Data logging goes wireless

By Simon Bramble
North European
Applications Engineer
austriamicrosystems
E-mail: simon.bramble
@austriamicrosystems.com

Over the last five years, Bluetooth's growth has been undeniably phenomenal. However, most consumers still think of Bluetooth as a wireless communication medium for cellphones to talk to headsets. Although this is still largely true, Bluetooth is finding its way into more routine forms of communication such as industrial data logging.

This article discusses the design of a Bluetooth data logger that takes data from an austriamicrosystems AS1530 12bit ADC via an MCU and a Bluetooth link to a PC, which collects the data in a spreadsheet. The trend in industrial process control has been to reduce the communications wiring across a plant with parallel data, paving the way for serial "Profibus" data. This design extrapolates that trend and completely removes the communications wiring.

The data logger circuit has an AS1530 ADC connected to a PIC MCU that communicates via an RS-232 cable to a BlueGiga WT12 Bluetooth module. This module transmits data over the Bluetooth link to a second identical BlueGiga WT12 module connected via an RS-232 cable to a PC running a Windows program written in Visual Basic Version 6. The Windows program loads the received data into an Excel spreadsheet for results to be analyzed and graphed.

The data logger is powered from a 5V bench power supply. To ensure safe operation of the MCU, the logger power supply is monitored using a very low-power supervisor, the AS1904. This device typically consumes 150nA of supply current. The current consumption of the data logger circuit is measured at 16mA, and the average current consumption of the BlueGiga module is specified as 44.7mA. So if the whole circuit has to be powered from a linear regulator, an AS13985 150mA low dropout linear regulator would more than suffice.

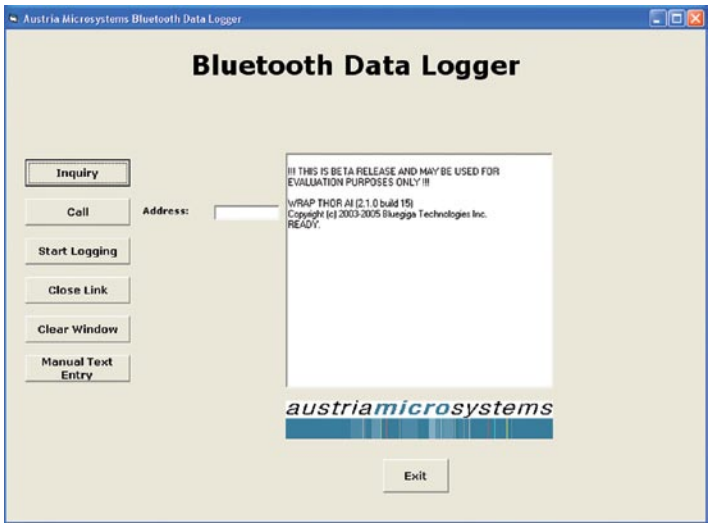

Figure 1: The program replicates Microsoft HyperTerminal and handles the search, connection and disconnection of the link with the data logger.

Analog front-end

The analog front-end consists of an AS1230, 12bit successive approximation register (SAR) ADC. If the input signal is less than the supply voltage of the ADC (in this case, 5V) and of suitably low source impedance, it can be fed directly into the ADC's input.

The ADC's input circuit can be modeled as a simple RC circuit,

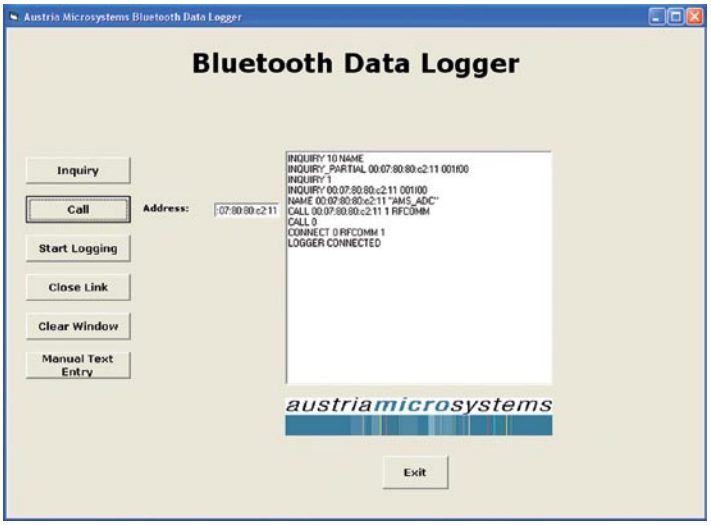

Figure 2: After reset, the user clicks the Inquiry button, which tells the Bluetooth module to send out an inquiry looking for all local Bluetooth devices.

