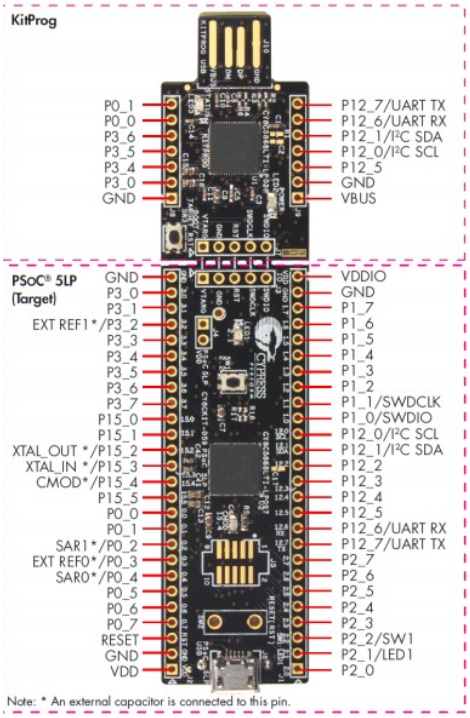
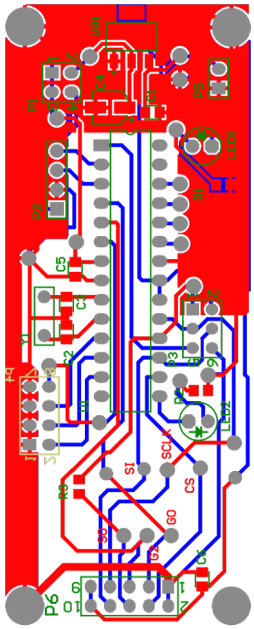
Laboratory 4

COMSYS 301 : Design : Hardware Software Systems

Objectives:

- To set up and program a UART block to receive a serial stream from an RF receiver. The RF stream is composed of localisation data from a simulated motion of a robot.
- Use the USBUART block to display PSoC generated data on a PC.
- Transfer the data to Matlab to visualize the simulated motion.





1. Abstract of the tasks

- Add an existing project to the current workspace
- Using a terminal program to view data sent from the PSoC
- Program a UART to receive a serial stream (ASCII or binary, selectable with a jumper)
- Visualize the data

2. The Serial Stream

The ASCII Stream

The simplest is a serial stream and hence will tackled first in this exercise. This stream is generated by the AVRMEGA8 (on the provided RF module) by inserting a jumper across pins 1-2 in P4 (`cc2500_mega8.pdf`). The structure of this stream is as shown below.

"*#rssi,index,[xloc,yloc,angle10],[ghost0_X,ghost0_Y,ghost0_SPEED,ghost0_DIR],[ghost1],[ghost2]*"

rss1: received signal strength indicator, signed 8 bit, good signal closer to 0dB, poorer signal closer to -120dB

index: packet number, continuously incrementing, unsigned 8bit, [0 to 255]

[xloc,yloc,angle10]:Robot's x, y location in pixels, angle10 is the robot angle multiplied by 10

[ghost0_X,ghost0_Y,ghost0_SPEED,ghost0_DIR]: ghost0's localization data

[ghost1]: Same as ghost0

[ghost2]: same as ghost0

For e.g.

#-50,57,[2000,2550,100],[33,300,0,4],[500,900,20000,3]

#-52,58,[2000,2550,100],[35,300,0,4],[500,905,20000,3]

Background: The RF module receives a RF packet from a PC. This packet is then restructured and retransmitted as a serial stream on the AVR's UART-TX line. The RF module has two LEDS; the red one is the power indicator and the green LED indicates when the RF module is receiving data packets. If its all working well, both LEDs will be lighted and the TX pin of the RF module will be show a digital stream if it is probed on an oscilloscope. If data are not received, the green LED will not be lighted and the serial stream will send a "#NORX".

The Binary Stream

If the jumper across pins 1-2 in P4 is removed, the RF module will transmit the data in a binary format instead of ASCII characters.

