PAR24B User's Guide

Release 05/01/98

Web: www.symres.com

LIMITED WARRANTY

WHAT IS COVERED

Symmetric Research warrants its PAR24B product will be free from defects in workmanship and materials for one year from the date of original purchase.

WHAT SR WILL DO

Symmetric Research will repair or replace defective PAR24B systems covered under this warranty at no cost to the customer other than shipping. The customer is responsible for shipping to SR manufacturing facilities.

WHAT IS NOT COVERED

Symmetric Research does not warrant the PAR24B product for use with customer provided power supplies or analog input voltages outside the range of values listed in this manual. Incorrectly connecting power or analog inputs may permanently damage the boards.

Furthermore, PAR24B systems that have been customer modified, including but not limited to changes to the analog input voltage range circuitry, are also not covered under this warranty.

Symmetric Research will at its discretion determine when any returned equipment has been run from incorrect power supplies, incorrect analog inputs, or without AGND connected, and is not covered by the terms of this warranty.

Symmetric Research is not liable for any loss, damage, or inconvenience, including direct, special, incidental, or consequential damages, resulting from the use or inability to use the PAR24B product.

TABLE OF CONTENTS

INTRODUCTION

The Symmetric Research PAR24B A/D data acquisition board provides an easy way to acquire high resolution data on your PC from 8 channels at low sampling rates. It has been designed to work in conjunction with EPP/BPP parallel ports supported by most current PCs. Interfacing to the parallel port, the system is ideal for use with PCs such as laptops where adding in an ISA or PCI bus card is not possible.

Based on the Harris HI7190 24 bit sigma delta A/D converter, the SR PAR24B has an individual A/D converter for each channel, minimizing channel cross talk, skew, and settling time. You get the full performance of a dedicated A/D converter on each channel. In addition all data from the A/D converters is FIFO buffered. This allows for continuous data acquisition even with long Windows task switching latencies.

Software included with the board is aimed at familiarizing the user with its technical operation and providing a high level interface. Win 95 32 bit DLL library functions are at the core of the software support, taking care of low level interfacing with the hardware. Example programs written in C show how to carry out basic operations, while finished Win 95 and DOS applications allow the user to immediately begin using the system. Virtual Instrument drivers and finished applications are also provided for National Instruments LabView.

Other items included with the board are a 9 vac power supply, a molded cable for connecting to the PC's parallel port, analog input test leads, complete circuit diagrams, and source code.

We hope you enjoy using the SR PAR24B!

INSTALLATION

Installation of the PAR24B is straightforward. You'll need to connect the power, connect the PC parallel port, install the software, and provide suitable analog inputs. The individual steps in more detail are:

CONNECTING POWER

The PAR24B requires power independent of the PC. Although it communicates its data to the PC over a cable connected to the parallel port, this cable does not supply power. A variety of power options have been provided.

Included with the system is a 9 vac wall transformer. This wall transformer should be plugged into a 110 vac wall socket. Then its 9 vac output connector should be plugged into J602 near the lower left corner of the board.

When properly powered you will see three green LEDs light up on the board. These three LEDs indicate that the digital and analog + and - supply voltages are all at proper levels. All three of these on board supply voltages are derived from the single 9 vac source.

Another possibility for powering the board is to use direct DC power. Two connector options have been provided for using DC power. There is a 6 pin Molex connector J600, and a three screw terminal block J601. If you use DC power, you must supply both +/- 12 vdc. Consult the circuit diagrams and board legend for proper connections.

As with the standard 9 vac power, when properly powered with DC supplies, you will see all three green LEDs light up. IF THE THREE GREEN LEDs DO NOT ALL LIGHT UP, DISCONNECT DC POWER IMMEDIATELY AND DETERMINE WHAT IS WRONG. Any one of the LEDs failing to light indicates that power has not been properly connected to the board.

The board is protected with diodes so that it is unlikely incorrectly connected power supplies will damage it. However, the PAR24B is not protected from excessive overvoltages. Connecting either AC or DC power that is badly out of range may damage the board. Customers using their own power supplies should be aware that they are responsible for providing the correct voltages.

We recommend using the supplied 9 vac wall transformer. There is no measurable improvement in resolution or 60 Hz noise rejection by using DC power. The 9 vac power is simple and easy to connect.

CONNECTING TO THE PC EPP/BPP PARALLEL PORT

Included with the PAR24B board is a 6 ft molded 25 pin D shell cable for connecting to the PC parallel port. The female end of this cable should be connected to J100 in the upper left hand corner of the PAR24B board, while the other end should be connected to the parallel port connector on your PC. Do NOT connect the cable to 25 pin RS232 interfaces. Only connect to PC parallel ports.

