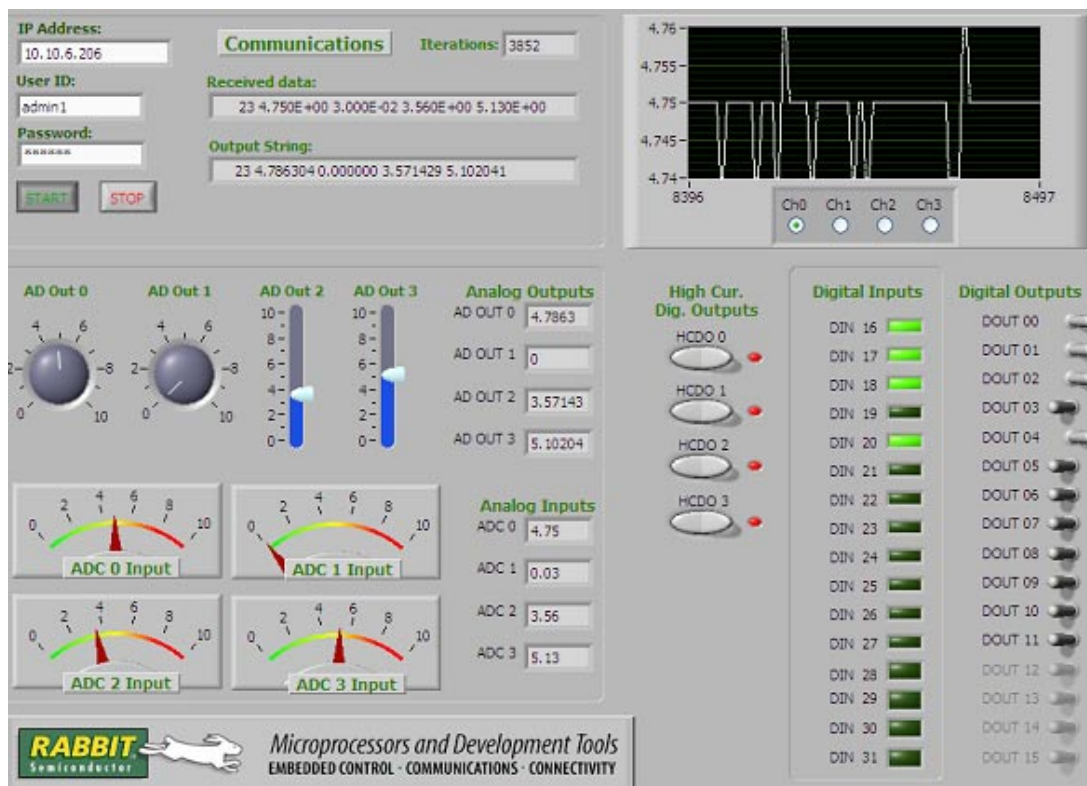


Networking a BL2600, BL2500, BL2100 or a BL2000 with a LabVIEW Virtual Instrument

The purpose of this application note is to demonstrate how to create a PC-based LabVIEW Virtual Instrument (VI) that interfaces via TCP sockets and Ethernet with one of the single board computers (SBC) listed in the title. The BL2600 is used to explain the application. For information on using the BL2500, the BL2100 or the BL2000 see the section titled, “[Using Other Boards.](#)” The figure below shows the VI developed for this document. LabVIEW is an excellent tool for quickly creating PC front ends for Supervisory Control and Data Acquisition (SCADA), Industrial Automation, and other applications where viewing and analyzing data which may be rapidly changing is required, and a web page interface just won’t do.

Figure 1. LabVIEW/BL2600 Virtual Instrument



BL2600 Features

The BL2600 is one of Rabbit Semiconductor's most feature-rich SBCs. Its features include:

- 512K each of static RAM and flash
- 44.2 MHz Rabbit 3000 MPU
- 16 protected digital inputs
- 16 I/O configurable as either inputs or sinking outputs
- 4 high-current digital outputs configurable as sinking or sourcing.
- 8 11-bit AD channels (4 configurable as current inputs)
- 4 12-bit D/A channels configurable as 0 to +10 V or -10 to +10 V
- RabbitNet expansion ports for additional I/O including digital, analog, relays and keypad/LCD
- 10/100 Base-T Ethernet
- Up to 5 serial ports

See www.rabbit.com/products/bl2600/ for more details.

