Lab 1: Basics of LabVIEW and USRP

Report Due: 21:00, 3/17, 2017

1 Overview

In this course, we would like to learn how communication systems work from labs. For this purpose, LabVIEW is used to simulate these systems, and USRP is used to implement these systems in hardware. This lab prepares you for the future labs and help you get familiar with these tools. Furthermore, the concept of packet transmission is also introduced in this lab.

The block diagram of a communication system is shown in Fig. 1, which summarizes the relations among the four labs we are going to do this semester.

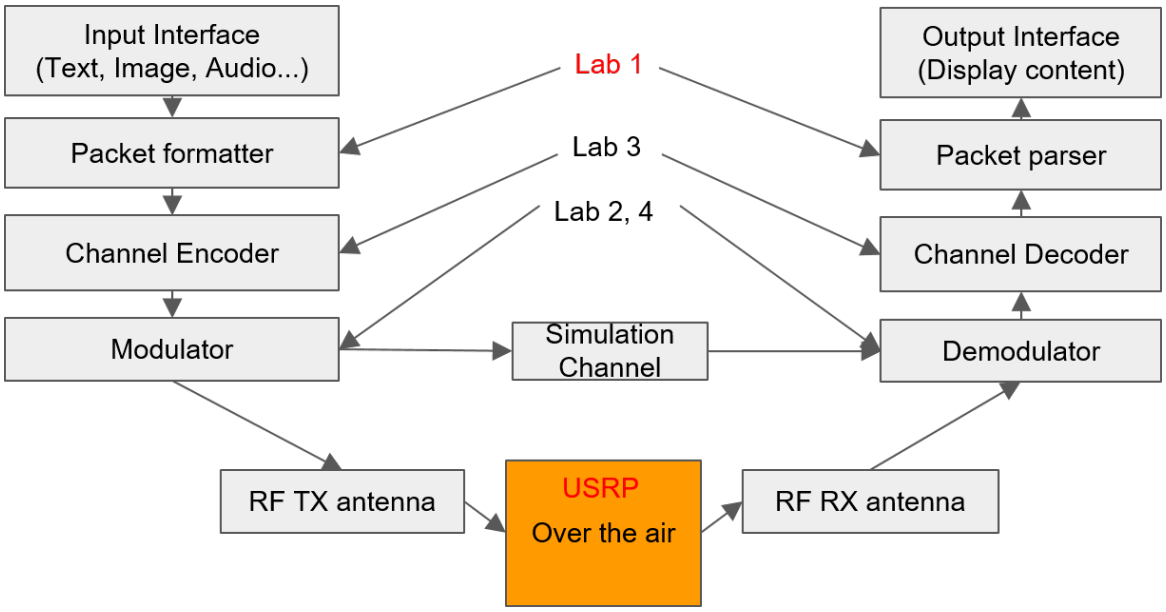


Figure 1: Block Diagram of the Four Labs

In this lab, we have two objectives. First, we learn how to convert message to bit-level and construct packets in the specified format. Second, we transmit the information through the simulation channel and the real channel. We will first use LabVIEW to simulate the system and then use LabVIEW to control USRP to transmit over-the-air.

2 Experiments

2.1 LabVIEW Simulation

There are two systems we need to build in this section. The first one consists of a *packet formatter* and a *packet parser*. The second one is a *baseband communication system simulator*.

2.1.1 Packet Formatter and Packet Parser

Introduction In modern communication systems, a message to be transmitted or received are often partitioned into segments. Receiver have to know how long a segment of message is in order to put all segments back together. Therefore, additional information will be added to each segment of message at the transmitter. Such additional information is stored in the so-called *header*, and the message segment is called *payload*. After adding a header to the paylaod, the overall data structure is called a *packet*.

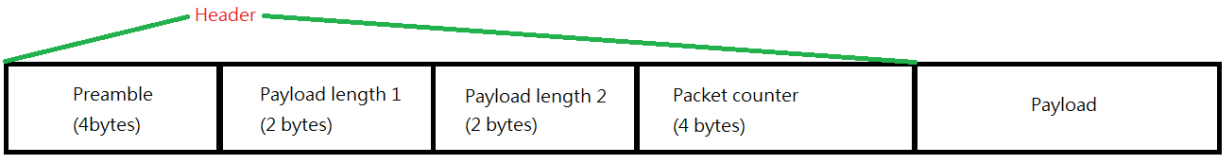


Figure 2: Header format in lab 1

In this lab, the header consists of three fields: 1) preamble, 2) payload length and 3) packet counter. Fig. 2 illustrates the allocated bytes (8 bits) for each field. The purpose of each field is listed as follow:

- **Preamble:** Preamble is a known bit sequence to both the transmitter (Tx) and the receiver (Rx). With preamble, Rx can detect the starting time of a packet. Also, preamble is often used to help Rx fix channel impairments. This will be covered in Lab 2.
- **Payload length:** Payload length indicates how many bytes are carried in the payload field. It is converted to a 16-bit sequence. For example, if the length of payload is 4 bytes (32 bits), the 16-bit payload length is "(LSB)0010000000000000(MSB)". This field is duplicated twice to serve as a checksum. As a result, the total 32 bits of payload length is "00100000000000000010000000000000". If the decoded payload length of the first and second part are matched, a packet is assumed correct. Otherwise, the receiver will declare error and drop the received packet.
- **Packet counter:** This field tells Rx the index of this packet. After receiving all packets, Rx sorts packets back according to these indexes and reconstructs the message.

After introducing the header of a packet, we can briefly explain the procedures to generate a packet. A packet is generated through the following steps:

1. **Source conversion:** Information sources such as texts, images, video, etc., are first converted into a byte array. The bytes that each packet carries is then determined at Tx, which forms the payload of a packet.
2. **Header generation:** The length of the payload is converted to a 32-bit sequence, following the rule described above. Along with packet counter, the header is then generated and attached in front of the payload.

3. **From bytes to bits:** Before passing the packet to the modulator, bytes are converted to bits following standard rules.

Next, we turn to packet parser, which uses header of a packet to extract the information in the payload. The procedure introduced here is easier to implement for LabVIEW beginners. However, you are encouraged to think about better and faster methods to extract packets.

1. **Search for preamble (bits):** Since Rx knows what preamble is, it searches this special sequence from all bits it received.

2. **Extract the header:** If the preamble sequence is found, the next 32 bits are assumed to be the field that represent the payload length. Next, the 32 bits are split into two 16-bit sequences and compared with each other. If the two payload length are matched, the next 32 bits are collected as the packet counter and the rest $8 \times$ payload length bits are collected as the payload. Otherwise, Rx declares an error and does nothing.

Assignment Templates of the packet formatter and parser are already included in the project file. The template for Formatter is `lab1_packet_formatter.vi` and the one for Parser is `lab1_packet_parser.vi` respectively. Please implement these two blocks.

- **Formatter:** The inputs to this block are: 1) Preamble bytes array, 2) Payload bytes array, and 3) Packet counter (unsigned integer). Refer to the format shown in Fig. 2 and generate header accordingly. Next, you have to combine header and payload to generate a packet. Finally, convert bytes to bits before connecting to output port.

- **Parser:** The inputs to this block are: 1) Bitstream and 2) Preamble sequence (bytes). Your job is to use this preamble sequence to search from the received bitstream. If a sequence is equal to preamble, then a packet is assumed to be detected. After detecting a packet, you should check the header and extract payload. Remember to convert payload bits into bytes array.