The PAR24B has been designed to work with IEEE 1284 type EPP/BPP parallel ports. This type of port has dramatically improved performance over the standard Centronics/SPP parallel port on the original PC. It will also work with standard PS2 style bidirectional BPP parallel ports found on many computers.

Most users with PCs manufactured after 1995 will have EPP compatible parallel ports even though they may not realize it! To make sure you are running in EPP mode, check your CMOS setup. Somewhere in the CMOS menus, you will find an option for selecting the parallel port mode. The available choices usually are:

SPP	Standard Parallel Port mode
bidirectional	IBM PS2 style bidirectional parallel port mode
ECP	Enhanced Communications Port mode
EPP	Enhanced Parallel Port mode

Chose EPP mode for use with the PAR24B board. EPP mode is backward compatible with SPP, so you will be able to continue using other SPP peripherals you may have without reconfiguring your CMOS again. Note that ECP mode IS NOT compatible with EPP mode. ECP is generally used for interfacing with peripherals like external CD ROM players and is not compatible with the EPP mode used by the PAR24B board.

If your CMOS does not have EPP mode, then chose a bidirectional mode if possible. Usually the word bidirectional will be included in the mode description, although there is great variability from computer to computer. We have even found computers that support bidirectional mode when the CMOS is set to SPP!

In addition, you will want to set the parallel port address. The SR software defaults to the IO address 0x378. Most users will already find their CMOS assignment set to this address. Although the SR software can be used with other addresses, we recommend using the 0x378 default.

Users with PC's manufactured before 1995, or lacking EPP/BPP compatible parallel ports may install an Enhanced IO card in their ISA bus. These cards are available at many computer stores. Check to make sure the card has IEEE 1284 compatibility. If it does, it will support EPP mode. These cards are inexpensive, usually $40 or less. SR can also supply these cards if you need one.

If you use an Enhanced IO card in your PC you will probably have to configure it for EPP mode by setting jumpers on the card rather than your CMOS. Refer to the documentation that came with the card. Also configure the add in card for the 0x378 base address if possible.

So how do you verify if the PC's EPP port is communicating with the PAR24B board correctly? After installing the software, run the \sr\par24b\diags\diag.exe program. This program will automatically check to see if a PAR24B board is properly connected and let you twiddle the yellow LED. If the yellow LED toggles, you are correctly connected to the PC. Note that diag.exe depends on par24b.dll. Make sure \sr\par24b\dll is on your execution path.

SOFTWARE INSTALLATION

A single floppy disk in compressed PKZIP format comes with the PAR24B board. To install it, run the install.bat file from the a: or other floppy drive. This will automatically unzip and create the default \sr\par24b directory structure on your hard disk. By default, all SR software products are installed as subdirectories of \sr. We recommend you keep only one copy of the software on your hard disk and that you use the default directory structure. This makes maintenance and installing upgrades easier.

If you have the PAR24B powered on and connected to your PC's parallel port, you can verify correct operation by running the \sr\par24b\diags\diag.exe program. This program is designed to be run from a Win 95 DOS command line. Follow the on screen messages for more information.

At the core of all the 32 bit software supplied with the system is par24b.dll. This DLL library has the low level functions that are required to communicate with the PAR24B. Programs that use par24b.dll must be able to find it at run time. If Win 95 cannot, it will issue an error message letting you know. The best way to inform the system of the location of par24b.dll is to add its location to your execution path. The following command can either be executed from the command line or added to your autoexec.bat file:

```
set path=%path%;\sr\par24b\dll
```

For DOS users, 16 bit software support is also included with the system. This can be found in the \sr\par24b\16bit directory. Please refer there for more information.

Finally, note that you should remove all printer drivers from the parallel port you are using. Under Win 95 multitasking, these drivers can contend for the parallel port while the PAR24B is executing. HP printer drivers are particularly troublesome in this regard. Click on the HP printer driver dialog Exit button to disable it at run time.

Included with the software are many readme.txt files and source code files with comments. We encourage you to refer to these for more information about the software.

CONNECTING ANALOG INPUTS AND ANALOG GROUND

To complete installation you will want to connect analog inputs to the PAR24B board and verify you are getting the correct values.

Inputs to the Harris HI7190 A/D chips are differential. Differential inputs provide considerable noise immunity over single ended inputs, but also require some care to use them correctly. The basics are as follows:

Access the inputs by connecting to the ejector latch header J500 on the front edge of the board. There are eight pairs of pins on this 8x2 header. Each pair of pins has the + and - differential input for a single channel. You can look at the back side of the board to see how the analog traces run, or consult the circuit diagram. There is also a printed legend on the board for the connections.