with R representing the source impedance of the signal and C the sampling capacitor of the ADC. An RC circuit under charge is represented by the equation:

$$V = V_{max}\left(1 - e^{\frac{-t}{CR}}\right)$$

where V_{max} is the charging voltage and V is the voltage across the capacitor. The sampling capacitor is 18pF and the sampling time (t) is 390ns. Now, the worst-case change in voltage that the ADC input has to deal with is when one input channel is set at 0V, and the adjacent channel is set at 2.5V. In this case, the input capacitor has to charge to 2.5V within an accuracy of half LSB in 390ns.

So

$$2.5V - \frac{1}{2}LSB = 2.5V - \frac{2.5V}{(2x4096)}$$

Hence,

$$2.5V - \frac{2.5V}{(2 \times 4,096)} = 2.5 \times (1 - e^{\frac{-t}{CR}})$$

so

$$1 - \frac{1}{8192} = 1 - e^{\frac{-t}{CR}}$$

$$\frac{1}{8192} = e^{\frac{-t}{CR}}$$

$$R = \frac{-t}{C \times \ln(\frac{1}{8192})}$$

This yields a minimum source impedance of R = 2.4kΩ.

Now, the input impedance of the mux is 800Ω, so the source impedance of the signal has to be less than 1.6kΩ.

If the source impedance of the signal to be digitized is too high, it can be amplified and/or buffered using an op amp.

The PIC16F627 MCU reads data into and out of the AS1530 using an SPI with a chip select (CSN), serial clock (SCLK), data input (D_{in}) and data output (D_{out}). The CSN line frames the data, while data is clocked into and out of the device using the rising edge of the SCLK line. The data stream is made up of eight control bits that select the input channel, the input range and the power mode, and 16 subsequent bits that contain the output data. Once the conversion is completed, the ADC goes into its programmed power mode (full power down, reduced power or normal operation).

To achieve the desired accuracy, pay close attention to the board layout and decoupling of the supply pins to the chip and the reference. The analog and digital supplies must be kept separate, although both supplies can originate from the same source. This is best achieved by