2.1.2 Advenced techniques

Before we begin to introduce the second system, we would like to show you some useful techniques in developing communication systems with LabVIEW and USRP. The advanced LabVIEW programming techniques are: 1) Queues, 2) Global Variable, and 3) Event Structure. These techniques will help you understand the design of the second system more clearly. In the rest of this part, we will use an example to practice these techniques.

Example: Sine-wave signal generator Fig. 3 is the block diagram of the signal generator. This program is also included in the LabVIEW project file. The following steps explain how to build this example:

1. **While loops:** The first step is to create two while loops, one for transmitter and the other for receiver. In the rest of this part, transmitter loop and receiver loop are called "tx loop" and "rx loop" for convenience.

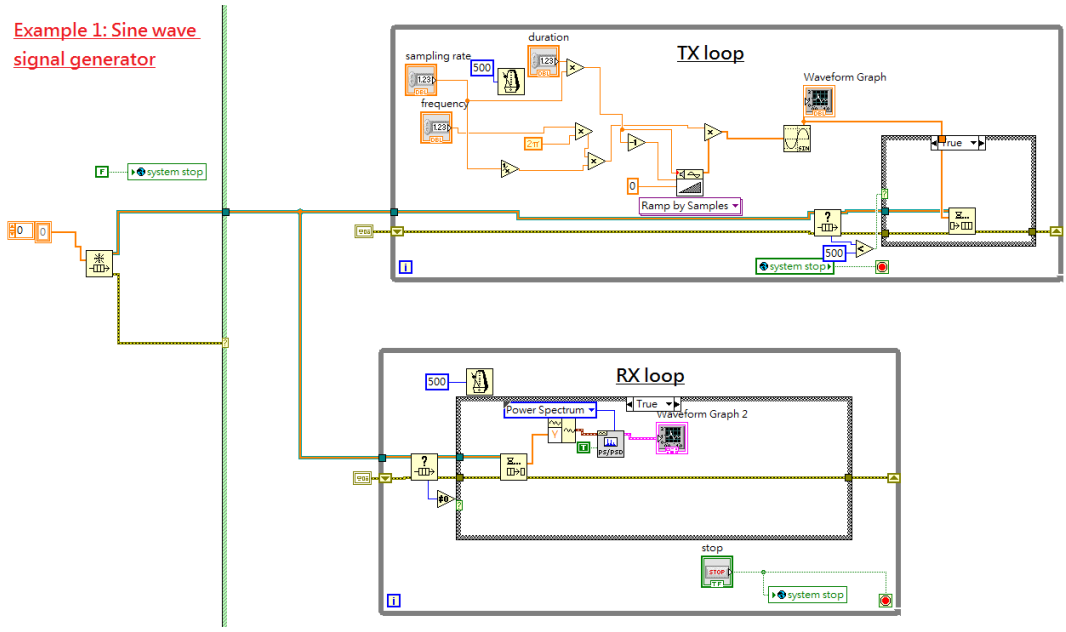


Figure 3: block diagram for example 1

2. **Global variable and event structure:** As you learn from basic LabVIEW programming, the stopping conditions for different loops are independent. However, to make our program clean and convenient to use, there are two approaches to stop multiple loops with single control: 1) global variable and 2) event structure. We use the first method in this example. The block diagram, as a result, is shown in Fig. 4. As for the second method, you can refer to the attached LabVIEW example `event_structure.vi`.

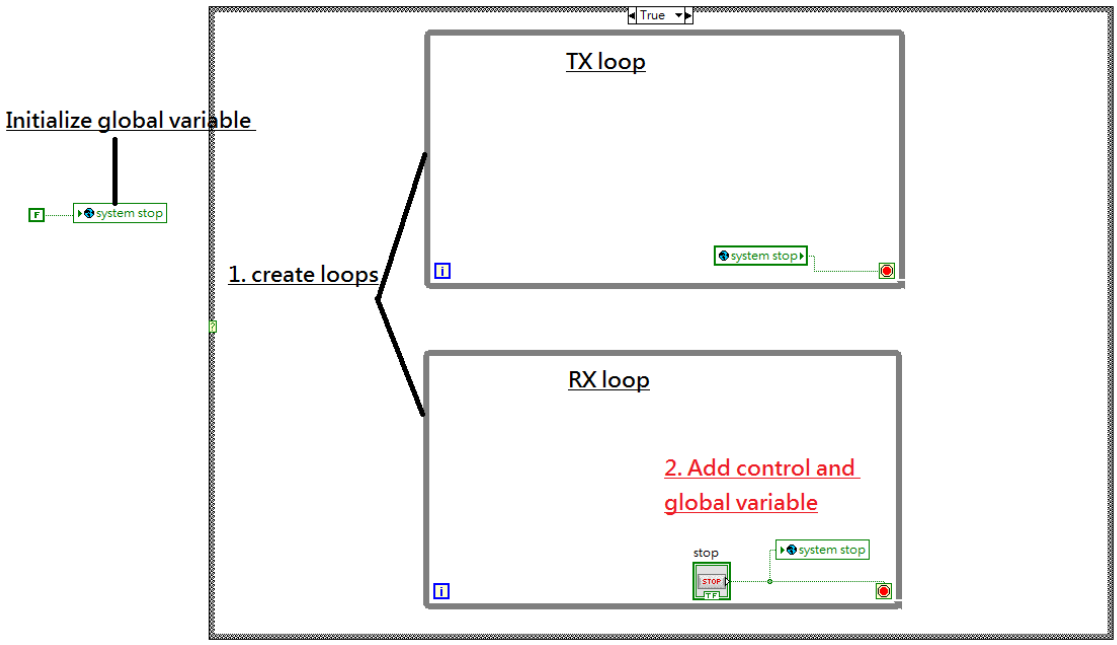


Figure 4: step 1,2 of example 1

3. **Queue module:** This is a module enables us to pass data from one loop to the other.

In our case, queue module will be a tunnel to pass sine wave generated from tx loop to rx loop. The following are some advices when using queue modules: (see Fig. 5)

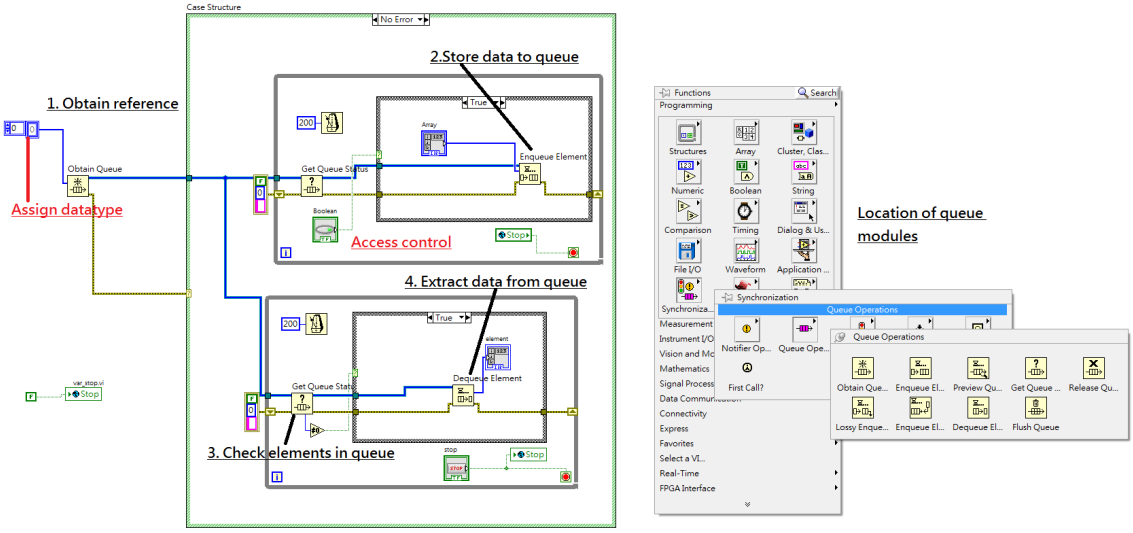


Figure 5: Queue modules

- **Pass by reference:** Output of "Obtain queue" is in fact a reference to certain location of your computer memory. So when releasing a queue, it is enough to call "release queue" just once.
- **Query before access:** Use "Get queue status" before enqueue/dequeue data to make your system stable.
- **Initialize error:** Use shift register to record status of queue. Make sure shift registers are initialized to default status. It can be done simply by right click on shift register, and select "add constant."