Here are just a few of the things you can do with the BL2600 alone that you cannot do with LabVIEW and a PC:

- Interact with the SBC without a PC via web browser, serial terminal, or other means.
- Run autonomous control programs without a PC
- Define and send e-mail alarms (requires extra software from LabVIEW)
- Configure or reprogram the BL2600 with a browser (you can even create a secure web server on the BL2600 using SSL).
- Implement PID loops or Fuzzy Logic (requires extra software from LabVIEW)

The BL2600 is fully programmable with Dynamic C. It can be programmed to run with full autonomy to control processes or partial autonomy and an external interface to a keypad, a web browser or other human interface. The versatility of Dynamic C and LabVIEW make it easy to create applications with different degrees of autonomy. A BL2600 control application that runs completely autonomously but allows real-time viewing of data in a VI is another possibility.

For this application, we will use a simple engine that exchanges data continuously with the VI and allows the VI to fully control most of the digital and analog outputs and monitor the inputs.

About LabVIEW

National Instruments (www.ni.com) is a well-known company specializing in hardware and software for data acquisition and SCADA applications. LabVIEW is their flagship software. It is a graphical programming tool for creating PC interfaces with a wide variety of capabilities. The basic version used for this application, LabView Express 7.1, includes all the features needed to create the VI discussed here and shown in [Figure 1](#).

LabVIEW has an intuitive and easy-to-use user interface. It also has excellent help facilities. It is suggested that people new to LabVIEW use the tutorial section to learn their way around; Windows users with C programming or graphical programming experience should have a short learning curve, and many will not need to finish the tutorial before creating their own VIs.

LabVIEW users new to Rabbit products will appreciate the low cost of Rabbit-based SBCs and Dynamic C and the versatility they provide for creating intelligent sensors and control applications.

The focus in this Application Note is on setting up the TCP communication engine and a few of the trickier elements of the sample application.

Running the Sample Application

The sample application can be run without connecting any devices to the BL2600. You can loop back I/O on the SBC to see the VI in action. You can also connect to external devices if you prefer, or use a voltmeter to view outputs. Read the *BL2600 User's Manual* before using the board's I/O lines.

BL2600

The sample program to run the SBC is `BL2600_VI_Engine.C`. The program can be found in the ZIP file accompanying this document, `AN407.zip`.

Connecting I/O

Before you power up your SBC, make any or all of the connections in [Table 1](#) to set up some looped back I/O.

Table 1. Loop-back I/O Connections

Digital I/O	High Current Digital I/O	Analog I/O
J3:DIO00 — J3:DN16	J13:HOUT0 — J3:DN28	J11:AN0 — J11:AV0
J3:DIO01 — J3:DN17	J13:HOUT1 — J3:DN29	J11:AN1 — J11:AV1
J3:DIO02 — J3:DN18	J13:HOUT2 — J3:DN30	J11:AN2 — J11:AV2
J3:DIO03 — J3:DN19	J13:HOUT3 — J3:DN31	J11:AN3 — J11:AV3
J3:DIO04 — J3:DN20		<i>(Analog Current I/O unused in this example)</i>
J3:DIO05 — J3:DN21		
J3:DIO06 — J3:DN22		
J3:DIO07 — J3:DN23		
J3:DIO08 — J3:DN24		
J3:DIO09 — J3:DN25		
J3:DIO10 — J3:DN26		
J3:DIO11 — J3:DN27		

Connecting to the Network

You must have an available fixed IP address on your local network to run this application. You can connect the BL2600 Ethernet port to a hub or switch, or connect it directly to your computer's Ethernet port using a cross-over cable. If you use a cross-over cable, the PC's IP address must be set to a fixed address that is different from the IP address of the BL2600.