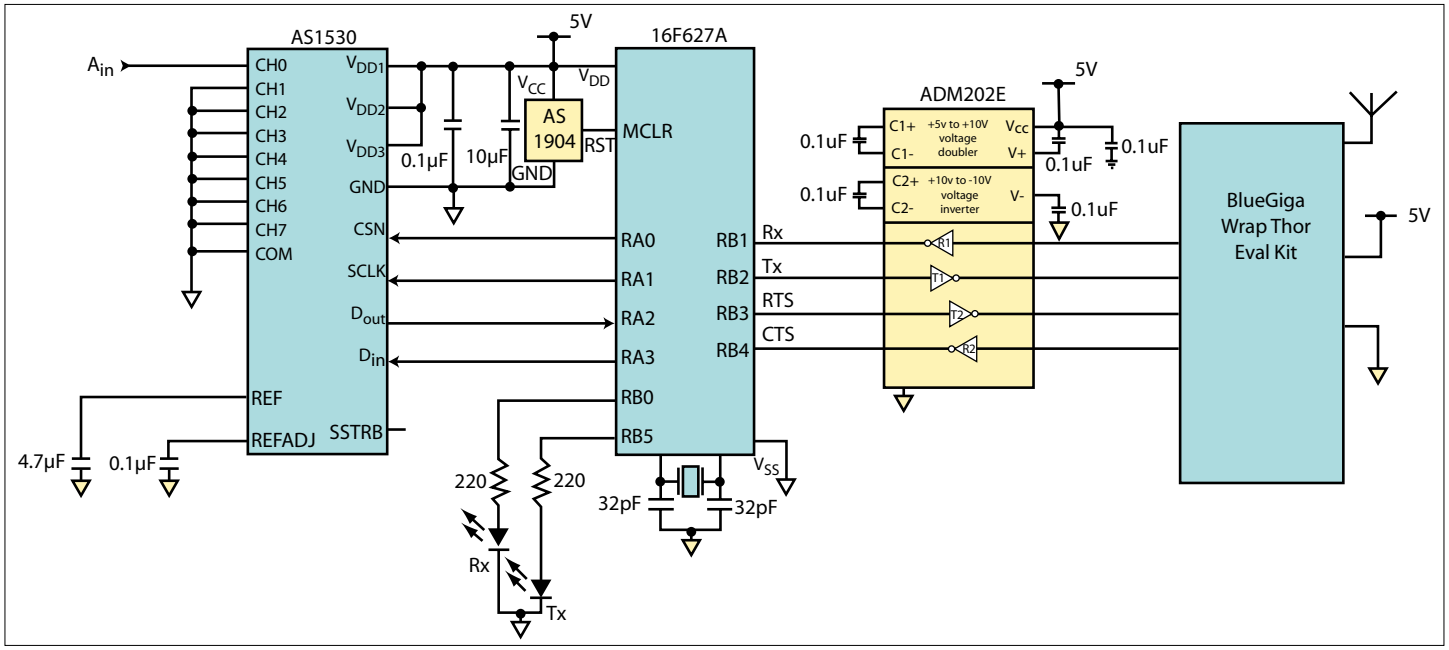

Figure 3: The data logger circuit has an AS1530 ADC connected to a PIC MCU that communicates via an RS-232 cable to a BlueGiga WT12 Bluetooth module.

routing two tracks from the main board supply, a thick track to the analog supply (pins V_{DD1} and V_{DD3}) and the same for the digital (pin V_{DD2}). The current taken by the digital supply modulates a voltage onto the digital supply line. If this also powers the analog supply to the chip, this voltage modulation could corrupt the LSBs of the ADC. Providing separate supplies eliminates this problem.

The ground supply to the AS1530 is just as important as the analog and digital supplies. A thick track running from the GND pin to the main board supply ground provides low impedance for the return current and is essential for best performance.

Decoupling capacitors should be placed close to each of the analog and digital supply pins of the chip, with their ground connections close to the GND pin of the AS1530. This ensures that the AC voltage seen across the power supply pins of the chip remains at zero. A 10µF tantalum capacitor in parallel with a 100nF ceramic provides optimum broadband attenuation of any noise, which may appear on the line and provide a low impedance path for any current surges that may be demanded by the AS1530.

This design uses the internal reference of the AS1530. If an external reference is desired, it needs to be decoupled with a local low-impedance capacitor of 4.7µF close to the REF pin. The input impedance to the reference pin of a SAR ADC changes with digital code due to it being connected directly to the R-2R ladder.

Software end

At the data logger end, the MCU is a PIC16F627A. Its code is written in C using a Hi-Tech C compiler. The MCU's internal UART is set to a baud rate of 115.2Kbps to match the default baud rate of the Bluetooth module. Data is sent over the RS-232 link via an industry standard (2Tx/2Rx) level translator.

All communications to the Bluetooth modules are done over an RS-232 interface. As with many things in life, the simple aspects can cause the most hardship, and RS-232 is no exception. In the days when the RS-232 specification was written, computers were called terminals and they sometimes linked to communication devices like modems. Hence, it was decreed that two types of equipment could be used to connect to each other using the RS-232 standard. These were the data terminal equipment (DTE) and data communication equipment (DCE). It was not surprising that computers (or terminals) were configured as DTE, and most devices that connect to them were configured as DCE (like modems). They were mostly connected using a standard nine-way RS-232 cable, with each pin at one end connected through its associated pin at the other end. No wires cross over inside the cable.

Now, DTE sends data out on pin 3 and receives it on pin 2. On the other end, DCE receives data on pin 3 and transmits on pin 2. There are two other signals used in 90 percent of the RS-232 applications: ready to send (RTS) and clear to send (CTS). Both devices at each end of the cable have an RTS and a CTS signal. The RTS at one end connects directly to the CTS at the other end and vice versa. The RTS is an outgoing signal, while the CTS is an incoming signal.

