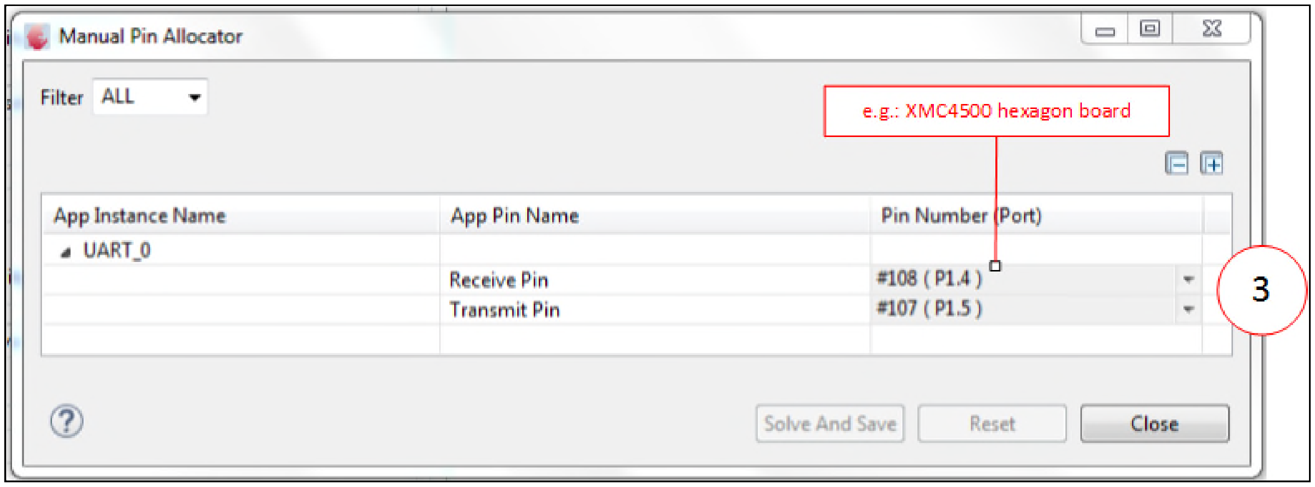
☐ Format error in stop bit 1:

☐ Synchronization break detected:

1. Enable "End of transmit callback" checkbox. Input a valid function name in the textbox next to it. Enter "EndofTransmit".
2. Enable "End of receive callback" checkbox. Input a valid function name in the textbox next to it. Enter "EndofReceive".

**Note:** When the checkbox is enabled, a popup message will be displayed. This message indicates that the text in the textbox is not a valid function name. Close the warning popup by clicking the "OK" button at the bottom of the popup. This warning will be cleared when a valid function name is entered in the textbox.

**Manual pin allocation**


Manual Pin Allocator

Filter: ALL

e.g.: XMC4500 hexagon board

| App Instance Name | App Pin Name | Pin Number (Port) |
|---|---|---|
| UART_0 | | |
| | Receive Pin | #108 ( P1.4 ) |
| | Transmit Pin | #107 ( P1.5 ) |

3. Select the pins for data transmission and reception.
   **Note:**The pins are specific to the development board chosen to run this example. The pins shown in the image above may not be available on every XMC boot kit. Ensure that a proper pin is selected according to the board.

**Generate code**
Files are generated here: `<project_name>/Dave/Generated/' (`project_name' is the name chosen by the user during project creation). APP instance definitions and APIs are generated only after code generation.

**Note:** Code must be explicitly generated for every change in the GUI configuration.
**Important:** Any manual modification to APP specific files will be overwritten by a subsequent code generation operation.

**Sample Application (main.c)**

```
#include <DAVE.h>


uint8_t data[] = "Infineon Technologies";
uint8_t rec_data[10];
int main(void)
{
  DAVE_STATUS_t status;

  status = DAVE_Init();      /* Initialization of DAVE APPs  */

  if(status == DAVE_STATUS_FAILURE)
  {
    /* Placeholder for error handler code. The while loop below can be replaced with an user error
handler */
    XMC_DEBUG(("DAVE Apps initialization failed with status %d\n", status));
    while(1U)
    {
    }
  }

  UART_Transmit(&UART_0, data, sizeof(data) - 1); //Transmit the string "Infineon Technologies".

  while(1U)
  {
  }

  return 1;
}

void EndofTransmit()//Callback functin for "End of transmit" event.
{
 UART_Receive(&UART_0, rec_data, sizeof(rec_data));
}

void EndofReceive()//Callback function for "End of receive" event.
```

```
{
 UART_Transmit(&UART_0, rec_data, sizeof(rec_data));
}
```

**Build and Run the Project**

**Observation**

- Open hyperterminal application. Choose the COM port to which the device **UART** pins are connected. Set the configuration as following,
    - baudrate = 19200
    - data bits = 8
    - number of stop bits = 1
    - no parity

  Open communication port. When the application is downloaded, the first message on the console will be displayed as, "Infineon Technologies". Enter a string of 10 bytes. The same string will be transmitted back by the device.