Background: The localisation data is stored as a datatype of `struct` . To ensure the start of packet (SOP) is correctly identified (i.e. syncronised) each packet is preceded with two bytes. These bytes have been defined using -

```
#define SOP 0xaa
```

The struct is defined as

```
typedef struct data_main {
      int8        rssi;             // Received signal strength indicator in dB
```

```
    uint8       index;              // index number of packet. incremented number
    uint16      robot_xpos;         //
    uint16      robot_ypos;         //
    uint16      robot_orientation;
    uint16      g0_xpos;            //
    uint16      g0_ypos;            //
    uint16      g0_speed;           //
    uint16      g0_direction;       //
    uint16      g1_xpos;            //
    uint16      g1_ypos;            //
    uint16      g1_speed;           //
    uint16      g1_direction;       //
    uint16      g2_xpos;            //
    uint16      g2_ypos;            //
    uint16      g2_speed;           //
    uint16      g2_direction;       //
} vtype1;
```

Comments on Methodology

The data packets are transmitted regularly (sort-off regularly!!). Since the RF stream is an independent process (technical speak – "on a different clock domain") it is impossible to know when the data will actually be transmitted. Hence, polling the serial receive register/flag is a inefficient (not impossible) way to pick up the stream. The correct way is to use an *interrupt*. The provided PSoC project has the provisions for an interrupt based solution. If you choose this methodology, will need to code the interrupt service routine and some support code in `main.c` to complete this task.

3. Tasks

In this lab, instructions will be minimal. Refer to lab1 if you don't remember how to achieve a task.

3.1. Opening the workspace

As in lab 1, begin by opening the psoc_intro.cywrk workspace.

3.2. Download lab4 from CANVAS

Extract it in the folder containing the previous lab experiments.

Add the project to the workspace

Make the 'lab4' project active.

3.3. Viewing and exploring the schematic

Open lab4.cydwr, look at the *Pins* tab. How many in/out puts are there?

Open TopDesign.cysch, look at all the components. Look at their configuration parameters.

The schematic contains two main objects: a UART and a USBUART. The UART is a general purpose serial communication block while the USBUART is a bridge between a USB device and a UART. The USBUART can be used to transfer data between the PSoC and a PC. On examining the code, you will find that a simple but sufficient software interface is provided. You could build on this interface to create a custom interface.

Examine the schematic and artwork of the provided RF receiver module. The RF receiver will re-transmit (wired connection) this data using its own UART connected to Port P2. Naturally, to exchange data between data-source and a data-sink, the transmit pin of one must be connected to a receive pin of the the other.

3.4. Electrical Connections (refer to the Kit-059 schematic and the RF Module Schematic)

Until now you have been using only one USB port – Port J10 ("USB Finger Connector"). In this exercise, the USBUART Bridge uses the USB pins of the PSoC (Target device on the Kit-059). These are connected the other USB port – Port J6 (USB Micro-B connector). Hence, you will need make use of both the USB ports; one for programming and the other for data exchange between the PsoC and a PC

Disconnect the PSoC from both of the USB cables.

1. Connecting GND
    a. Identify the GND pin of the RF module (one of two pins in the white 2-pin polarized connector next to "Led1".
    b. Identify the GND pin on the Kit-059 (on J1, besides P1.7).
    c. Connect the two GND using any one of provided wires.
2. Connecting the positive supply
    a. Identify the Vcc pin of the RF module (on the white 2 pin polarized connector next to "Led1".
    b. Identify the VDD pin on the Kit-059 (on J1, besides P1.7).
    c. Connect the Vcc and VDD using another wire.
3. Connecting the serial terminal
    a. Identify the Tx pin of the RF Module (Port 2, pin 3)
    b. Identify the Rx pin on the PSoC (P2.2)
    c. Connect the Tx and Rx pins using the third wire.
    d. Ensure a jumper is inserted across pins 1-2 in P4. With jumper in place the RF module sends the RF data as ASCII but without the jumper, the binary stream will be sent.
4. Connect both the USB cables to the PSoC and the PC.

3.5. Test with terminal

Before you begin, it is advisable to check that your PSoC is communicating with a terminal program (PuTTy for example).

1. Start a terminal program.
2. Configure it using the following settings: 57.6k baud, disable flow control, one stop bit, echo off.
3. Connect both the USB cables and lab4 programmed into the PsoC. (For this test, the RF module may be connected or could also be disconnected).
4. Establish a connection between the terminal and the Psoc.
5. Type "qwerty34" on the keyboard while the terminal window is selected. Note that what you are typing will not be displayed directly on the screen but it is sent nonetheless. The PSoC should echo "you entered: qwerty34".

3.6. Changing the main.c file

An examination of the project files will show

1. Three components: UART, USBUART and a LED
2. 4 inputs/outputs
3. 'main.c' containing support for the USBUART: can send a string to a terminal and receive a string from the the terminal
4. Blank functions (prototypes) provided for the UART. You have write something.

Hint: An interupt will be generated everytime a character is received. You need to write some code, probably in 'isrRF_RX.c' and in the 'main.c' to handle the receipt of each character. You could use the LED terminal P2.1 and a scope to help understand and debug. Keep in mind the end use of this code and what overhead it introduces.

4. Your notes

You are encouraged to take notes of the quick questions and remarks made in the lab because some of them may be in the quiz.