Configuring the IP Address of the BL2600

The only change you need to make before compiling the BL2600 program is to set the IP address in the file `/LIB/TCP_CONFIG.LIB`. Change the macro definition of `_PRIMARY_STATIC_IP` to the appropriate IP address, e.g.,

```
#define _PRIMARY_STATIC_IP "10.10.6.206"
```

Save the changed library file before compiling `BL2600_VI_Engine.C`

Compile and Run BL2600_VI_Engine.C

Compile the program and run it as you normally would. See the "Getting Started" section of the *BL2600 User's Manual* if you haven't done this yet.

The VI

The Zip file accompanying this document contains the VI LabVIEW file. It also contains an EXE version of the VI that can run stand-alone after you have installed the parts of LabVIEW necessary for running EXE files. Executable files cannot be modified with LabVIEW. The software needed to create EXE files from VI files is not part of the basic LabVIEW package and must be purchased separately from National Instruments.

You can run either version of the VI, `BL2600_VI_01.VI` or `BL2600_VI_01.EXE`.

Running BL2600_VI_01.VI

Double click on the file, `BL2600_VI_01.VI`. You must have LabVIEW version 7.1 or later installed to open this VI file. Now, click the Run Continuously button (Figure 2) on the LabVIEW control bar.



Figure 2.

When you start the VI, it will loop waiting for you to click the Start button. Before pressing the Start button, you must enter a password and user ID. "BL2600" is the default for both.

Running BL2600_VI_01.EXE

To run the stand-alone executable you need to run the LabVIEW installation in the `/install` folder (unless you already have LabVIEW or a LabVIEW client engine installed). The installation program is `SETUP.EXE`. Once you have run the installation, you can run `BL2600_VI_01.EXE` by double clicking on it; then, click the Run Continuously button (Figure 2) so that it appears black.

As described above, when you start the VI, it will loop waiting until you click the Start button. Once the password is accepted, you can adjust the analog controls and flip toggle switches and watch what happens. The high current digital output rocker switches are set to only assert the outputs until the switch is released. This operation is easily changed in LabVIEW with the properties dialog for those elements.

How it Works

This section describes details of both the LabVIEW VI and the program running on the BL2600.

The BL2600

The following pseudocode describes the operation of the BL2600:

```
INITIALIZE BOARD, I/O, AND TCP/IP
LOOP FOREVER
[
  WHILE( TCP/IP INITIALIZATION NOT COMPLETE )
  [
    CALL TCP_TICK() TO SERVICE TCP/IP
  ]
  CALL TCP_LISTEN() TO INITIATE LISTENING FOR VI OPENING SOCKET
  WHILE( SOCKET NOT ESTABLISHED )
  [
    TCP_TICK()
  ]
  WHILE( NO BYTES RECEIVED FROM SOCKET )
  [
    TCP_TICK()
  ]
  VERIFY PASSWORD
  IF ( PASSWORD VERIFIED )
    SEND "ACKPW" MESSAGE
  ELSE
    SEND "NOACK" MESSAGE
  IF ( PASSWORD VERIFIED )
  [
    DO
    [
      READ VI PANEL INPUT DATA STRING FROM TCP/IP SOCKET
      PARSE STRING FOR OUTPUT VALUES AND SET BL2600 OUTPUTS
      READ BL2600 INPUTS AND FORM STRING MESSAGE FOR VI
      SEND BL2600 INPUT STRING TO VI
    ] WHILE ( TCP_TICK(SOCKET) DOESN'T RETURN 0 )
  ]
]
]
```

VI Overview

The LabVIEW program is laid out as a “flat sequence” structure. The sequences are labeled as Frames 1-7. The high level operation of the code is described by the following pseudocode:

```
FRAME 1:
    BADPW = FALSE
    INITIALIZE OTHER VARIABLES

FRAME 2:
    WHILE( START BUTTON NOT HIT ) DO NOTHING

FRAME 3:
    LOAD IP ADDRESS AND PASSWORD FROM PANEL
    OPEN CONNECTION

FRAME 4:
    SEND PASSWORD

FRAME 5:
    RECEIVE “ACKPW” OR “NOACK” MESSAGE FROM BL2600

FRAME 6:
    WHILE( BADPW == FALSE AND STOP BUTTON NOT HIT )
    [
        IF( PASSWORD ACK'ED ) // (CASE STRUCTURE)
        [
            /*** THESE 3 LINES REPRESENT THE NESTED “STACKED SEQUENCE”
            GET VI PANEL INPUT DATA TO SEND TO BL2600
            SEND PANEL INPUT DATA
            RECEIVE BL2600 INPUT DATA, UPDATE VI PANEL DISPLAY
        ]
        ELSE
        [
            PRINT BAD PASSWORD MESSAGE
            BADPW GOOD = TRUE
        ]
    ]

FRAME 7:
    CLOSE TCP/IP SOCKET
    GOTO FRAME 1:
```

