Технические характеристики регистратора PAR-4CH

Количество каналов		4
Разрядность АЦП	Бит	24
Тип входов (дифференциальный /с общей землей)		Дифференциальный
Диапазон входного напряжения	В	±10
Коэффициент преобразования при коэффициенте усиления 1	дискр/В	838860
Входное сопротивление	кОм	16
Частота дискретизации	Гц	10-1000
Напряжение питания	В	9-24
Потребляемая мощность	Вт	1
Диапазон рабочих температур	°С	+40-10
Габаритные размеры	мм	139x165x41
Масса	кг	0,5
Производитель		Symmetric Research, www.symres.com
Интерфейсы		EPP/Bidirectional, TTL (4 входа)

SR PARxCH PRODUCT FLYER

The Symmetric Research PARxCH family of A/D boards provide 24 bits of analog to digital conversion for 1, 4, or 8 analog input channels. Based on 24 bit sigma delta A/D converters, these systems provide high resolution at sampling rates in the 10-1000 Hz range, with a maximum rate of 5 kHz. They also feature an individual 24 bit A/D converter per channel for exceptionally low crosstalk and no channel skew.

The boards interface with industry standard EPP or BPP PC parallel ports. They are also equipped with a large 2Mb DRAM FIFO memory buffer for storing data while waiting for the PC. The large hardware buffer provides for continuous data acquisition even with long networking latencies.

Software support is provided at no cost for Windows 2K/XP and Linux, including both GUI and console acquisition applications. Source code and circuit diagrams are included for system customization.

HARDWARE FEATURES

- Ideal for applications requiring 24 bit A/D conversion on 1, 4 or 8 channels at low frequencies
- Individual 24 bit A/D converter per channel
- Highest resolution sampling at 10-1000 Hz
- Differential analog input voltage range of +/- 10 volts
- 4 digital inputs/outputs under program control, PAR8CH samples digital inputs in synch with analog

- PARxCH board sits outside the PC with its own linear power supply for quiet operation
- Designed for use with EPP/BPP PC parallel ports on desktop, laptop, and notebook computers
- 2Mb FIFO buffering minimizes PC cpu load, ideal for networked environments (4Mb for PAR8CH)

- Multilayer board with split power and ground planes for low noise
- Small footprints: PAR1CH = (3.38 x 5”) PAR4CH = (5.25 x 6”) PAR8CH = (7 x 7”)
- Parallel port interface cable, external power supply, and enclosure included
- Compatible with SR PARGPS time stamping module and AMP4CH-DF amplifier

SOFTWARE FEATURES

- Ready to go acquisition applications for immediate data acquisition and display
- DLL function library support for Windows and Linux programming
- Full documentation including source code and circuit diagrams

PRICES

SR Product	Description	Price
PAR1CH	1 channel 24 bit A/D system for PC parallel port	$220
PAR4CH	4 channel 24 bit A/D system for PC parallel port	$575
PAR8CH	8 channel 24 bit A/D system for PC parallel port	$980
PARGPS	PARxCH GPS timing and location accessory	$285

Symmetric Research **Email: info@symres.com** **Web: www.symres.com**

Initial product release 2000 January 2012 cash prices listed

PARxCH User's Guide

Manual Release 01/30/07

Web: www.symres.com

LIMITED WARRANTY

WHAT IS COVERED

Symmetric Research warrants its PARxCH 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 PARxCH 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 PARxCH 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 system.

Furthermore, PARxCH 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 PARxCH product.

TABLE OF CONTENTS

INTRODUCTION

The Symmetric Research PARxCH family of 24 bit A/D data acquisition systems are designed for acquiring high resolution data on the PC at sampling rates from DC to several kHz. The x in the product name represents the number of analog channels and can be 1, 4, or 8. In addition to the analog channels, these A/D boards can also acquire one bit of information from each of 4 digital channels.

Connecting to the PC parallel port, the PARxCH sits outside the PC in its own vinyl covered steel enclosure for quiet noise free operation. Because it interfaces to the parallel port, the system is easy to use with closed PCs such as laptops.

Based on the Burr Brown ADS1210 24 bit sigma delta A/D converter, the PARxCH has an individual converter per channel architecture. This minimizes channel cross talk, skew, and settling time. You get the full performance of a dedicated A/D converter on each channel. The AS1210 converters are "instrument grade" meaning they maintain full accuracy right down to DC. In addition, data from the A/D converters is buffered with a deep 2Mb memory array on the PARxCH. This allows for continuous data acquisition even with long PC task switching latencies.