The A/D converters return a count which is proportional to the difference in voltage between the + and - pins. However, they can only do this within limits. If the absolute common mode voltage is too high then the converters will be beyond their specifications. **To maintain a common analog ground, you must connect the analog ground screw lugs S500 and S501 to your system ground.**

If the analog ground of the PAR24B is floating with respect to the system analog ground then it will likely drift with time and pin the A/D converters. A system that is initially working may appear to start failing. This condition usually does not cause permanent damage. Connecting the PAR24B analog screw lugs to your analog system ground will avoid this. Note also that simply connecting the - input of a channel to your system analog ground is not sufficient.

Normally, the full scale analog input voltage of the HI7190 converters is +/- 2.5 volts. By adding jumpers to the board you can increase this to +/- 10v. Installing jumpers on J400 - J471 will enable simple resistive voltage dividers on the inputs and increase the range to +/- 10v. The board is normally shipped with these jumpers on. Remove them for the +/- 2.5 v range. Consult the circuit diagram for more details.

Note that when running with the resistor dividers enabled the input impedance of the system is lowered. With the +/- 10v range it will be somewhere around 13k Ohms. With the +/- 2.5 v range it will be somewhere around 1M Ohms. Consult the Harris spec sheets for details about the input impedance characteristics.

Finally, note that any analog inputs left floating tend to be very susceptible to noise. When doing tests for channel resolution, or simply to suppress noise, short unused inputs together. A simple jumper can be very handy for this purpose.

Use the \sr\par24b\simple\simp24b.exe program to acquire some values and verify the results.

INSTALLATION CHECKLIST:

* Are ALL THREE green power good LEDs on ? If not, turn power off immediately and find out what is wrong !
* Make sure the \sr\par24b\dll directory is on your execution path so par24b.dll can be found at run time.
* Run the \sr\par24b\diags\diag.exe diagnostic. This will check that you are correctly connected and communicating with the PC's parallel port. If you can't twiddle the PAR24B yellow LED on and off, check your PC's CMOS parallel port settings.
* If diag.exe is successful, proceed on and try out simp24b.exe and scope24b.exe to acquire some values and see the system work.
* Connect a good analog ground from your source devices to the screw lugs S500 and S501 on the front edge of the board. Even though the PAR24B inputs are differential, they still require a good analog ground reference to avoid clipping.
* Do not touch the PAR24B analog input circuitry while in use. Your body has voltages that will easily swamp out 24 bit accuracy.
* Short any channels not in use. Floating channels are susceptible to noise.

DLL LIBRARY SUPPORT

Win 95 32 bit DLL library support is at the core of the software supplied with the SR PAR24B board. With these functions, board operation can be controlled from high level languages without having to know the low level details of how the board operates. These functions can be called from Microsoft Visual C and Basic, and National Instruments LabView. This chapter covers usage from programming environments like Visual C. For information about usage from LabView, see the LabView chapter.

The outline of how to use the par24b.dll functions is fairly simple. First call the Init function to set the sampling rate and other parameters. Once the board has been initialized, the Start function should be called to begin execution. While executing, IsReady should be used to determine when the PAR24B FIFO is half full with acquired data. Then use GetData to move the data from the FIFO into a PC memory array, after which it can be displayed or saved to the hard disk.

There are generally two schemes for determining when IsReady comes true. The first is infinite loop polling. While easy to program, this method wastes considerable PC horsepower. A better way is to set up a PC timer tick that will test the IsReady function slightly more often than the PAR24B FIFO becomes half full. This method greatly reduces the number of PC cycles used in polling and fits in very well with the multitasking capabilities of Win 95.

It is also possible to have the PAR24B interrupt the PC when the FIFO is ready with data. Generally there is little improvement in performance using interrupts over a timer tick. Interrupts are considerably more difficult to use with Win 95 than timer ticks, and we generally recommend using a timer tick in most applications.

When using the par24b.dll library, be sure to include the header file par24b.h in any C source code. The prototypes in this file make sure the correct parameters are passed to the functions. You will also find the defined constants in par24b.h useful for making your programs more readable. Also make sure the par24b.dll library is on your execution path so Windows can find it at run time.

16 bit DOS users can recompile the PAR24B library to a standard static LIB. See the \sr\par24b\16bit directory for examples of how to do this.

The following is a discussion of each par24b.dll function. Refer to the programs simp24b.exe and scope24b.exe for complete examples of how to use the library functions. For a completely bare bones program demonstrating how to use the DLL library, refer to \sr\par24b\examples\sample.c.