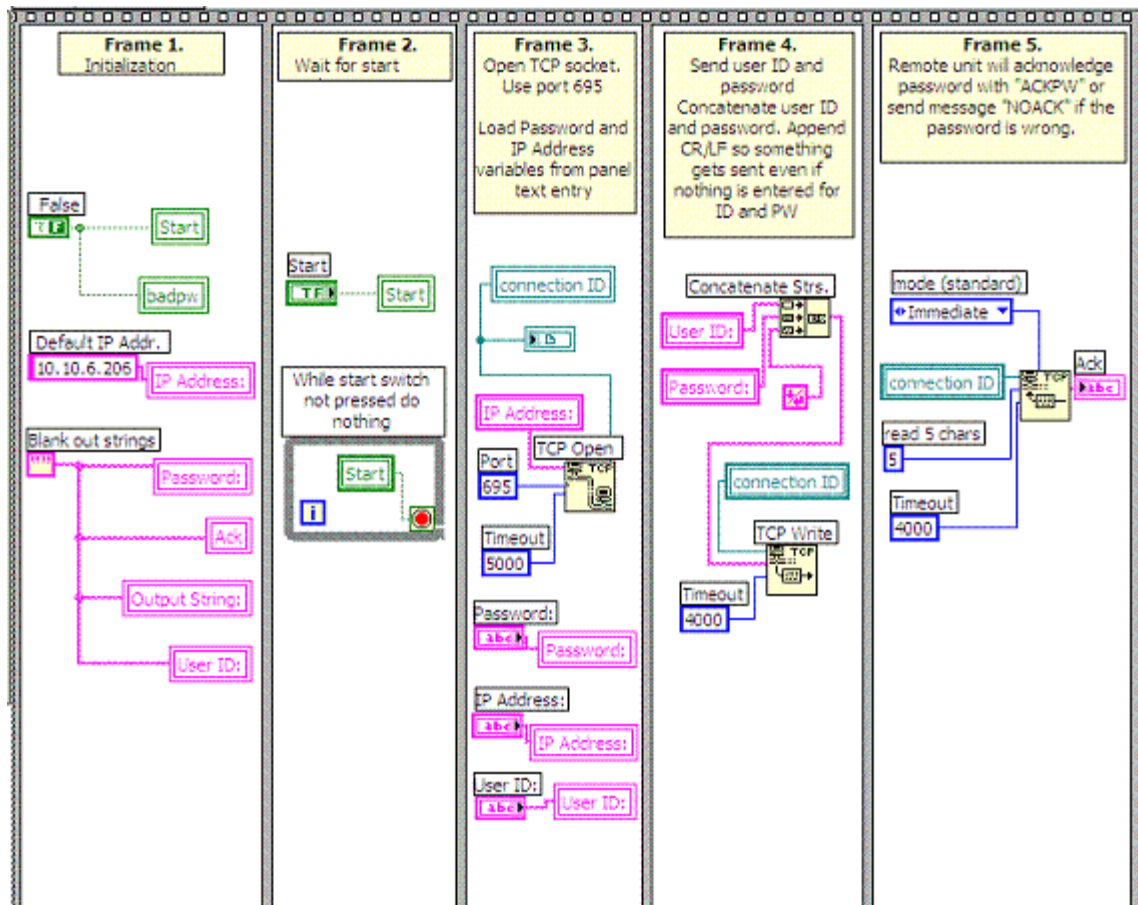
If you have LabVIEW you can use it to look at the block diagram for BL2600_VI_01.VI.