4. **Signal processing blocks:** After completing the structure of tx loop and rx loop, we are ready to build signal processing blocks to generate the sine wave. There are three parameters for this simulation: 1) duration, 2) sampling rate, and 3) frequency.

   - **Duration T_d:** This is the duration of observation time of each iteration in our simulator. The unit is second.
   - **Sampling rate f_s:** Specify how many samples generated per second.
   - **Frequency f_0:** The frequency of the sine wave.

   Mathematically, in tx loop, we generate sample according to:

$$s(n) = \sin\left(2\pi f_0 \frac{n}{f_s}\right) \quad \text{for } 0 \le n \le (f_s T_d - 1)$$

   At receiver side, we observe the power spectrum of the transmitted signal. To do this, we use `FFT power spectrum and PSD.vi` located in "Signal Processing/Waveform Measurements" and add a "waveform graph" to observe the results. Fig 7 is the block diagram

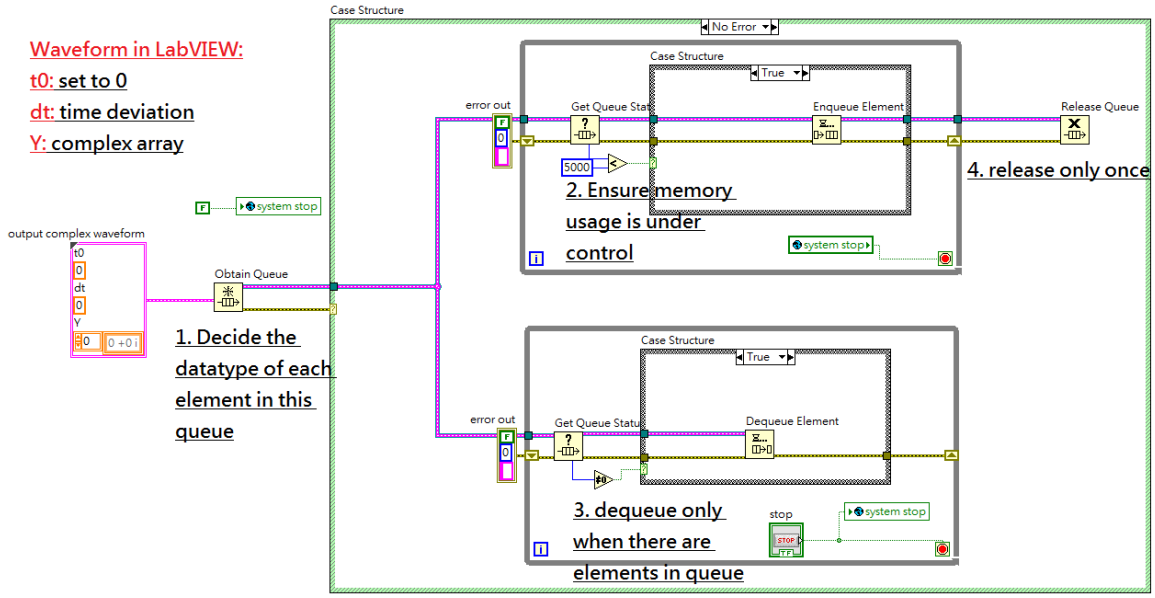


Figure 6: step 3 of example 1

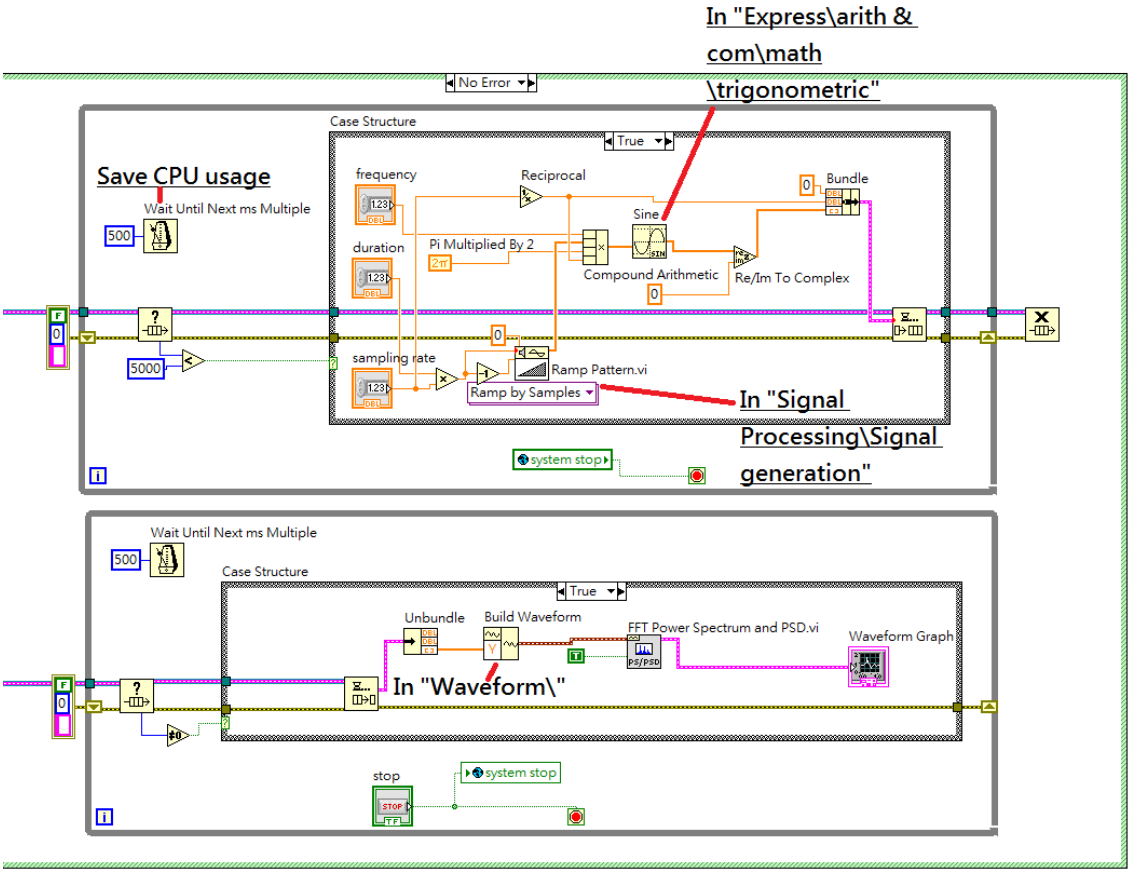


Figure 7: step 4 of example 1

after connecting required blocks. Note that “Wait Until Next ms Multiple” are added to

avoid too much CPU consumption.

