ICS ELECTRONICS
division of Systems West Inc.

IEEE 488 APPLICATION BULLETIN

GENERATING SERIAL WAVEFORMS AND TRANSFERRING DIGITAL DATA WITH ICS's DIGITAL INTERFACES

INTRODUCTION

This Application Bulletin describes how to generate serial waveforms from the digital outputs in ICS's GPIB and Serial Interfaces and techniques to increase the serial data rate. The same techniques can be used to transfer multiple bytes of data to a digital device.

This Application Bulletin has been revised to include examples using the 4803's new BINary format command that was added to the 4803 when the firmware was upgraded to Revision 5.

BACKGROUND

Some test and measurement applications require that the test engineer generate a serial waveform to send data or commands to the device under test. If the serial data format uses asynchronous ASCII characters, then the test engineer has a wide choice of serial data sources. He can use the PC's serial COM ports or one of several GPIB-to-Serial interface products that are on the market. All of these sources are programmable and can generate a variety of asynchronous serial formats and data rates.

Some test applications require non-asynchronous data formats. In these applications, the test engineer needs to generate binary data, a data clock and often a data gate or strobe signal to go with the data. Data rate accuracy is normally not the major consideration as the clock signal is supplied with the data. These kinds of waveforms can be generated by successively outputting digital words to the same data port. By properly coding the digital words, different bits can be used create the serial data, the serial clock and any other needed waveforms.

GENERATING SERIAL WAVEFORMS

Figure 1 shows a typical serial waveform with data, a clock signal and a data gate. The data signal waveform outputs four bits, 1 0 1 0. The clock waveform is pulsed when the data is stable. The data gate encloses the active data waveform. The bit pattern along the bottom of Figure 1 represents the digital data that has to be outputted to generate the waveforms.

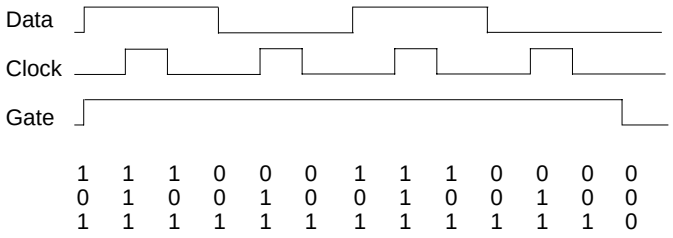

Figure 1 Basic Serial Waveforms

The output sequence to generate the waveform sets the data and then pulses the clock for each data bit. The sequence is repeated for each serial bit. If a data gate or strobe pulse is required, it is coded into the output data as shown in Figure 1.

This technique can be expanded to include additional waveforms or for full byte-wide data words.

IMPROVED SERIAL WAVEFORM CODING

The output sequence used in Figure 1 requires three digital words per bit. While workable, it is time consuming. The serial output can be speeded up by returning the clock to its inactive level at the same time that the data waveform is changed to its next value as shown in Figure 2. For most serial devices, it is only necessary to have the data stable on one edge of the clock.

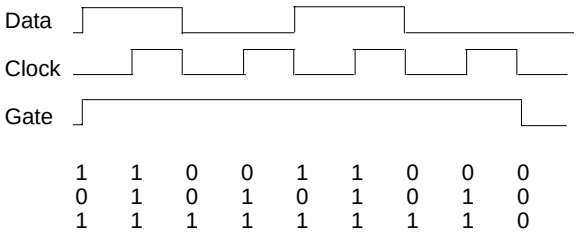

Figure 2 Improved Serial Waveforms

The technique shown in Figure 2 eliminates one output step per bit and improves the serial data rate by 33% over the basic serial waveform in Figure 1.

SPEEDING UP SERIAL DATA WAVEFORMS

In general, serial data waveform speed depends upon the computer program, the number of characters sent to the interface to generate the waveforms and how the interface is used. Check your program against the following guidelines:

1. Minimize the number of output steps used to create the serial waveform. See Figures 1, 2 and 8.
2. Use the shortest commands possible. In ICS products, use the short form commands instead of the longer SCPI commands to output the data. This minimizes GPIB bus transfer time and the interface's parsing time.
3. Use compiled instead of interpretive programs.
4. Use precanned messages where possible.
5. Replace DO-LOOP and FOR-NEXT structures with linear coding. Listing each output command separately is more work but it will execute faster in slow computers.

Steps 3, 4 and 5 are very important if you are using an older or slow computer.

GENERATING SERIAL DATA WITH A 4803 OR A 4863 GPIB TO DIGITAL INTERFACE

ICS's Model 4803 and 4863 are GPIB-to-Digital Interfaces whose outputs can be controlled by the user. There are two ways to send the 4803 and 4863 digital data. One way is to send the data with a command. The 4803 or 4863 parses the command and then outputs the data parameter to the selected port. The second and faster way is to configure a port as an output port and then transparently send it data in the dual or secondary address mode.