Initialization

The first five frames shown in Figure 3 perform fairly self-explanatory initialization and password verification. The labeling feature of LabVIEW was used to describe what happens. The user ID and password are concatenated and appended with a CR/LF in Frame 4 so the VI will not try to send a NULL string. The password is sent in unencoded text. Plenty of arithmetic functionality is available in both LabVIEW and Dynamic C for those who want to do some encoding on the password.

The default error handling of TCP/IP in LabVIEW was adequate for running in the LabVIEW environment. For the stand-alone executable, some additional error handling was added per LabVIEW samples. The additional error handling is not shown here for the sake of brevity, but is present and clearly documented in the VI source files. The default time-outs on the TCP/IP functions, expressed in millisecond units, are rather long, and are shortened to just a few seconds.

Figure 3. Block Diagrams of Initialization Frames for the VI

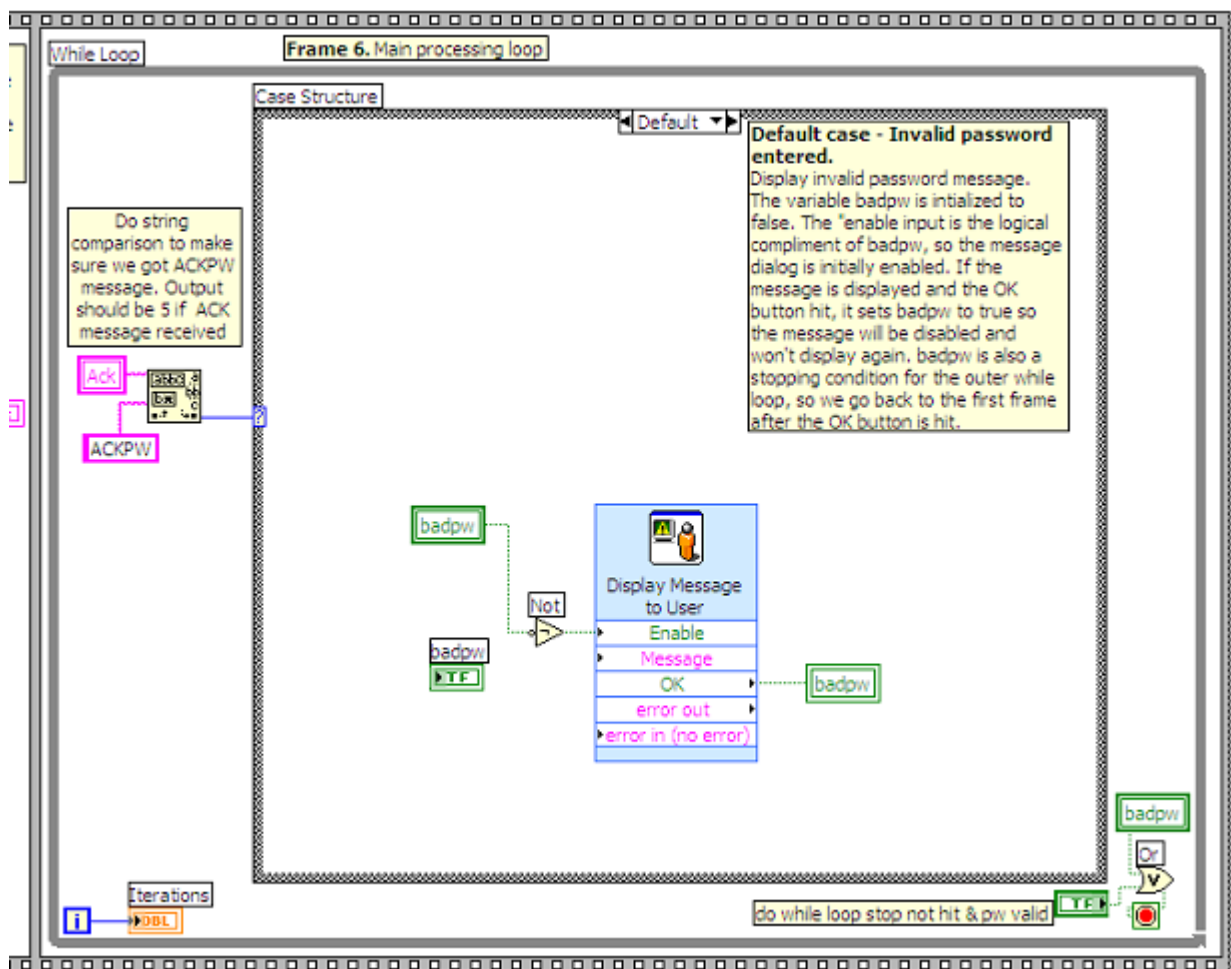


Main Processing Loop

Figure 4 shows Frame 6, the main processing sequence. The frame’s processing is a while loop that executes as long as the Stop button is not hit, and the boolean variable “badpw” is false. The construct in the lower right corner is the while loop’s stopping condition. The construct on the left side in the middle uses a string comparison to test for a properly ACK’ed password. The large case structure nested in the while loop is the functional equivalent of an if/else statement, for which LabVIEW does not have a separate structure.

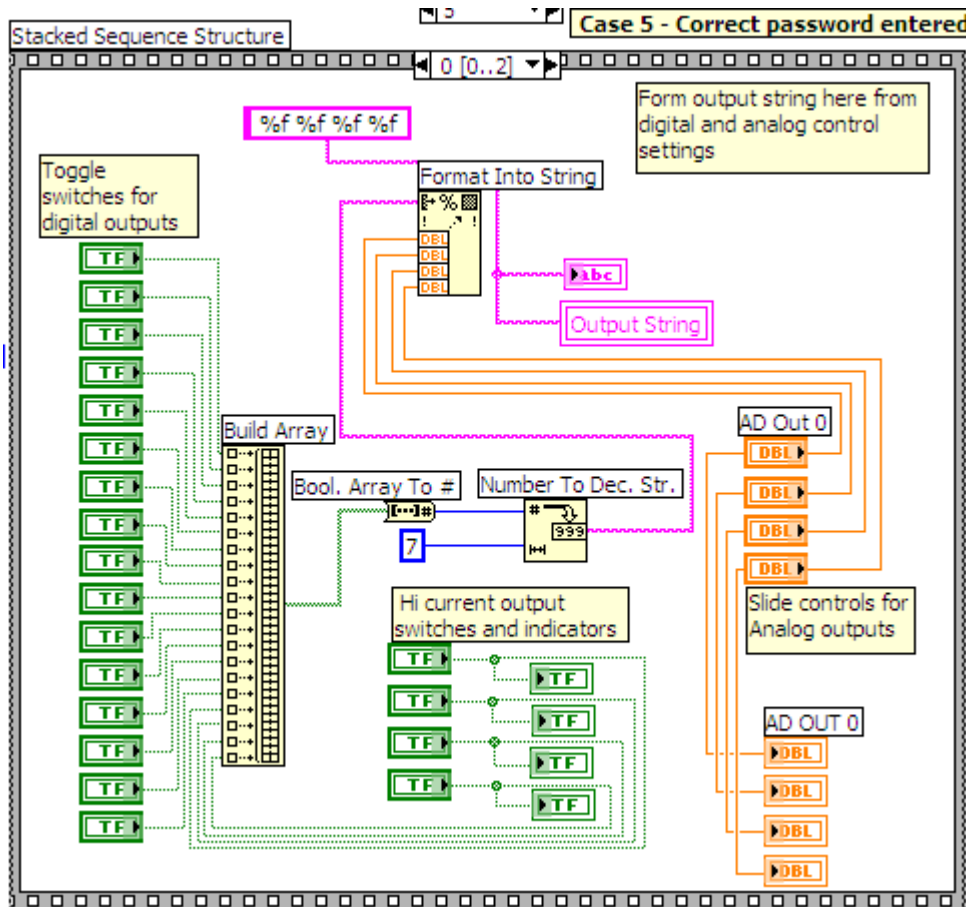
The default case shown in Figure 4 is a non-ACK’ed password, which causes a dialog message “Invalid Password” to display on the VI, and the while loop to exit. An iteration variable is created automatically at the lower left when a while loop is created, and here we hook the variable to a VI display indicator to show the number of iterations.