5. **Control panel:** After the block diagram part of this example is complete, we can now test our program in the control panel. Fig. 8 is a result with parameters: $f_0 = 10^4$, $f_s = 4 \times 10^4$, $T_d = 10^{-1}$.

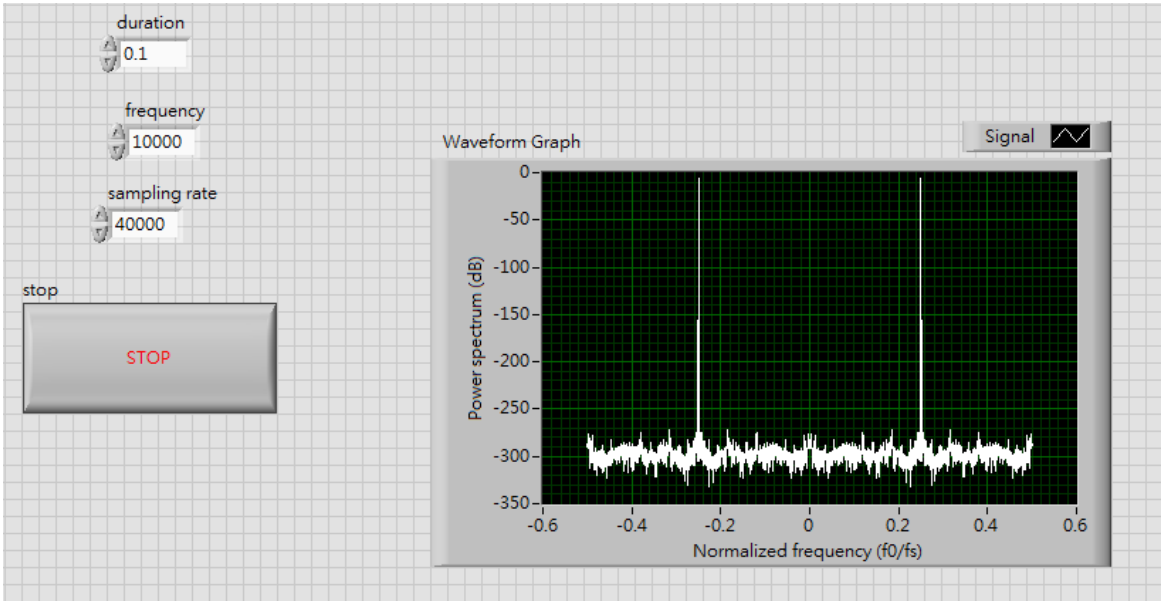


Figure 8: result of example 1

2.1.3 Baseband communication system simulator

After the packet formatter and parser are finished, we can test them with the simulator. `lab1_byte_packet_generator.vi` is the file for this part. The simulator is built with LabVIEW built-in modules. The rest are organized as follow: First we will explain the structure of this system. Then we will introduce some important blocks with their parameters. The aim here is to make you understand how to use built-in LabVIEW modules. As for the principles or algorithms inside these blocks, most of them will be introduced in later labs.

The overall structure of the communication system is shown in Fig. 9. The architecture of the system can be divided into six parts. Each part are implemented as a loop. We will not explain the details of each loop. The goal of this part is to learn how to use this simulator. Therefore, we will explain the parameters of some LabVIEW blocks. Modules to be explained are listed below. All of them are located in "RF communications/Modulation/Digital":

- `MT Generate System Parameters.vi`: Use this module to specify the modulation scheme you want to use for a system. The inputs include: samples per symbol, M-ary and other options for certain modulation. The outputs include a symbol map which is a complex array of size M. The purpose of the input "samples per symbol" is for pulse shaping filtering purposes. This block is often used in system configuration part as shown in Fig 9.

- `MT Generate Filter Coefficients.vi`: This module will generate the coefficients of some commonly used pulse shaping filters, including Raised cosine, Root-raised cosine

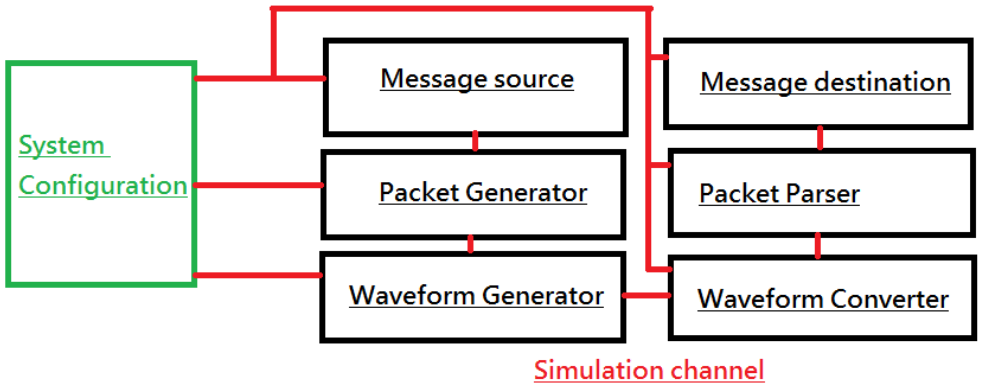


Figure 9: Structure of the Second Simulator

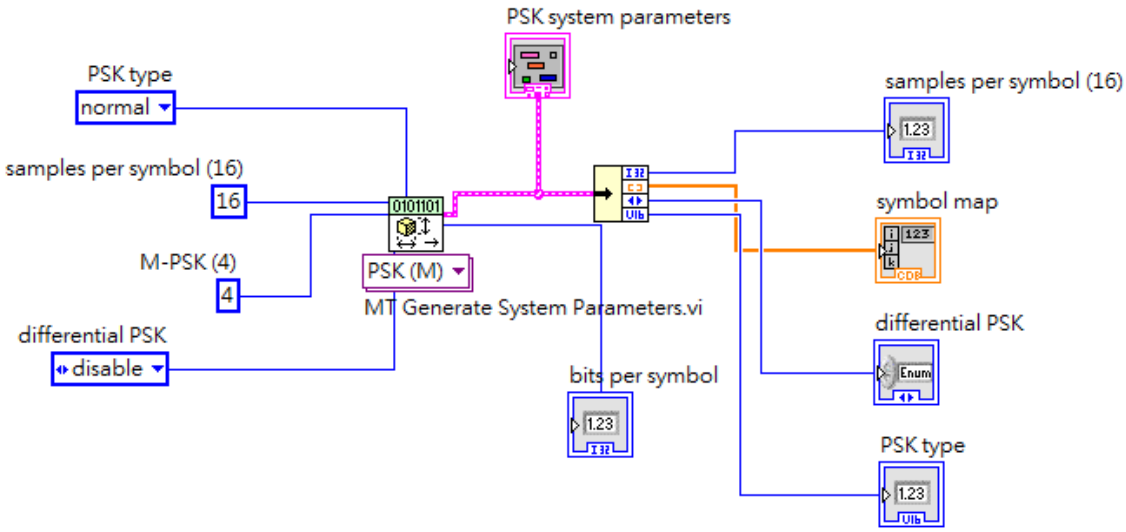


Figure 10: `MT Generate System Parameters.vi`

filters, etc.. Note that both pulse shaping and matched filter coefficients will be generated once the samples per symbol parameter is specified for both of them. The variable "filter parameter" can change the roll-off factor of pulse shaping filters. You can find some examples by pressing "Ctrl+h -> detail help." A handy tip is that you can reuse the "samples per symbol" in system parameters to this block.