Software that comes with the board has both high level finished applications ready for immediate use and low level function libraries for those wishing to do custom programming. Support is provided for DOS, Win9x, Win2000/XP, Linux, and LabView. The Win2000/XP and Linux software features true kernel mode drivers for good performance and reliable operation.

Other items included with the system are a 9vdc wall transformer, a cable for connecting to the PC parallel port, an analog input terminal board, and circuit diagrams. Everything needed to be up and running right away.

We hope you enjoy using the SR PARxCH

INSTALLATION

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

CONNECTING POWER

The PARxCH requires power independently of the PC. Included with the system is a 9vdc wall transformer. This wall transformer should be plugged into a 110vac wall socket, and the 9vdc connector plugged into either of the two power jacks on the back of the PARxCH. The second power jack is available for daisy-chaining power to optional equipment such as the PARGPS time stamping unit. Many of our international customers will have been provided with a 220vac wall transformer. Refer to the label on the wall transformer for compatibility with your local wall power.

Once plugged in, you can verify the wall transformer is on by checking the right hand side green LED near the power connector on the back of the PARxCH. If it is on, your wall transformer is plugged in and powered on.

The left hand side green LED on the back of the PARxCH near the DB25 parallel port connector indicates whether the PC has sent a "power up" or "power down" signal to the PARxCH. In its power down mode, the PARxCH draws very little current. This can be useful for applications requiring only occasional data acquisition, helping to preserve the life of field batteries etc. Turning off your PC will also automatically power down the PARxCH.

The PARxCH can be powered with a wide range of wall transformers. Any AC or DC wall transformer with a voltage between 9 and 24 volts is acceptable as long as it has a 2.1mm, center-plus plug. If you have a choice, chose a voltage nearer to 9 volts rather than 24. Running at lower power supply voltages will reduce the heat dissipation of the internal regulators. If you are running the PARxCH outside of its enclosure in a PC104 application, you can also use the alternative power connector. The exact style and location of this connector depends on which of the PARxCH boards you are using, so please consult the circuit diagrams and board legend for proper connections.

The PARxCH is protected with diodes so that it is unlikely incorrectly connected power supplies will damage it. However, the PARxCH 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. Please read the limited warranty at the beginning of this manual.

CONNECTING TO THE PC PARALLEL PORT

Included with the PARxCH 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 the D-shell connector on the back of the PARxCH, 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 or to the 25 pin analog inputs on the PAR8CH. Only connect to PC parallel ports.

The PARxCH has been designed to work with IEEE 1284 type EPP/BPP parallel ports. This type of port has significantly improved performance over the standard Centronics/SPP parallel port on the original PC. The PARxCH will also work with standard PS2 style bi-directional 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. Typical CMOS parallel port options are:

SPP	Standard Parallel Port mode
bi-directional	IBM PS2 style bi-directional parallel port mode
EPP	Enhanced Parallel Port mode
ECP	Enhanced Communications Port mode

Chose EPP mode for use with the PARxCH. 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.

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

The EPP and bi-directional modes are also available indirectly as sub-modes of ECP. The sub-modes for straight ECP and the common ECP/EPP variation are often implemented differently from machine to machine. So you should run the diagnostic program in the diags directory to verify which modes work on your machine for any given CMOS setting.

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 wish to consider installing 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 base address 0x378 if possible.

So how do you verify if the PC's EPP port is communicating with the PARxCH board correctly? After installing the software, run the diag program in the diags directory giving std as the command line option. This will automatically check to see if a PARxCH board is properly connected and let you twiddle the front panel yellow LED. If diag succeeds and the yellow LED toggles, you are correctly connected to the PC.

SOFTWARE INSTALLATION

A CD-ROM is included with the PARxCH. Software for each supported operating system can be found in the respective directory. Each OS subdirectory includes a readme.txt file, a compressed PKZIP or Linux gzipped tar format file, and an installation batch or script file. To install the PARxCH software, change to the OS subdirectory appropriate for your computer and run the install batch script. This will automatically unzip and create the default \sr\parxch or Linux /usr/local/sr/parxch directory structure on your hard disk.

We highly 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. See the readme.txt files on the CD for more information.

For Win2000/XP and Linux installations, besides copying the software to your hard disk, you must also install a kernel mode device driver. Administrator or root permissions are required for this step, but once the driver is installed, ordinary users can use it freely.