SENDING DATA AS A COMMAND PARAMETER

The simplest data transfer method is using the byte output type commands to send digital data to a selected output byte. These commands include the port number and a data value in each command. Value is normally decimal but can be HEX coded with #h prefix. Figure 3 shows the byte output commands used to create the waveforms shown in Figure 2.

```
BO1 5 <lf>      'sends value 05 to port#1
BO1 7 <lf>      'sends value 07 to port#1
BO1 1 <lf>      'sends value 01 to port#1
BO1 3 <lf>      'sends value 03 to port#1
BO1 5 <lf>
BO1 7 <lf>
BO1 1 <lf>
BO1 3 <lf>
BO1 0 <lf>      'sends value 00 to port#1
```

Figure 3 Byte Out Commands

The 4803/4863 parses each command to output the data. This takes 12 milliseconds per data byte for early boards with a 6 MHz clock (12.88 MHz crystal). Newer boards with 20 MHz clocks take 3.5 ms.

For each command, the GPIB bus sends an unlisten command, addresses the 4803 or 4863 as a listener and sends it the data bytes. This typically requires three GPIB address bytes plus 6 data bytes for a total of 9 GPIB bus transactions per command. Concatenating the commands into one command line eliminates the 3 GPIB address bytes for each step. Concatenated commands are separated by semicolons. This reduces the above commands to:

```
BO1 5;BO1 7;BO1 1;BO1 3;BO1 5;BO1 7;BO1 1;BO1 3;BO1 0 <lf>
```

Figure 4 Concatenated Command String

Command length is limited to the size of the GPIB buffer. Normal input buffer size is 1024 bytes. Check your manual as some early units had 256 byte input buffers.

USING CONFIGURED OUTPUT BYTES

A faster way to output multiple digital bytes is configure some ports (bytes) in the 4803 or 4863's parallel interface as output byte(s) and then send the data to the output byte(s) with a command or as a transparent string. The number of output ports depends upon the required number of data lines. This method can be easily scaled up to transfer multi-byte words to a digital device. The following example configures byte 1 as an output and uses the PO command to send the same data as in Figure 4. Again, the data command length is limited by the size of the GPIB buffer.

Setup:

```
CONF:INP (@ 1:5)      'sets all bytes to inputs
CONF:OUT (@ 1) <lf>   'sets an output byte
```

```
CONF:OUT:POL 1 <lf>     'sets output polarity
CONF:STB 0 <lf>         'sets data strobe polarity
FORM:LIST HEX <lf>      'sets data format
```

Sending Data:

```
PO 5;PO 7;PO 1:PO 3;PO 5;PO 7;PO 1:PO 3;PO 0<lf>
```

Figure 5 Using a Configured Output Port to Generate Serial Waveforms

The configuration commands in the above example define the output port, the data polarity and the type of data that will be sent to the port. The HEX data format sends 8-bit values as two HEX digits, 00 to FF. There is only a single digit in the above example as the waveform in Figure 2 has only three lines. HEX and 4833 are the fastest 4803 and 4863 output formats. HEXList and ASCii are the slowest formats.

MULTIPLE DATA BYTES IN THE SAME COMMAND

The above example uses 5 GPIB bus transactions for each data byte. It also uses up a lot of 4803/4863 processing time as the 4803/4863 has to parse each command. When the 4803 or 4863 parses the command, it outputs data until it sees the semicolon or linefeed terminator. If multiple data bytes are sent with each command, the 4803 or 4863 places each byte in the configured output port(s). Figure 6 shows how to output the waveform in Figure 2 using multiple data bytes in one command.

Sending Data:

```
PO 3571357130<lf>
```

Figure 6 SENDING MULTIPLE DATA BYTES IN ONE COMMAND

SENDING DATA TRANSPARENTLY

The transparent data method bypasses the 4803/4863's parser and uses the 4803/4863's dual or secondary GPIB address mode. The dual address mode uses consecutive primary addresses such as 4 and 5. The lower address is used for setup commands and for transferring data with commands. The upper or higher address is only used for the transparent data. The secondary address mode uses one primary address and two secondary addresses, 00 and 01. Secondary address 00 is the command address, secondary address 01 is for transparent data. The choice of which addressing method to use is up to the user.

In either case, the output port is configured as in Figure 5 and the data is sent by itself to a different address. Figure 7 shows the additional address mode command.

Setup:

```
CONF:INP (@ 1:5)            'sets all bytes to inputs
CONF:OUT (@ 1) <lf>         'sets an output byte
CONF:OUT:POL 1 <lf>         'sets output polarity
CONF:STB 0 <lf>             'sets data strobe polarity
FORM:LIST HEX <lf>          'sets data format
SYST:COMM:GPIB:ADDR:MODE DUAL <lf>
                            'enables two primary addresses
```

Sending Data:

```
3571357130<lf>
```

Figure 7 Sending the Serial Data Transparently

The order of the address mode command is not important. It can be done at any time. Provide a short delay of 70 msec after the address command before sending the 4803 or 4863 any other commands or data.

OUTPUTTING DIGITAL DATA

Outputting data is very similar to generating serial waveforms except that special waveforms like a clock or a data gate are not required and that all bits are used for data. Data bit to data bit spacing is normally not critical. The goal is simply to transfer the data. The data must be stable and then a clock generated to latch in the data into the receiving device as shown in Figure 8.