- `MT Modulate XXX.vi`: There are many types of modulation you can select. All of them will transform bit-stream according to symbol map generated in `Generate System Parameters.vi`. The input "symbol rate" is optional. But when using USRP, it must be specified. For that case, you can divide "IQ-rate" by "samples per symbol" to obtain symbol rate. "Reset" is for the case when IIR-filtering is required for your application. Specifically, set it to "true" for FIR-filters and "false" for IIR-filters.

- `MT Demodulate XXX.vi`: This block is the counterpart of "MT modulate XXX." It will map complex waveform to bit stream. The inputs are easy to understand except for "synchronization parameters." However, this is an optional input, and it is ignored in our example.

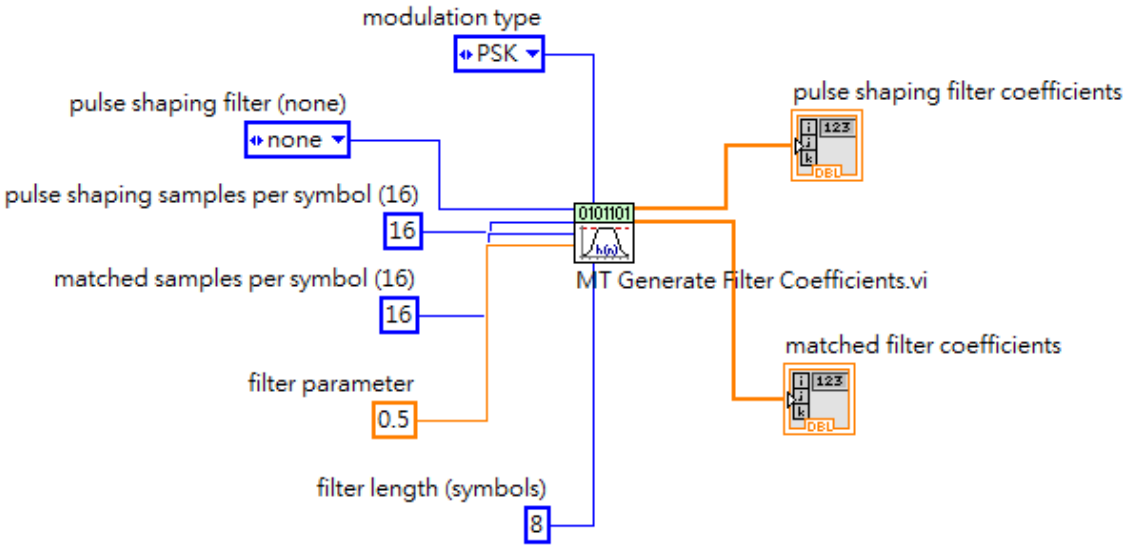


Figure 11: `MT Generate Filter Parameters.vi`

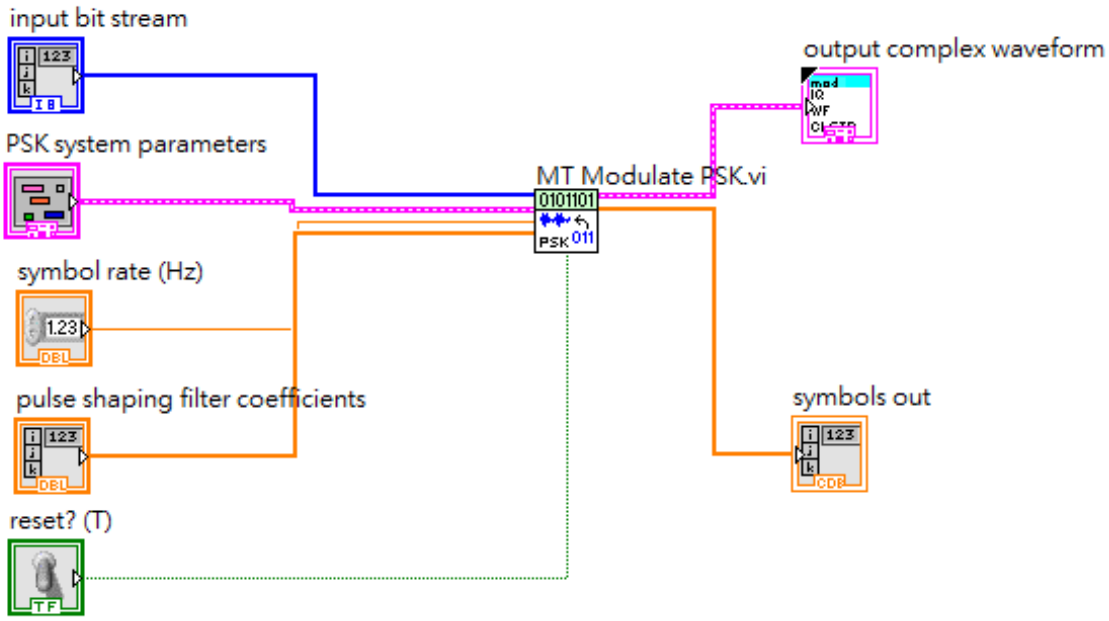


Figure 12: `MT Modulate PSK.vi`

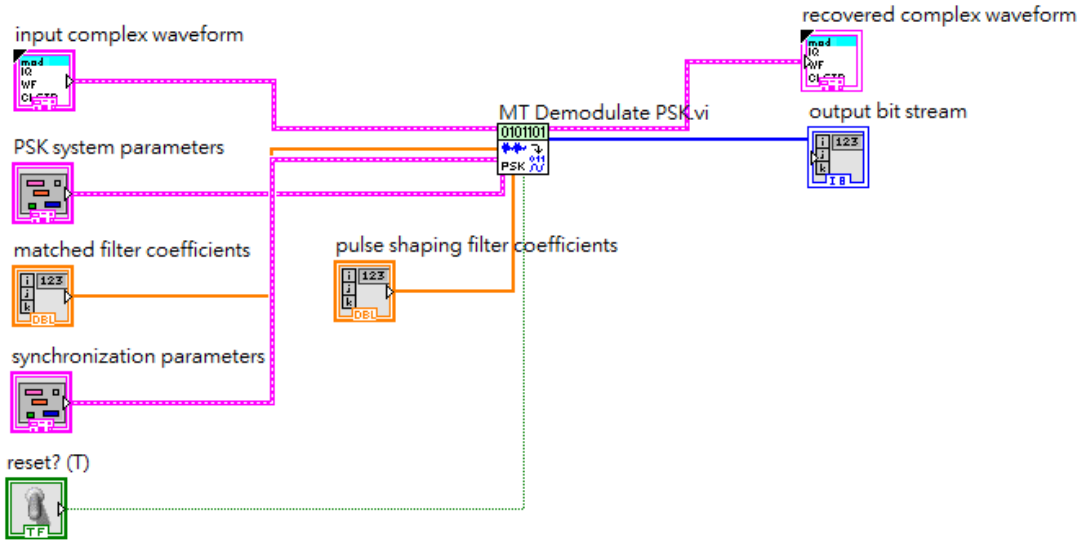


Figure 13: `MT Demodulate PSK.vi`

Now, we briefly explain the design of this example. First, in system configuration part, we use system parameter and pulse shaping generator to create coefficients and symbol map that will be used for Waveform generator/converter loops. Second, the message source loops are created. In this case, we use texts as source. After converting texts to bytes arrays, we use queues to pass bytes stream to packet formatter. The operation of packet formatter is the same as in previous section. Thirdly, bit-stream of packets are passed to waveform generator to form waveform. Note that this is a simulation example. So a simulation channel `Add AWGN noise.vi` is placed here to include the effect of channel noise. The rest of the signal flow is reverse to how waveforms are generated. Fig 8. is a result from the simulator.