Before any data is sent, both ends set their RTS lines (as they are both free to send and receive data). If the transmitter asserts RTS, this in turn asserts the CTS line at the receiver since they are directly connected. When the transmitting device starts to send data, it de-asserts its RTS line, hence de-asserting the CTS of the receiver. De-asserting the CTS tells the receiver that it is not clear to send data. This tells the receiver to hold off transmitting any information. If the transmitter sends too much information (and overloads the receiver), the receiver is free to de-assert its RTS line (de-asserting the CTS line of the transmitter) to indicate the transmitter to stop.

The Bluetooth module is configured as DCE, so the MCU at the AS1530 end is configured as DTE. Thus, it transmits its data on pin 3 and receives it on pin 2. It also uses RTS/CTS handshaking, transmits RTS on pin 7 and receives CTS on pin 8 of the nine-way connector. It should be noted that if the MCU is to be connected to HyperTerminal on the PC (for testing), the Tx and Rx lines need to be swapped over, as do the RTS and CTS lines. This converts the MCU from DTE to DCE.

A standard RS-232 level translator was used to interface the MCU with the RS-232 data line.

It should also be noted that a carriage return is needed at the end of every command sent to the Bluetooth module. This simple instruction is overlooked when typing on a keypad. If it is not sent, the designer will have hours of undeserved diagnostics as to why the module is not behaving itself.

The Bluetooth module has the facility to be programmed with a "friendly" name, allowing other Bluetooth devices to identify it by something other than its 12-digit hex address. When the data logger powers up, the 16F627A

programs the Bluetooth module at the data logger end to have a "friendly" name of AMS_ADC.

The Windows program was written in Visual Basic Version 6. The program replicates Microsoft HyperTerminal and handles the search, connection and disconnection of the link with the data logger. **Figure 1** shows the Windows screen.

The text screen in **Figure1** shows the message from the Bluetooth module once it has been reset. On the left are the controls needed to communicate with the Bluetooth module. After reset, the user clicks the Inquiry button, which tells the Bluetooth module to send out an inquiry looking for all local Bluetooth devices (**Figure 2**).

Clicking the Inquiry button writes the text "INQUIRY 10 NAME" to the Bluetooth module, telling it to wait for 10s before timing out and inquiring for all the "friendly" names of the local Bluetooth devices.

The Bluetooth module then echoes back how many devices are found, their 12-digit hex address and friendly names. It can be seen that the device AMS_ADC has an address 00:07:80:80:c2:11. The user highlights this address and copies it to the Address: window. By clicking the Call button, the Windows program sends out a CALL command to this address, and this is shown by the line: CALL 00:07:80:80:c2:11 1 RFCOMM.

The Bluetooth device echoes back with a CALL 0 statement, telling the user that a CALL has been initiated. Once communication with the remote logger has been established, a CONNECT statement is returned. Once a connection is established, all future text is generated by the data logger and not the Bluetooth module. The data logger then sends out the text "LOGGER CONNECTED" to show that communication has been established with the AS1530. The Windows program also opens a file called C:\AMS_ADC.xls and loads the received data into it.

The user then clicks the Start Logging button to clear the screen and switch the program to receive data and not text. All future data received is loaded into the C:\AMS_ADC.xls file.

The Windows program also allows the users to override the command buttons and enter their own text. The Clear Window button is clicked to clear the text screen. Entering text in the text screen and then clicking the Manual Text Entry button downloads all the text on the text screen to the Bluetooth module.

Clicking the Close Link button sends the command "+++" to the Bluetooth module, telling it to terminate the link. The Bluetooth module then drops the link and returns a message telling the user the link has been lost.

Figure 3 shows the circuit diagram of the data logger end. The PC end is simply a BlueGiga WT12 evaluation kit connected to the PC via an RS 232 cable.

The software has been written in a way that enables easy expansion of the system to allow the user to interrogate multiple data loggers. The software in the data logger can be improved to enable it to better handle commands from the host PC. Thus, it enables the PC to set the friendly name, and the user to poll and/or reset the system.