ICS ELECTRONICS
division of Systems West Inc.

IEEE 488 APPLICATION BULLETIN

EXECUTING PARTIAL READS ON ONE DATA STRING

INTRODUCTION

The normal practice in a GPIB test program is to read data from an instrument with a single read command. However in some cases, the user may want to read only a portion of the data and then go back and read the remainder of the data or just read another chunk of the data. Done incorrectly this will lead to lost data, test program hang ups and EABO timeout errors. This Application Note describes the correct way to make partial reads with ICS's 488-USB GPIB Controller and includes a sample C language program.

GENERAL CONCEPT

GPIB instruments are designed to stop outputting data when the GPIB Bus Controller asserts ATN or stops the handshake by asserting NRFD. If the instrument is unaddressed and then readdressed, its internal state machine may be programmed to discard the unsent portion of its response data. Alternately the instrument may be come confused when readdressed and hang up the bus.

When a standard read type command is used to read twice from the same device, it will unaddress the device and then readdress it as a talker. Depending upon the instrument's design, it may reply with new data, may discard any remaining data or may cause a bus hang up.

There are two possible solutions to this problem: Solution #1 is to use a read command that does not readdress the instrument at the start of the second read. Solution #2 is to disable readdressing so that the standard Read command will not unaddress and readdress the instrument.

PARTIAL READ COMMAND

A command like RcvRespMsg reads data from a previously address talker. If used in a loop, this command can be programmed to read as little as one byte of data each time it is called. Figure 1 on the next page is a listing of a C Language program that uses RcvRespMsg to read an IDN response in one byte chunks. Other GPIB Controllers will have a similar non-readdressing read command.

When doing partial reads, the use must be sure that he or she has satisfied the instrument by either reading all of the data or clearing out the remaining data. Some GPIB instruments will clear their buffers when sent the GPIB Device Clear command. Other GPIB instruments may force you to read all of the data to prevent a hang up. Leaving an unread portion of the data response in the instrument may also cause the instrument to generate a Query Error.

DISABLING READDRESSING

Some GPIB Controllers allow you to disable readdressing on a read or write command. However ICS's Model 488-USB does not disable readdressing at this time. If readdressing was disabled, standard reads could fail. If the user attempts to disable readdressing with the 488-USB, no error is produced but readdressing is not disabled.

SUMMARY

This Application Note has shown how a user can safely conduct partial reads from a GPIB Instrument with ICS's 488-USB GPIB Controller. Figure 1 on the next page is C language example of 1 byte long partial reads. Figure 1 can be used directly with ICS's and any other GPIB Controllers that support the National Instrument 'ib' and 488.2 command sets.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include <winsock.h>

#include "..\ICSDecl.h"


main ()

   {
   int     i, result;
   char    c, buf[4096];


   // GPIB init sequence
   SendIFC (0);
   ibsre (0, 1);

   // Send an *IDN? query string to device-4
   Send (0, 4, "*IDN?", 5, NLend);

   if (ibsta & ERR)
      {
      printf ("Error following the *IDN? send\n");
      return (1);
      }

   // Unlisten all, untalk all, 0 listen, 4 talk
   SendCmds (0, "?_ D", 4);

   if (ibsta & ERR)
      {
      printf ("Error after the SendCmds\n");
      return (1);
      }

   // Pre-fill the buffer with NULL's so it becomes a legal C string
   // after reading the data.
   memset (buf, '\0', sizeof(buf));

   // Read the *IDN? query response
   for (i= 0; !(ibsta & ERR) && !(ibsta & END); )
      {
      RcvRespMsg (0, &c, 1, STOPend);

      if (ibsta & ERR)
         {
         printf ("Error received during RcvRespMsg sequence\n");
         printf ("  ibsta = 0x%04X\n", ibsta);
         printf ("  iberr = %d\n", iberr);

         break;
         }

      if (isprint (c))
         buf[i++] = c;
      }

   result = ibsta;

   printf ("Final result after the read:\n");
   printf ("  string: \"%s\"\n", buf);
   printf ("  count = %d bytes\n", strlen(buf));
   printf ("  ibsta = 0x%04X\n", result);

   return (0);
   }
```

Figure 1 C Language Partial Read Example Program