In summary, we can build a communication system simulator with built-in LabVIEW modules. The details behind the blocks remain unknown. Therefore, in the following labs, it is our goal to show you how to implement these blocks by yourselves. Furthermore, you will also build advanced modules that LabVIEW does not have.

Assignment Use the parameters listed in Table 1 to run the program `lab1_byte_packet_transceiver.vi`.

Set	E_b/N_0	Pulse Shaping	M-PSK	Samples per Symbol	Filter Length
1	10	root raised cosine	2	4	11
2	10	root raised cosine	4	4	11
3	15	raised cosine	4	4	11

Table 1: Simulation Parameters

2.2 USRP Implementation

After the simulation part is complete, we can use USRP to do experiment through over-the-air transmission. The goal is to learn how to use USRP modules. There are two parts of this

section. The first part is to show you how to reprogram a LabVIEW simulation into an USRP experiment program. The second part is the USRP version of the baseband communication system simulator.

For these examples, the recommended RF parameters are: Carrier frequency=915MHz, number of samples=20000.

2.2.1 USRP sine-wave signal generator

To reprogram a LabVIEW simulation to a USRP experiment, two more loops are required. These loops are the interfaces between USRP and computer. In the rest of this section, we call these two additional loops "USRP TX" and "USRP RX" respectively. The following steps will convert example 1 to USRP experiment system.

1. **Use USRP modules:** The USRP modules are located in "Instrument I/O/Instrument Devices/NI-USRP." The commonly used modules are Rx and Tx. These two categories contain the configuration modules and data pass modules. Some advanced configuration can be setup through "NI-USRP Property Node." For example, MIMO cable setting and Local oscillator frequency are some properties can be set through that block.

2. **Create USRP configuration controls:** In this example, we are going to integrate one transmit antenna and one receive antenna to our program. Three control should be added and connect to USRP configuration block: IQ rate, carrier frequency and gain. In the bottom of the block, the port named "active antenna" are a string to select one of the two antennas of USRP. The name of them are written on the front side of USRP hardware, which is "TX1/RX1" and "RX2," respectively.

3. **Create USRP TX/RX loops:** After completing the configuration of USRP, we can create two more loops in our program. One loop is for the data ready to send to USRP and transmit to the air. The other is for the signal received from the air and passed back to computer. Note the in USRP RX loop, the control "number of samples" is a parameter that tell USRP the size of data required to passed to computer. Combined with while loop, this can be viewed as the processing rate of the system.

4. **Modify TX/RX loop:** After loops for USRP are created, we return to TX and RX loops to add modifications. First, use queue to exchange waveforms between USRP loops and signal processing loop. Second, since processing rate is slower, the time counter module can be removed. Fig. 17 shows the modifications of first step. Fig. 18 shows the modifications on signal processing blocks.

5. **Modify Signal processing loops:** The last step is to replace sampling rate parameter originally connected to simulator with USRP IQ rate.

Assignment

1. **Implementation:** Refer to Section 2.2.1 and convert
   `lab1_example_sine_signal_generator.vi`
   to an USRP program.

2. **Experiment:** Use the parameters listed in Table 2. Paste the power spectrum in the report and compare the differences.

Set	IQ rate	Sine Frequency	duration
1	400k	40k	0.01
2	200k	40k	0.01
3	400k	100k	0.01

Table 2: Parameters for USRP example 1

2.2.2 USRP packet transceiver

The program is similar to the one in Section 2.1.3. The connections of inputs and outputs are already done. Fig. 19 is the structure of the program. The program is provided to you as a template system. In later labs, additional blocks will be added to it. Note that in "Packet formatter" loop, the block you implemented is connected. Similarly, "Packet parser" is also connected to proper inputs/outputs. Make sure the blocks are complete before running this program.

Assignment Change "M-PSK" with $M \in \{2, 4, 8\}$ and paste "Constellation map" in the report. Fig. 20 shows how to store data from LabVIEW graph. This is for those who want to make a figure with other tools such as MATLAB or gnuplot.

3 Exercises

- LabVIEW
  1. Complete all ***Assignment*** in Section 2. Please provide your source code.
  2. Transform the specified message into the bit-level by looking ASCII table, and compare it with the provided bits. Refer to `lab1_commlab_is_fun.vi`.
  3. Try to read profile `lab1_exercise_pkt_bitstream.txt`, and recover the original message. Show your recovered message on the report.
  4. Follow our format to build packets, and try to transmit packets on simulated channel. Compare the original message and recovered message under different noise power.
- USRP
  1. Use USRP to transmit packets. Compare the original message and recovered message under different signal power. Use `lab1_usrp_packet_transceiver.vi` for this exercise. You can change transmit power by the control "Transmit power(dB)."
  2. Choose different parameters on pulse shaping filter, and try to explain the impact of parameters on frequency-domain. What is the role of this filter?
  3. (Optional) Try to interfere the reliability of USRP transmission on the specified parameter setting: IQ rate=200k, Carrier frequency=915M, gain=0. Explain the method that you use.

4 Lab Report

There is no format requirements for your lab report. In the report, you should address the results of the exercises mentioned above. You should also include your simulation program in the appendix of the report. Include whatever discussions about the new findings during the lab exercise, or the problems encountered and how are those solved. Do not limit yourself to the exercises specified here. You are highly encouraged to play around with your simulation program on self-initiated extra lab exercises/discussions.