Figure 4. VI Main Processing Loop Showing the Default Case of the Nested Case Structure



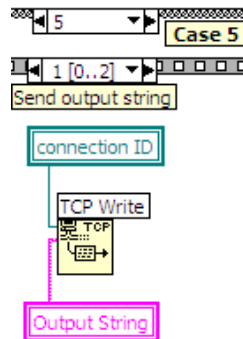
The case for an ACK'ed password (output of the string match function equals 5) contains a nested “stacked sequence” structure. The only difference between a flat sequence and a stacked one is the way it is shown in LabVIEW. One form can be changed to the other with a right mouse menu button command. (Changing a flat sequence to a stacked one will resize all the sequence frames to the size of the largest frame.) The first frame in the sequence shown in [Figure 5](#) reads the VI inputs and forms them into a string. The format string with “%f %f %f %f” will look familiar to C programmers. The “Format into string” function takes an initial string as an input. The initial string here is formed by converting the boolean output of the toggle switches to a boolean array to a long integer to a decimal string. The “%f” format specifiers are then applied to the slide and knob controls for ADC output on the VI panel, and the resulting string is output to a string variable. The ADC output values are also fed to text outputs on the VI panel.

Figure 5. First Sequence of Stacked Sequence for Main Processing



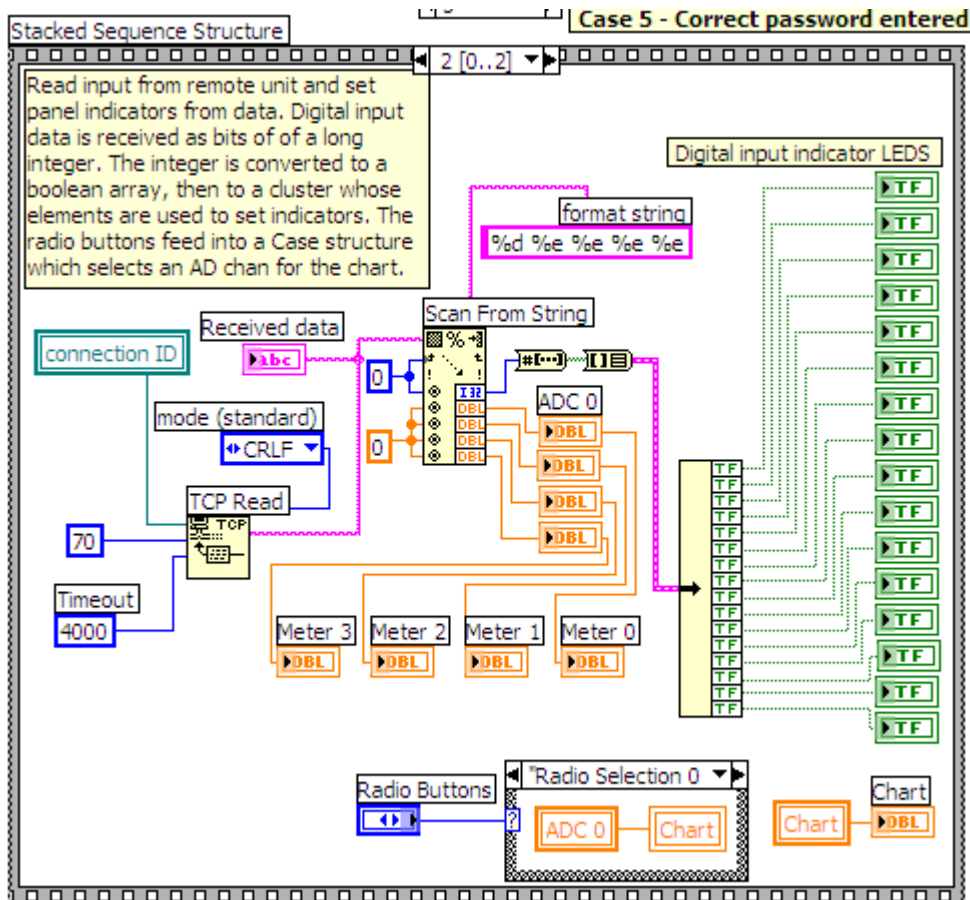
The second sequence in the stack (shown in Figure 6) simply sends the output string over the TCP/IP socket to the BL2600.