INITIALIZE THE PAR24B BOARD

C usage:

```
#include "par24b.h"
int PAR24B_Init(
                        int PortAddress,
                        int PortMode,
                        unsigned int *FifoSize

                        int Dc,
                        int Fp,
                        int Gain,
                        int CalMode,

                        long *HI7190_CrValue,

                        int *ErrorCode
                    );

void PAR24B_SpsToFp( double Sps, int *Fp);
void PAR24B_FpToSps( int Fp, double *Sps);
```

The Init function initializes the PAR24B board. If it is successful it will return a 1, otherwise 0. It takes several parameters controlling various parallel port and HI7190 A/D converter features and can be used as a presence test for the PAR24B in addition to initializing.

The first two parameters, PortAddress and PortMode control the parallel port base IO address and mode that the par24b.dll library will use in communicating with the board. Usually the PortAddress will be 0x378, while the PortMode should be 0 or 1 to indicate PS2 style bidirectional or EPP communications. Use the defined constants in par24b.h to specify these parameters and refer to the installation chapter of this manual for more information about parallel port settings.

On return from Init, FifoSize will have the physical size in bytes of the FIFO that is installed in the system. The value is determined dynamically by Init. If you know the FIFO size and would rather skip the dynamic sizing, pass a NULL pointer for FifoSize. Most PAR24Bs are shipped with a 4K FIFO, however any size from 256 bytes to 32 Kbytes may be installed. The FIFO is from the industry standard 72XX pin compatible series.

The next four parameters control the HI7190 A/D converter chips. These parameters specify their sampling rate and other basic features. Note that all eight HI7190 A/D converter chips are initialized and programmed in parallel with the same parameters. It is not possible to

program an individual HI7190 with parameters that are different than the others. The parameters are as follows:

Dc stands for data coding and not DC voltage or offset. It maps into the dc bit on the HI7190 control register, and controls whether unsigned offset or signed integer data is returned by the A/Ds. The value of this parameter should be 0 for unsigned offset or 1 for signed integer.

Fp sets the filter point for the A/D converters, which in turn directly controls the sampling rate. It maps into the fp bits in the HI7190 control register. To compute the value of Fp corresponding to a particular sampling rate, use the SpsToFp helper function. Note that if you specify a Fp sampling rate that is unacceptable for the HI7190, Init will fail and return 0.

Gain controls the internal HI7190 programmable gain amplifiers. As with the other parameters, this value maps into its respective bits in the HI7190 control register. This is an encoded value and not the gain setting itself. Use the defined constants in par24b.h to set this value. While the HI7190 does have programmable gain, it gives the best performance with a gain of 1. Many applications are best served by using a gain of 1 and then shifting the digital values left to provide a software gain.

CalMode controls the HI7190 calibration mode. This is an advanced parameter. Most applications should use self calibration specified by a value of 1. For more information about the other calibration modes, and Dc, Fp, and Gain parameters see the Harris specification sheet.

If initialization of the HI7190s is successful, the long pointed to by HI7190_CrValue will be filled with the readback value from the HI7190 control registers. Besides checking whether Init returns a 0 or 1, you can check the bits in this value to make sure all HI7190 control register values were written out and read back successfully. To ignore this return value, pass a NULL pointer.

Finally, the last parameter is a pointer to a returned error code value. Besides returning an overall 0 or 1, Init will also fill in the ErrorCode to indicate more precisely why it failed in the event of an error. As with the pointers for the other returned values, pass a NULL pointer to ignore this parameter. See par24b.h for a list of the possible ErrorCode values.

A typical calling sequence for Init would be:

```
#include "par24b.h"

 PortAddress = PAR24B_PORT_ADDRESS_1;
    PortMode = PAR24B_PORT_MODE_EPP;
          Dc = PAR24B_DC_SIGNED;
          Fp = PAR24B_SpsToFp( 30.0);
        Gain = PAR24B_GAIN_1;
```

```
     CalMode = PAR24B_CAL_MODE_SELF;

if ( !PAR24B_Init(
                    PortAddress,
                    PortMode,
                    &FifoSize,
                    Dc,
                    Fp,
                    Gain,
                    CalMode,
                    NULL,
                    NULL
             ) )

       printf( "error, unable to initialize");
```

START AND STOP EXECUTION

```
C usage:
             #include "par24b.h"
             int PAR24B_Start( void);

             int PAR24B_Stop( void);
```

After successfully initializing the PAR24B, you will want to start it executing at some point. Calling Start begins the HI7190 A/D converters executing. Until Start is called, the HI7190 converters are held in idle mode and do not sample their analog inputs. Once executing, the converters begin sampling their analog inputs and performing sigma-delta conversion with the parameters specified in Init.

Usually once execution is started you will want to begin testing for when the PAR24B FIFO becomes half full with acquired data. Use the IsReady function to test for that condition and GetData to transfer FIFO data over to the PC.