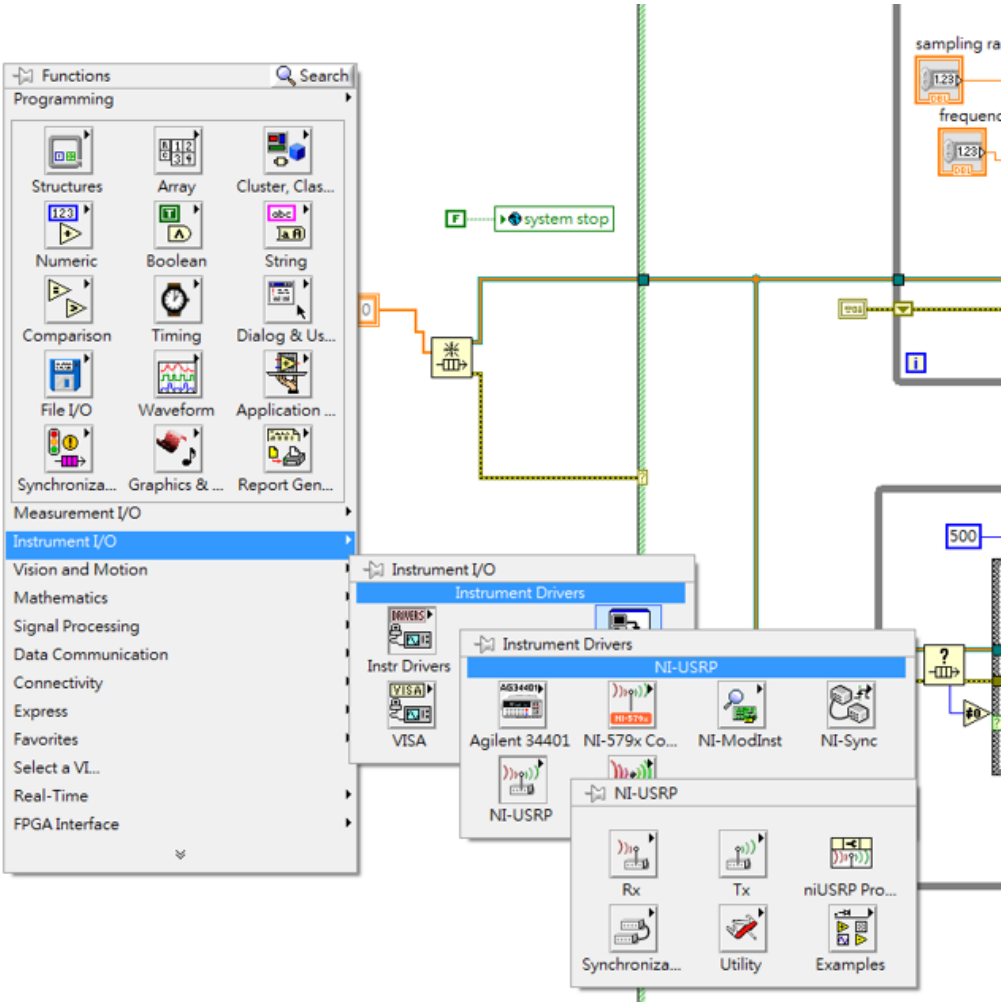


Figure 14: Step 1 of USRP reprogram example

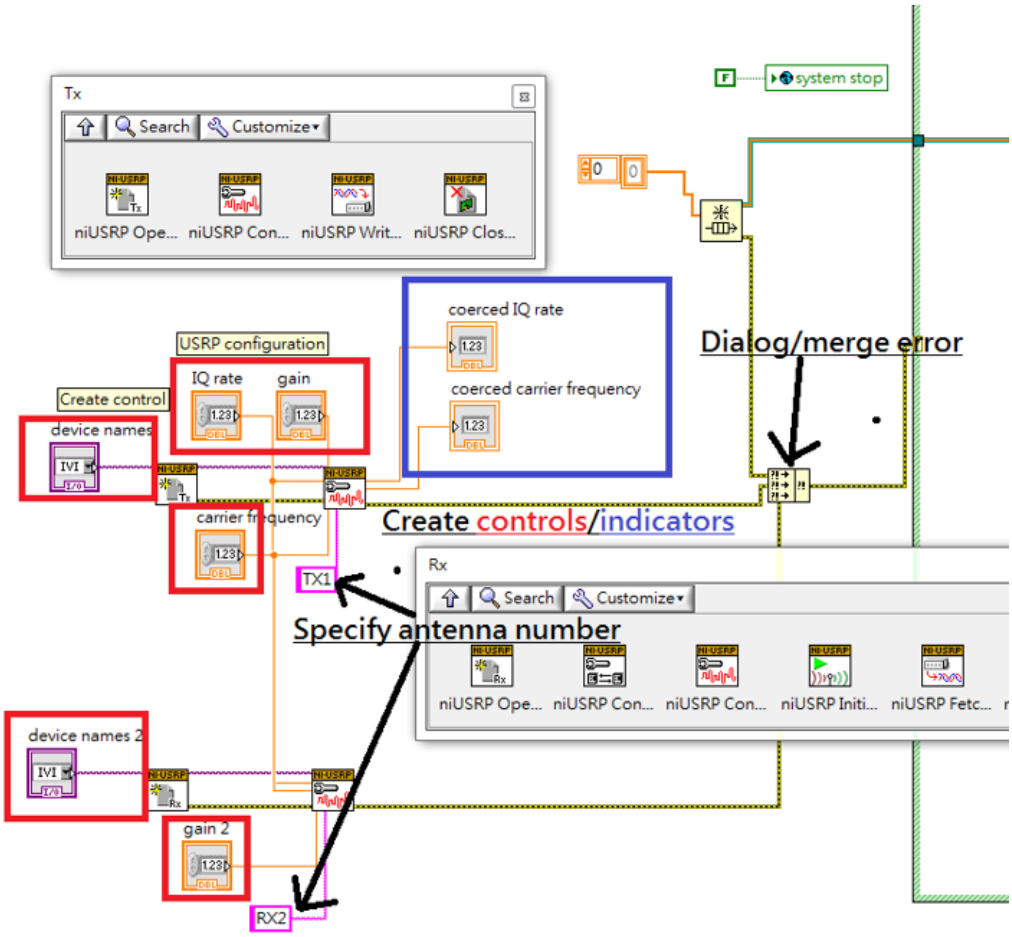


Figure 15: Step 2 of USRP reprogram example

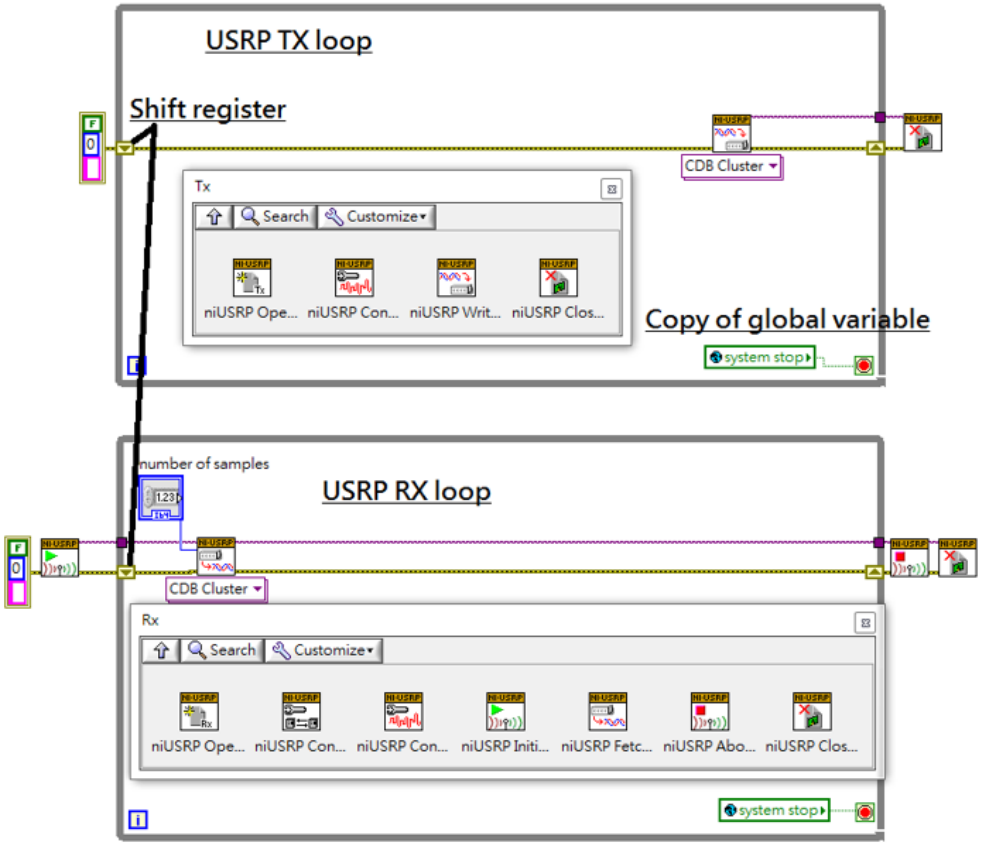
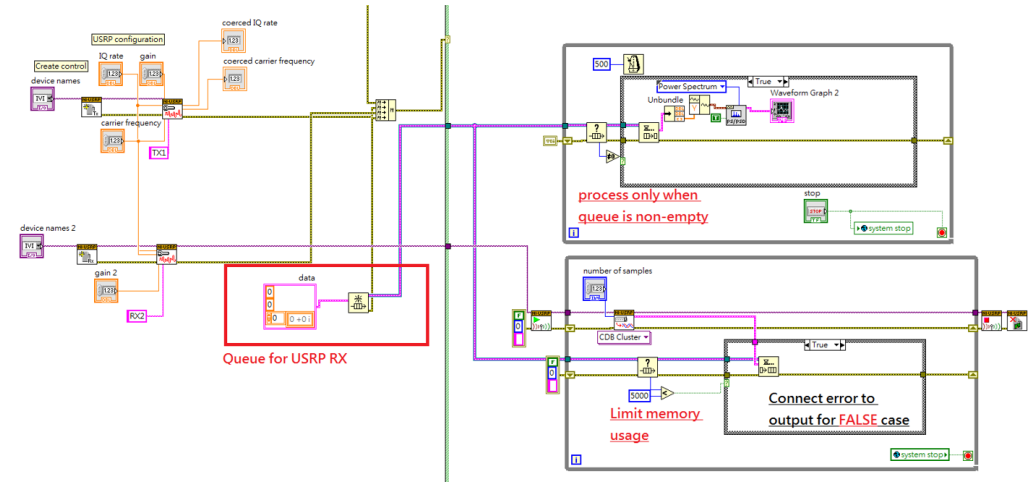
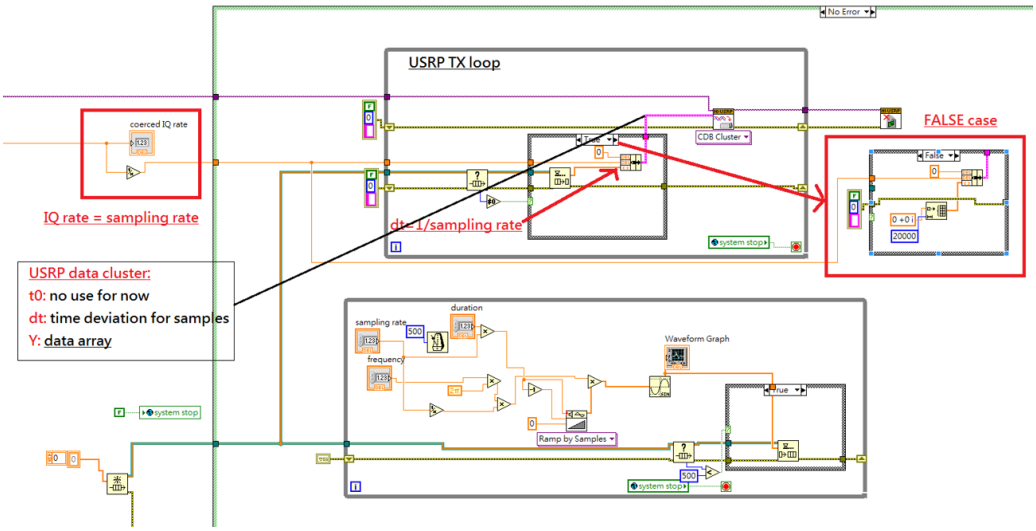


Figure 16: Step 3 of USRP reprogram example