To install the kernel mode device driver, change to the PARxCH driver subdirectory on your hard disk and run the indriver program with no arguments to see a list of valid driver names. Then run indriver again with the name of driver associated with the parallel port you are using.. You can remove the device driver using this name and the rmdriver program.

For example, to install the driver on LPT1 at address 0x378 and interrupt 7 with the PAR8CH as the default model use:

> indriver PAR8CH 0x378 7

To remove it use:

> rmdriver SrParXch0

If you forget the exact assigned device driver name, you can find it using the showdriver utility or on a list of installed drivers provided by the OS. For Win2000/XP use the Device Manager and look under the SR Instrumentation group. For Linux, run /sbin/lsmod. For additional information on installing the device driver, please refer to the readme.txt file in the driver subdirectory.

If you have the PARxCH powered on and connected to your PC's parallel port, you can verify correct operation by running the diag program located in the diags directory. This program is designed to be run from a command prompt. Follow the on screen messages for more information.

For programmers, the core part of the software supplied with the system is a collection of low level functions that are required to communicate with the PARxCH. For the 32 bit Windows systems, these functions can be linked in statically or used dynamically from the Dynamic Link Library parxch.dll. Programs that use parxch.dll dynamically must be able to find it at run time or Windows will give an error message. The best way to inform the system of the location of parxch.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\parxch\lib
```

For Linux systems, the core low level functions can also be linked in statically or used dynamically from the shared library parxch.so. Programs using the shared library dynamically should set the LD_LIBRARYPATH environment variable so the library can be found at runtime. One way to do this is by executing the following command line or adding it to your .profile file:

```
LD_LIBRARYPATH=$LD_LIBRARYPATH:/usr/local/sr/parxch/lib ; export  \
$LD_LIBRARYPATH
```

Finally, note that the PARxCH requires exclusive access to the parallel port. Do not daisy chain other physical devices or software drivers from the PARxCH port. Under multitasking OS's, these drivers can contend for the parallel port while the PARxCH is executing. HP printer drivers are known to cause trouble in this regard. Click on the HP printer driver dialog Exit button to disable it at run time. **Note:** While daisy-chaining devices off the parallel port will not work, it is perfectly fine to daisy-chain power to other devices like the PARGPS as long as the total current draw does not exceed the wall transformer specs.

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 PARxCH board and verify you are getting the correct values.

Inputs to the PARxCH are differential. Differential inputs provide significant noise immunity over single ended inputs, but also require some care to use them correctly. The basics are as follows:

Access the analog inputs by connecting to the 25 pin D-shell connector on the left hand side of the front panel. The input pins are organized into pairs starting from the left of the connector, where the (+,-) signal pairs are grouped on the connector. For the PAR1CH, channel 0 is on pins (1,6). For the PAR4CH, channels 0 to 3 are on pins (1,9) (2,10) (3,11) (4,12) respectively. For the PAR8CH, channels 0 to 7 are on pins (1,14) (2,15) (3,16) (4,17) (5,18) (6,19) (7,20) (8,21) respectively. The remaining pins are analog ground. Quick reference information is also printed on the label on the bottom of the PARxCH enclosure.

To make simple wire connections to the analog inputs, you may wish to use the TRM15E or TRMxxV terminal board included with the system. Note, that the numbers listed along the screw edge of the terminal board are not the same as the D-shell pin numbers. The terminal board is numbered so that screws (1,2) are channel 0 (+,-) etc. A number of other analog input connector options such as solder-cup D-shells are also available from SR.

The A/D converters return a count which is proportional to the difference in voltage between the + and - input 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 at least one of the analog ground pins to your analog ground reference. The analog ground pins are 2-5 and 7-9 for the PAR1CH, 5-8 and 13-15 for the PAR4CH, and 9-13 and 22-25 for the PAR8CH.**

If the analog ground of the PARxCH is floating with respect to the analog ground of your input circuit, 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 PARxCH analog ground pins to your input analog ground reference will avoid this. Note that simply connecting the - input of a PARxCH channel to your analog ground will work, but essentially turns the differential inputs into single ended.

The full scale analog input voltage range of the PARxCH is +/-10v. This means that if the + pin is at +10v and the - pin is at -10 volts, for a total difference of +20v, you will record the full positive counts. The full negative counts are recorded when the situation is reversed with -10v on the + pin and +10v on the minus. This means the 2**24 possible digital counts are spread over a range of 40 volts, which results in a scale factor of roughly 420,000 counts per volt or about 2 microvolts per count.. The system will survive modest overvoltages without any damage, but **overvoltage conditions should be avoided**. For example, do not

apply a static shock to the analog input pins. Symmetric Research reserves the right to determine when systems have been damaged by excessive overvoltage. See the warranty page at the beginning of this manual.

Finally, note that any analog inputs left floating tend to be susceptible to noise. When doing tests for channel resolution, or simply to suppress noise, short unused inputs together.

To acquire some values and verify the results, use the simp or scope programs.

CONNECTING DIGITAL I/O

The 15 pin D-shell connector on the right hand side of the front of the PARxCH enclosure is for digital input and output. Pins 1-4 are for input and pins 5-8 are for output. Pins 9-15 are digital ground. As with the analog input D-shell connector, a TRM15E or TRM15V terminal board can be attached to the digital I/O D-shell connector for simple wire connections. Note, that the numbers listed along the screw edge of the terminal board are not the same as the D-shell pin numbers. The terminal board is numbered so that screws (1,2) are pins 1 and 9, etc. The yellow LED provides an additional digital output bit that is under program control.

INSTALLATION CHECKLIST:

* Plug in the wall transformer and check the green LED on the right hand side of the PARxCH back panel. If it is not on, the wall transformer is off or otherwise failed.

* Connect the supplied 25 pin D-shell cable to your PC parallel port and run the diagnostic program in the diags directory. This will show whether the PC's parallel port is correctly connected or not. If diag reports an error, check your CMOS settings for the correct parallel port mode. Also check to be sure the kernel mode device driver is installed for Win2000/XP and Linux.

* Connect analog signals from your source devices to the input pins on the analog input D-shell connector on the left. Be careful to avoid static discharges when touching the analog input pins.

* Try out the simp and scope programs to acquire some data and see the system work. If you are using dynamically linked libraries, make sure the \sr\parxch\lib directory is on your execution path or Linux LD_LIBRARY path so parxch.dll or parxch.so can be found at run time.

* Connect an analog ground from your source devices to the analog ground pins on the analog input D-shell connector. Even though the PARxCH inputs are differential, they still require an analog ground reference to avoid clipping.

* Do not touch the PARxCH analog input circuitry while in use. Your body has voltages that will easily corrupt 24 bit accuracy. Short out any channels not in use. Floating channels are susceptible to noise.

FINISHED APPLICATIONS
SIMP, SCOPE AND OTHER PROGRAMS

The SR PARxCH comes with finished application programs you can run immediately after installing the system. These programs will help you become more familiar with the system and may also fill your entire acquisition needs. The source code is included for those wanting to modify these programs for custom applications.

The simp and scope programs have both been written in C. Simp is a simple text only command line program, while scope has a full Windows GUI graphical interface and display. Under Linux, only the simp program is available.

In addition to these basic data acquisition programs, the PARxCH software also includes several other interesting programs. These are briefly described below.

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

simp

This program is a simple console program for text based data acquisition and is located in the simple directory. It is designed to be run from a command prompt and save data to an output file. Simp is appropriate for applications divided between many windows, where one window acquires data while other windows perform downstream processing.

The initialization parameters for simp are specified in the file simp.ini. Refer to inisyntx.txt for details about its syntax and parameters.

scope

This is a Windows GUI application and is located in the scope directory. It displays data as horizontal traces on the screen in oscilloscope like fashion.