At the close of your program you may wish to call the Stop function to put the HI7190 converters back into idle mode and halt their execution. Calling Stop is optional, but if you do not, they will continue executing.

IS DATA READY

```
C usage:
             #include "par24b.h"
             int PAR24B_IsReady( void);

             int PAR24B_Overflow( void);
             int PAR24B_IsEmpty( void);
```

Once the PAR24B has started executing, the function IsReady should be called to see when data is ready for downloading to the PC.

When the PAR24B FIFO is half full, this function will return a 1, otherwise a 0. IsReady should be polled either in an infinite loop or with a timer tick to see when data is ready. Generally polling via a timer tick is preferred because that makes better use of the PC's cpu.

Once data is ready, call GetData to transfer it to the PC. Getting data will automatically set IsReady back to 0 and the application program can then wait until it returns 1 again.

You also may want to check for the PAR24B FIFO overflowing. If you wait too long before calling GetData, it is possible for the FIFO buffering to be exceeded and data lost. At 10 Hz the FIFO will give a total of 17 seconds of buffering before overflow. At higher sampling rates this will be proportionally reduced.

There are two ways to check for overflow. You may simply call the Overflow function. If it returns a 1, overflow has occurred. 0 indicates all is ok. If an overflow has occurred, you can call GetData to go ahead and download data, but some data will have been lost.

The second way to check for overflow is simply to call IsReady immediately after a call to GetData. If it returns 1, then data is coming in so fast that you are unable to keep up with it.

The programs \sr\par24b\examples\sample.c and \sr\par24b\simple\simp24b.c both show how to use the Overflow and IsReady functions to get data from the PAR24B board.

The function IsEmpty is included for users wishing to test the empty flag on the FIFO. When IsEmpty returns 1 then the FIFO is completely empty. If some data has been acquired and saved in the FIFO by the A/D converters, then IsEmpty will return 0.

GET DATA

C usage:

```
#include "par24b.h"
unsigned int PAR24B_GetData(
       long *Values,
       unsigned int Nvalues
       );

void PAR24B_FifoSizeToNvalues(
       unsigned int FifoSize,
       unsigned int *Nvalues
       );

int PAR24B_ReadFifo( long *Values);
```

Once the IsReady function has returned true, call GetData to copy the half full FIFO data to the PC for further processing. To use GetData, pass a pointer to an array to store the values in and the dimension of the array. The dimension of the array cannot be arbitrary! It is fixed by the size of the FIFO you have installed on the PAR24B. To compute the dimension of the Values array in longs, use the FifoSize returned by the Init function and the FifoSizeToNvalues helper function.

How is the data returned by GetData organized? Each 24 bit A/D value is packed in a long 32 bit integer with the high byte set to zero. Depending on how you have set the Dc parameter in the Init function the 24 bit value will either be an unsigned offset binary value, or a 24 bit twos compliment signed value.

Values are returned in multiplexed fashion. The first value from channel 0 will be in Values[0], the first value from channel 1 in Values[1], and so on. The second converted value from channel 0 will be in Values[8], where in general the value from the nth conversion on channel c will be given by:

```
Value at nth conversion from channel c = Values[n*8 + c]
```

The function ReadFifo reads a single sample from each channel. The array pointed to by Values should be dimensioned for 8 longs. This function is included only for those applications needing to read a single set of values at a time out of the FIFO rather than an entire block.

A typical code fragment for using GetData would be:

```
/* Declare an array to save acquired values in */
PAR24B_FifoSizeToNvalue( FifoSize, &Nvalues);
Values = (long *)malloc( Nvalues*sizeof( long));

/* initialize the board and start execution ... */
while ( 1 ) {

       if ( PAR24B_IsReady() ) {

              /* Check for FIFO overflow. */
              if ( PAR24B_Overflow() )
                     Error( "Fifo overflow ...\n");


              /* Get the data, and do something with it. */
              PAR24B_GetData( Values, Nvalues);

                     SaveValuesToDisk( Values);
              }
       }
```

See the programs \sr\par24b\examples\sample.c and \sr\par24b\simple\simp24b.c for complete code listings showing how to use these functions.

USER PROGRAMMABLE OUTPUT BYTE

C usage:

```
#include "par24b.h"
void PAR24B_UserByte( int Value);
```

In addition to A/D input capability, the PAR24B board also has a user programmable digital output byte. The bits of this byte may be set as desired to send signals to external equipment. The physical values are available on the header J220 on the PAR24B board, and the value of bit 0 is reflected by the yellow LED D220.

To program the output byte, simply call UserByte with the desired output value. Note that in order for UserByte to work, the PAR24B board must first be initialized by calling Init. This makes sure the parallel port has been correctly initialized.

See \sr\par24b\diags\diag.c for an example of how to use UserByte.