Figure 6. Second Sequence of Stacked Sequences for Main Processing



The third sequence in the stack (Figure 7) does approximately the reverse of the second frame, receiving the data sent by the BL2600 and extracting from it values for the panel display. The time versus voltage chart on the VI panel has channel selector radio buttons on the front of it. This is actually a separate control placed on top of the plot graphic. The radio buttons control integer output feeds into a case structure that selects the ADC input variable to feed into the chart.

Figure 7. Third Sequence of Stacked Sequence for Main Processing



Using Other Boards

You can switch out the BL2600 using the BL2000, BL2100 or the BL2500. The only difference in set up will be the loop back I/O connections, which are shown below. You will also need software specific to the target board. The main screen for the BL2600, shown in [Figure 1](#), will be different when using one of the other boards; the other screens shown throughout this document vary slightly from board to board. The password and user ID for the BL2000, BL2100 and the BL2500 will default to “Rabbit” for both.

Using the BL2000

The software in `AN407_20.zip` and `AN407_20b.zip` (both available with this application note) is for use with the BL2000. The former includes LabVIEW support and the latter does not.

Table 2. BL2000 Loop-back I/O Connections

Digital I/O	Analog I/O
J9:OUT0 – J2:IN0	J4:DAC0 – J4:ADC0
J9:OUT1 – J2:IN1	J4:DAC1 – J4:ADC1
J9:OUT2 – J2:IN2	
J9:OUT3 – J2:IN3	
J9:OUT4 – J2:IN4	
J9:OUT5 – J2:IN5	
J9:OUT6 – J2:IN6	
J9:OUT7 – J2:IN7	

Using the BL2100

The software in `AN407_21.zip` and `AN407_21b.zip` (both available with this document) is for use with the BL2100. The former includes LabVIEW support and the latter does not.

Table 3. BL2100 Loop-back I/O Connections

Digital I/O	Analog I/O
J13:OUT10 – J11:IN10	J1:DA0 – J1:AD0
J13:OUT11 – J11:IN11	J1:DA1 – J1:AD1
J13:OUT12 – J11:IN12	J1:DA2 – J1:AD2
J13:OUT13 – J11:IN13	J1:DA3 – J1:AD3
J13:OUT14 – J11:IN14	
J13:OUT15 – J11:IN15	
J7:OUT09 – J4:IN09	
J7:OUT08 – J4:IN08	
J7:OUT07 – J4:IN07	
J7:OUT06 – J4:IN06	
J7:OUT05 – J4:IN05	
J7:OUT04 – J4:IN04	

Table 3. BL2100 Loop-back I/O Connections

Digital I/O	Analog I/O
J7:OUT03 – J4:IN03	
J7:OUT02 – J4:IN02	
J7:OUT01 – J4:IN01	
J7:OUT00 – J4:IN00	

Using the BL2500

The software in AN407_25.zip and AN407_25b.zip (both available with this application note) is for use with the BL2500. The former includes LabVIEW support and the latter does not.

Table 4. BL2500 Loop-back I/O Connections

Digital I/O	Analog I/O
J3:OUT0 – J11:IN0	J1:AD0 – external power supply
J3:OUT1 – J11:IN1	
J3:OUT2 – J11:IN2	
J3:OUT3 – J11:IN3	
J3:OUT4 – J11:IN4	
J3:OUT5 – J11:IN5	
J3:OUT6 – J11:IN6	
J3:OUT7 – J11:IN7	

More Information

Application Note 408 discusses a more sophisticated data acquisition system that includes a VI interface, and a web browser interface for configuring user access, e-mail alarms and I/O, and for viewing data. You can read more at the Rabbit Semiconductor website.

- Rabbit Semiconductor: www.rabbit.com
- National Instruments: www.ni.com