



Transferring Large Data Sets from the CSA over GPIB

Overview

This application note is intended for individuals who will be writing programs to control the CSA using GPIB. First, please refer to the CSA user manual. It contains valuable information on using GPIB with the CSA.

Summary

There are many applications for the CSA. Some of these applications may result in the CSA acquiring large amounts of data. In these situations the GPIB programmer must be aware of potential problems resulting in transferring these large data sets from the instrument to the host PC. In general, the host PC side must choose timeout values and receive buffer sizes appropriate to the transfer times and data sizes expected from the instrument.

Content

The example used in this application note is that of a channel card that is programmed to its maximum number of readings (112000). An example SCPI command that programs a channel card to this value looks like:

```
:Config:Meas 5 1510 112000 IMMEDIATE 1
```

Please refer to the user manual for the format of commands. This command tells channel card number 5 to setup for taking 112000 measurements at 1510 nm. The preceding command is then followed by **:meas?** which actually starts the measurement process and the transferring of data back to the host PC.

In this example, this results in approximately 3 Mbytes of ASCII data to be transferred.

Also, in this example, it takes approximately 25 seconds from the time the **:meas?** command was issued until all of the data transfers to the host PC.

The host PC must setup its timeout values and receive buffer sizes to allow for the transfer times and data sizes to be received from the instrument. For this example, use a timeout value of at least 30 seconds and a receive buffer size of 3.5Mbytes. If you are writing in 'C' and are using the National Instruments (registered) drivers, the code looks like:

```
#define RESPSIZE 3500000 // expected size of response
char *response;        // defines a pointer to returned data
char cmd[100];         // define a string to hold commands we'll issue
ibtmo(devID, T30s);   // sets up a 30 second timeout for GPIB xfers
response = (char*)malloc(RESPSIZE); // allocate space for returned data
if(response == NULL)
{
// if we get here, we cannot allocate the memory. Don't continue.
// perform error handling
}
strcpy(cmd, ":Config:Meas 5 1510 112000 IMMEDIATE 1"); // load command string
ibwrt(devID, cmd, strlen(cmd)); // setup the measurement
strcpy(cmd, ":meas?"); // load command string
ibwrt(devID, cmd, strlen(cmd)); // start the measurement
ibrd(devID, response, RESPSIZE); // read the response
if((ibsta & ERR) || (ibsta & TIMO)) // Examine ibsta for errors
{
// if we get here there was an error or timeout.
// always check for a timeout.
}
}
```

The previous example code works by dedicating the host PC while waiting for a response. The host PC executes the `ibrd(...)` command for the full 25 seconds while waiting for data.

There is another approach that can be taken which will free up the host PC during this time. This involves the use of multiple reads. This approach is a little more complicated and requires some understanding of the processing that occurs within the CSA.

When the CSA gets the `:meas?` command it immediately responds with: `~>` (and some end of line terminators). This is intended to inform the outside world that a possibly lengthy command has begun and a significant amount of time may be needed to process the command. The EOT (End of Transmission) is not sent at this time and will not be sent until the command is fully processed. The processing of the command occurs in two parts. The first part is measurement acquisition where data is being collected by the CSA. In this example this takes about 10 seconds. The second part is the transferring of the measurement data out GPIB. This takes about 15 seconds in this example. At the end of the transmission, EOT is issued.

To perform multiple reads, first set the timeout value to a small timeout. For example, one second after the `:meas?` command is issued, the next read should return: `~>`. The CSA will then be busy acquiring data. Reads that are performed within the next 10 seconds will not return data and will timeout. Finally, after 10 seconds, the measurement data has been collected and the CSA is ready to send it out GPIB. When reads are issued they will contain data. However, even though data is being returned, the reads on the host PC will indicate a timeout. This is because all of the data cannot be returned in one second (the timeout value we set on the host PC), but each read returns a portion of the dataset. These portions can be concatenated together to reconstruct the entire dataset without gaps.

To abort the transfer of data, issue a command to the CSA. A good one to use is `*IDN?` Write this command to the CSA. The transfer will stop. Next issue a read to get the CSA's identification string.