FINISHED APPLICATIONS / SIMP24B / SCOPE24B

The SR PAR24B comes with two finished application programs you can run immediately after installing the system. These programs not only help you become more familiar with the system, but also may fill your entire acquisition needs. The source code is included for those wanting to modify these programs for custom applications.

The \sr\par24b\simple\simp24b.exe and \sr\par24b\scope\scope24b.exe programs have both been written in C with simp24b.exe being a simple text only command line program, while scope24b.exe has a full Win 95 GUI graphical interface and display.

For more details than covered here, refer to the readme.txt files in the respective directories and comments in the source files.

simp24b.exe

This program is a simple acquisition kernel. It is designed to be run from the Win 95 DOS command line, and will save its data to an output file. Simp24b.exe is appropriate for applications divided between many windows, where one window acquires data while other windows perform downstream processing.

The initialization parameters for simp24b.exe are specified in the file simp24b.ini. Refer to that file for more comments about its syntax and parameters.

If you have trouble running simp24b.exe it may be because the system cannot find par24b.dll at run time. Make sure \sr\par24b\dll is on your execution path so par24b.dll can be successfully found.

A 16 bit version of simp24b.exe can be found in \sr\par24b\16bit. This version of the program is suitable for running from a Windows 3.1 or 16 bit DOS command line. See the readme.txt file in the 16bit directory for more information about how to build 16 bit applications.

scope24b.exe

This is a full blown Win 95 GUI application. It will display your data as horizontal traces on the screen in scope like fashion.

To start scope24b.exe, execute it from the Win 95 command line or the Win 95 Start menu Run command. After the program is running, you will have menu commands across the top of the window that you can select from to control the program features. In addition to controlling items like the sampling rate, you can also control the output file naming, and display

properties. Refer to the online help and readme.txt file for more information about scope24b.exe.

There is no 16 bit version of the scope24b.exe program. It is assumed that users wanting graphical output will at least be running Win 95. However, for LabView users, there is a finished scope like application program written in LabView. Refer to the following chapter and \sr\par24b\labview directory for more information.

LABVIEW SUPPORT

Support for National Instruments LabView is provided in the LabView LLB VI library \sr\par24b\labview\par24b.llb. This LLB contains 3 application and 14 utility VIs. The application VIs include complete data acquisition programs that rely on the underlying utility VIs. The utility VIs are in turn wrappers around the corresponding core par24b.dll DLL functions which supply the low level support for all SR PAR24B software.

To use the PAR24B LLB library, first put \sr\par24b\dll on your path so Win 95 can find the core DLL at run time. You can then access the VIs in par24b.llb by using the File>Open menu option to open \sr\par24b\labview\par24b.llb. Alternatively, it may be more convenient to have those VIs available from the User Libraries button on the Function Palette. You can do this using the Edit>Edit Control & Function Palettes... menu option or by copying par24b.llb into the user.lib subdirectory of your LabView directory. For further details, please refer to file \sr\par24b\labview\readme.txt.

The PAR24B application VIs are described below. For additional information, please refer to the descriptions of the underlying DLL functions or use the Window>Show VI Info... menu option to access the information stored within each VI itself. The VIs have been saved unlocked and with their block diagrams included so that you can modify them as needed to fit your application.

PAR24B LED Demo

This VI initializes the PAR24B and allows you to toggle the on board yellow LED. The yellow LED is bit 0 of the User programmable Byte. Because this VI is so simple, it does not even start acquisition. It just lets you toggle the yellow LED to visually verify that initialization was successful.

To run the demo, first use the LabView Operating tool to set the Port Mode switch to BPP or EPP and the Port Address value as appropriate for your computer. Then press the Run Arrow on the toolbar. If PAR24B initialization is successful, you can press the Toggle LED button. With each press, both the yellow LED indicator on the LabView front panel and the physical LED on the board will change state. To end the demo, press the red STOP button.

PAR24B Meter Demo

This VI samples the PAR24B at a slow rate and displays the results as digital readouts. After the front panel controls have been set, the PAR24B is initialized and acquisition is started. The Is Empty function is called to determine when a set of eight new samples, one for each

channel, is available and then Read Fifo is called to read in those samples. Since the desired sampling rates are slower than the minimum rate that can be assigned to the HI7190 A/D converters, several adjacent samples are acquired and averaged together before being displayed. This additional averaging has the added benefit of improving the resolution of the results.

The results can be displayed as "counts" or volts. When counts is selected, the 24 bits received from the A/D converters are displayed as six hex digits. When volts is selected, the count values are converted to volts by multiplying by a scale factor and adding an offset. The scaling factor and offset are computed in the calibration frame using default values chosen so +/-10v is full scale and there is no offset. For the volts display to be accurate, you will require different calibration values appropriate for each A/D chip.