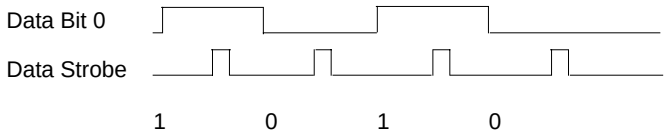

Figure 8 Data Byte Waveforms

In the serial waveform examples, the clock was coded as part of the output waveform. Digital data transfer uses the 4803 or 4863's ability to automatically generate data strobes to clock data into the receiving device instead of coding the clock as a separate bit.

Both devices generate a data strobe when they detected a command terminator or separator. The 4803 and 4863 will also generate a data strobe if a comma is placed between multiple data bytes. Figure 9 shows the Sending Data portion of Figure 6 recoded to generate a data strobe between each data byte.

Sending Data:

```
PO 3571357130<lf>       ' original line - generates a data
                          strobe only on last character
PO 3,5,7,1,3,5,7,1,3,0<lf> 'recoded - generates a data
                          strobe between each data byte
```

Figure 9 DATA STROBES FOR EACH BYTE

MULTI-BYTE WORDS

The only change needed to output multi-byte words is to configure additional ports as outputs. Because there is very little overhead in outputting the additional bytes, multi-byte words have a higher data transfer rate. Figure 10 shows how to configure the 4803 to transfer a 16-bit wide word.

Setup:

```
CONF:INP (@ 1:5)          'sets all bytes to inputs
CONF:OUT (@ 1,2) <lf>     'sets two output bytes
CONF:OUT:POL 1 <lf>       'sets an output polarity
CONF:STB 0 <lf>           'sets data strobe polarity
FORM:LIST HEX <lf>        'sets output data format
```

Sending Data:

```
PO 0001,FED3,0002,1453,0003, --- 4567<lf>
```

Figure 10 Transferring 16-bit Wide Data

The data was outputted with a command in the above example. The loop repeat time is the same for the transparent or the command transfer methods. There is no significant difference with either one. Note that spaces may be inserted between the data groups as they will be ignored by the selected format command. The HEX ist and ASCii formats that require commas between data values, need a second comma to generate a data strobe. i.e. 0,1,,254,... etc.

MAXIMIZING DATA TRANSFER RATE WITH THE 4803's BINARY FORMAT COMMAND

Data transfer rate can be greatly improved by using the 4803's new BINary output (LISTen) format. The BINary output format transfers binary bytes between the GPIB bus buffer and the digital interface and in the dual or secondary address modes. Binary data strings do not have separators. Spaces are not allowed between data bytes. Data is simply transferred to the output bytes until all bytes have data and then a strobe pulse is generated. If there is more data in the buffer, the cycle repeats until all of the data has been outputted. To maximize the data transfer rate, combine the binary data into output strings, up to 1024 characters long. Send each string to the 4803. The 4803's output burst rate is > 58 k bytes/sec for one byte-wide words to > 43 k words/sec for four byte-wide words.

Setup:

```
SYST:COMM:GPIB:ADDR:MODE DUAL
                          'enables two primary addresses
CONF:INP (@ 1:5)          'sets all bytes to inputs
CONF:OUT (@ 1,2)          'sets an output bytes
CONF:STB 0                'sets data strobe polarity
FORM:LIST BIN             'selects binary output format
```

Sending Data:

```
0001FED3000214530003 --- 4567
                          'data ends with EOI asserted on last
                          character
```

Figure 11 Binary Transfer of 16-bit Wide Data

OUTPUTTING SERIAL DATA WITH A 2303 OR a 2363 SERIAL-TO-DIGITAL INTERFACE

ICS's Models 2303 and 2363 Serial-to-Digital Interfaces convert asynchronous serial data into digital I/O signals. They use the same commands and have the same digital interfaces as their GPIB equivalents. However they do not have a transparent data transfer mode so the user will have to use either the byte out commands (BOn) or data output commands (PO) to output the data. BOn commands send data to the specified port (n) and do not pulse the data strobe output. PO commands send data to the byte(s) configured as outputs and do pulse the data strobe output line. All of the above single address mode examples work with the 2303 and the 2363.

SUMMARY

This application bulletin has described how the to use ICS's GPIB to Digital interface modules and cards to generate serial waveforms. This application note also describes ways to speed up the user's program and how to minimize the number of data bytes needed to generate the serial waveforms. In some cases, the user will be able to reduce the number of data bytes to one byte per bit. The same technique can be used to transfer multiple data bytes to another digital device. The 4803's new BINary format has the highest data transfer rates. HEX or 4833 formats with multiple data bytes per command maximize the data transfer rates for older firmware.

While this application bulletin was written for GPIB interfaces, the Model 2303 and Model 2363 Serial-to-Digital Interfaces can also be used to generate serial waveforms but cannot take advantage of the 4803 and 4863's transparent data transfer capability.