(a) Step 4



(b) Step 5

Figure 17: Step 4-5 of USRP reprogram example

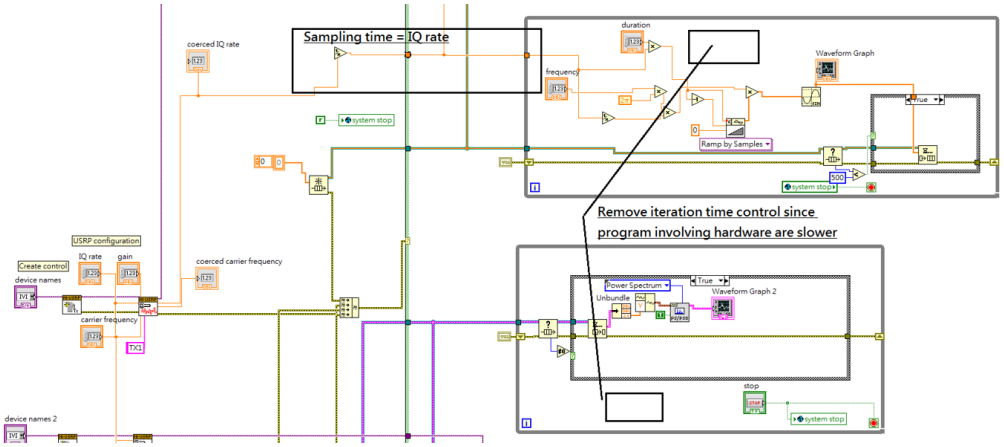


Figure 18: Step 6 of USRP reprogram example

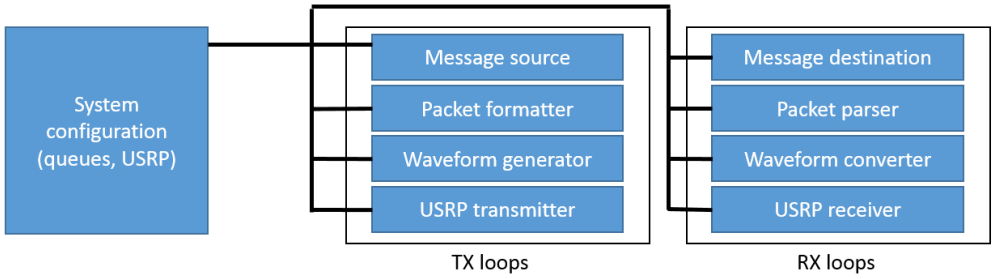


Figure 19: System structure of USRP Packet Transceiver

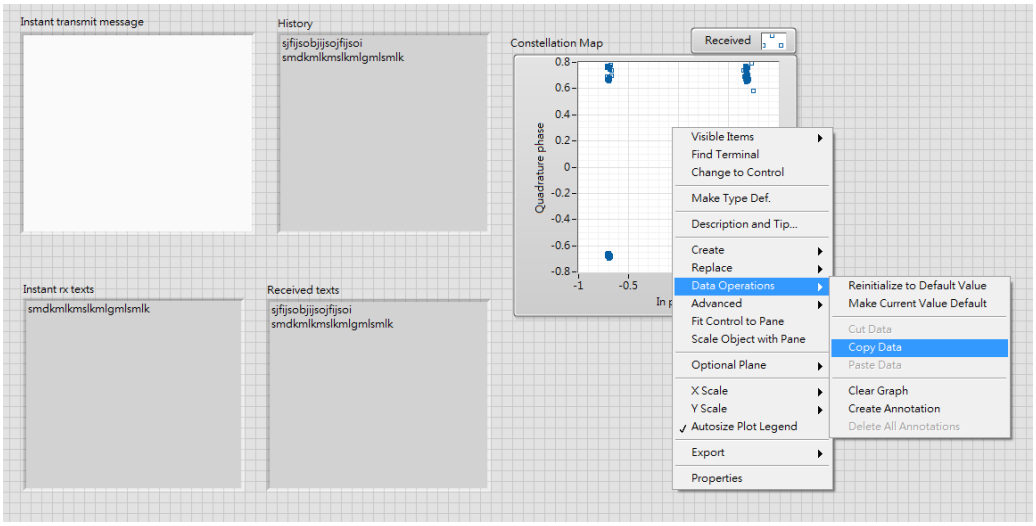


Figure 20: Copy data from LabVIEW graph