To run this demo, press the Run Arrow on the toolbar. This enables the Port Mode switch, the Port Address control, the Units switch, and the Sample Period dial. Once you have set these controls to their desired values, press the green START button to begin acquisition. Acquisition will continue until you press the red STOP button to end the demo.

PAR24B Scope Demo

The scope demo samples the PAR24B at a faster rate and displays the results as horizontal waveforms like an oscilloscope. In addition, both a header file containing information like the channels and sampling rate, and a binary data file containing the data points are written out to the LabView default directory that is set from the Edit>Preferences menu option. These files use the same format as those output by the C language programs simp24b.exe and scope24b.exe.

After the front panel controls have been set, the PAR24B is initialized, the header file is written, and acquisition is started. Because the sampling rate for this demo VI is much faster than for PAR24B Demo Meter, a buffer worth of samples is allowed to accumulate before they are read from the FIFO instead of reading them one set at a time. So, the Is Ready function is called to determine when a buffer worth is available and the Get Data function is called to read in those samples. They are then written out and displayed on the multiplot graph.

To run this demo, press the Run Arrow on the toolbar. This enables the Port Mode switch, the Port Address control, and the Sample Rate dial. Once you have set these controls to their desired values, press the green START button to begin acquisition. Acquisition will continue until you press the red STOP button to end the demo.

PAR24B Utility VIs

The following table lists the utility VIs in par24b.llb and their arguments. In order to allow complete flexibility for controlling the flow of execution, dummy inputs and/or outputs are

provided for any function not having at least one input and output. PAR24B Start is an example.

Function	In/Out	Type	Description	Range
PAR24B Init	input	int32	Port mode	0 = BPP, 1 = EPP
	input	int32	Port address	0x378 default
	input	double	Sampling rate	9.541 to 1953.125
	input	int32	Data coding	0 = offset, 1 = signed
	input	int32	Gain	0 to 7 (for 1 to 128)
	input	int32	Calibration mode	0=none, 1=self, 5=SOIG
	output	int32	Init result	1 = ok, 0 = failure
	output	uint32	Fifo size	256 to 32768
	output	int32	HI7190 CrValue	see Harris spec sheet
	output	int32	Error code	0 to 8, see par24b.h
	output	double	True sample rate	
PAR24B Start	input	int32	flow_in	dummy
	output	int32	flow_out	dummy
PAR24B Stop	input	int32	flow_in	dummy
	output	int32	flow_out	dummy
PAR24B Get Data	input	int32*	Data values_in	
	input	uint32	Nvalues	
	output	int32	Get Data result	1 = data read, 0 = not
	output	int32*	Data values	array of int32 values
PAR24B Fifo Size to Nvalues	input	uint32	Fifo size	256 to 32768
	output	uint32	Nvalues	
PAR24B Is Empty	input	int32	flow_in	dummy
	output	int32	Empty result	1 = empty, 0 = not
PAR24B Is Ready	input	int32	flow_in	dummy
	output	int32	Ready result	1 = ready, 0 = not
PAR24B Overflow	input	int32	flow_in	dummy
	output	int32	Overflow result	1 = overflow, 0 = not

PAR24B Read Fifo	input	int32*	Data Values in	
	output	int32	Read Fifo result	1 = FIFO read, 0 = not
	output	int32*	Data Values	array of int32 values

PAR24B Sps To Fp	input	int32	FP in	
	input	double	Samples / sec	9.541 to 1953.125
	output	int32	FP value	2047 to 10

PAR24B Fp To Sps	input	double	Samples / sec in	
	input	int32	FP value	2047 to 10
	output	double	Samples / sec	9.541 to 1953.125

PAR24B User Byte	input	int32	User Byte	0x00 to 0xFF
	output	int32	flow_out	dummy

PAR24B Write Header File	input	int32	Data coding	0 = offset, 1 = signed
	input	int32	Word size	4
	input	int32	Record pts	Nvalues, default 85
	input	int32	Channels	8
	input	path	File name	scope.hdr
	input	string	Id	SR PAR24B
	input	error	Error in	
	input	int32*	Channel array	0,1,2,3,4,5,6,7 default
	input	string*	Channel title	[8], 12 char
	input	float	Sample rate	9.541 to 1953.125
	input	float	Gain	0 => 2**0 to 7 => 2**7
	input	float	Filter coeff	0
	input	float	Filter scale	1
	output	error	Error out	

PAR24B Write Data File	input	path	File name	scope.out
	input	int32*	Wave array	acquired data values
	input	error	Error in	
	output	error	Error out	

APPENDIX A: ELECTRICAL SPECS & CALIBRATION

POWER SUPPLY REQUIREMENTS:

On board, the PAR24B uses three DC voltages to power its chips. These are +5v to power the digital circuitry, and separate +/- 5v to power the analog portion of the board. These voltages are all generated by on board linear regulators.

When using the included 9 vac wall transformer, all three of these voltages are generated automatically from the single AC supply. If using DC power, +/- 12 volts should be connected, and both digital and analog + power will be derived from the +12. Note that if all three green LEDs are on then power is supplied correctly. If not, something is wrong. Turn power off immediately and fix it!

POWER SUPPLY REQUIREMENTS		
Board function	**VOLTAGE**	**CURRENT**
digital power	+5 vdc	100 ma
analog + power	+12 vdc	20 ma
analog - power	-12 vdc	20 ma

MAXIMUM ANALOG INPUT VOLTAGE RANGE:

The absolute maximum analog input voltage range of the HI7190 is +/- 2.5 volts. This range can be extended to +/-10.0 volts by enabling the resistor dividers on each input with the jumpers J400 - J471. Do not exceed the allowed input ranges, or you may damage the A/D chips.

All of the PAR24B analog inputs are differential. This means that the absolute difference between the + and - inputs on any given channel must be less than 2.5 (or 10.0 with the resistor dividers) volts. In addition, for the system to work correctly, ***you must*** establish a common analog ground that is connected to one of the screw lugs on the front edge of the board. The inputs are differential only up to the point where the A/D converters saturate When analog ground floats, it can easily drift to voltages that exceed the saturation levels of the A/Ds even though the absolute difference is still within range. Connecting an analog ground reference avoids this.

CALIBRATION (COUNTS PER VOLT):

Calibration refers to the number of counts per volt that an A/D system has. Generally, it is important to know the calibration if you are doing quantitative work.

No A/D converter can give exactly reproducible results on every conversion. All converters return at best a distribution about a single central value for a given input voltage. The Harris HI7190 is no exception. In addition, components vary slightly from board to board. Users must measure and calibrate their systems for best accuracy in a particular application and should use the values given here only as a guide.

Suppose that the system has its resistor jumpers off for +/-2.5 volt input. Theoretically, the volts per count should be 5.0 v / 2**24 = 0.298 microvolts since the 24 bit count span is spread out over the full input voltage range. This is a small voltage! Inverting this gives the counts per volt:

TYPICAL COUNTS PER VOLT
3355443 (decimal)

Measurements show that sampled devices achieve this value with a typical error of +/-300 counts. However, you must measure your individual A/D converters for an accurate calibration.

In addition to knowing the counts per volt of an individual A/D, you should also measure its typical offset count at 0 volts. The primary source of offset error in the system is within the internal amplifier of each A/D. Measurements have shown that typical offset errors can be as much as a few thousand counts and are highly variable from converter to converter.

There are no potentiometers or hardware adjustments for tweaking the counts per volt and offset on each channel. The most reliable scheme for dealing with absolute system calibration is to adjust for them in software. Subtracting out offsets and multiplying by slope correction factors is the best way to apply individual channel calibration.

APPENDIX B: FIFO BUFFER DEPTH

When programming the PAR24B it is important to know how deep its FIFO buffer depth is. If you are using the supplied software, it will set itself up automatically, but you may be curious to know how to do buffer depth calculations anyway.

The standard factory supplied FIFO is 4 Kbytes deep, where each of the 8 channels is feeding serially into one of the bits. Since there are 24 bits per A/D sample, there are a total of:

4096 bits per channel / 24 bits per sample = 170 samples per channel

when the FIFO is ***completely full***.

The PC is usually signaled that the FIFO has data ready when the FIFO is ***half full***. When the half full signal is received by the PC, it will read 170 / 2 = 85 samples from each of the channels. Thus the typical acquisition buffer for use with the software should be declared at:

85 samples * 4 bytes per sample * 8 channels = 2720 bytes

assuming the 24 bit (3 byte) samples are stored in 32 bit (4 byte) long integers.

Knowing that there are **85** ***samples per channel per buffer*** makes it easy to calculate buffer acquisition time. If you are sampling data at 10Hz then the buffer depth is 8.5 seconds, at 100Hz it is 0.85 seconds. You can experimentally measure these times using a stopwatch and the simp24b.exe program. Just measure the time between screen displays going by.

If you are writing your own custom software and using a timer tick to control when the IsReady function is called, you will want to set the tick rate to something less than the acquisition time per buffer. At 10Hz something like 4.5 sec would be reasonable.

APPENDIX C: PAR24B CIRCUIT DIAGRAMS

The following pages have the PAR24B circuit diagrams. Refer to them for detailed information about power supply connections and jumper settings.

INDEX