To start scope, execute it from the Windows command line or the Windows Start menu Run command. When the program is running, you can select from menu commands to control the program features such as sampling rate, output file names and display properties.

Besides using the scope menus, you can also control the program features from an initialization file. This file can be given on the command line when starting scope or read in from the Options menu after scope is running. A default initialization file scope.ini is

provided, but you can also generate one that reflects your current scope menu settings at any time from the Options menu.

Refer to the online help and readme.txt file for more information about scope and refer to the scope.ini initialization file and inisyntx.txt for details about its syntax and parameters.

There is no DOS or Linux version of the scope program since the graphical user interface requires the Microsoft Foundation Class (MFC) library and must be able to find the MFC DLL files at run time. It is assumed that users wanting graphical output will at least be running Windows 95. However, for LabView users, there is a finished scope like application program written in LabView. Refer to the following LabView chapter for more information.

view

This is a Windows GUI program located in the view directory for viewing previously acquired .OUT or .DAT data files as horizontal traces on the screen. You can easily scroll forwards and backwards through the data files. There is no DOS or Linux version of this program.

diag

This is a simple diagnostic program that lets you verify your board is functioning correctly. It is located in the diags directory and tries several parallel port address and mode combinations to determine which is the best setting for your computer. It also allows you to toggle the yellow LED on the PARxCH front panel.

sample

Located in the examples directory, sample is a very basic text only program to acquire and display data. It shows the essential features of interacting with the PARxCH. For full featured acquisition, see the simp program in the simple directory.

meter

This is a command line, text only data acquisition program that demonstrates one way to use the PARxCH at slower sampling rates. It is located in the meter directory and includes comments describing how to determine the channel offsets for use in a calibrated system. For LabView users, there is a finished meter like program with a graphical interface. Refer to the following LabView chapter for more information.

2task

Multitasking systems such as Windows and Linux allow several tasks or programs to run at the same time. The 2task directory includes a demo batch or script file that shows how to run the simp acquisition program while simultaneously running a "downstream" processing program to perform some additional functions on the acquired data. Under Linux, the acquisition task is run in the background and the processing task is run in the foreground. While under Windows, both tasks are run in separate windows.

digio

This is a command line, text only program that lets you test the digital I/O. See the comments at the top of the source file for information on how to set up a simple loopback test jig using a terminal board and a few wires.

Conversion programs: out2asc, dat2asc, out2sud, dat2sud, etc

The convert directory has several subdirectories that contain programs for converting the .out or .dat binary data files output by simp and scope into other formats. Currently, converters are provided for ASCII text and IASPEI SUDS formats. Under Linux, converters for the Seismic Unix (SU) format are also provided. Additional converters will be added in the future. The existing conversion programs can also serve as an example for writing customized converters to your preferred data format.

LIBRARY FUNCTIONS

The library functions are at the core of the software supplied with the SR PARxCH. They allow users to control board operation from high level languages without having to know the low level details of how the system operates. These functions can be statically linked to C programs for any OS. In addition, they are available as a Dynamic Link Library (DLL) under Windows and as a shared library (.so) under Linux. When used as a DLL, these functions can be called from other high level programming languages such as Microsoft Visual 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. For data acquisition applications that are ready to run with no programming required, see the Finished Applications chapter.

The outline of how to use the PARxCH library functions is fairly simple. First call the Open function to open the driver and initialize the sampling rate and other parameters. Once the board has been initialized, call Start to begin acquisition. Then use ReadData to move the data from the PARxCH FIFO into a PC memory array, after which it can be displayed or saved to the hard disk. Also, the Ready function can be used to determine when the PARxCH contains at least one point of acquired data. When you are done, call Stop and Close to stop acquiring data and close the driver.

There are generally two schemes for determining when to call ReadData. 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 calls ReadData in about the time the PARxCH takes to acquire the amount of data you want to read. This method greatly reduces the number of PC cycles used in polling and fits in well with the multitasking capabilities of Windows or Linux.

It is also possible to have the PARxCH 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 Windows than timer ticks, and we generally recommend using a timer tick in most applications.

When using the parxch library functions, be sure to include the header file parxch.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 parxch.h useful for making your programs more readable. When using dynamic linking, make sure the parxch.dll or parxch.so library is on your Windows execution path or Linux LD_LIBRARY path so it can be found at run time.

The following is a discussion of each PARxCH library function. Refer to the programs simp.c and scope.cpp for examples of how to use the library functions. For a bare bones program demonstrating how to use the library, refer to sample.c in the examples directory.

OPEN AND CLOSE DEVICE

```
C usage:
       #include "parxch.h"

       DEVHANDLE ParXchOpen(
                           char* DriverName,
                           int XchModel,
                           int PortMode,
                           double Sps,

                           double *ActualSps,
                           int *Error
                           );

       int ParXchClose( DEVHANDLE handle );
```

The Open function opens the PARxCH device driver and provides a device handle that is passed to the remaining library functions. It also initializes the PARxCH board using default values based on the requested sampling rate. Open should be the first library function called.

It returns a valid device handle on success and the value BAD_DEVHANDLE otherwise. If the optional Error argument is provided, it is filled with an error code giving more detail about why the open failed. See parxch.h for a list of the possible error codes. Pass NULL to ignore this parameter.

Open takes a few parameter that identify the specific driver and model and control some parallel port and ADS1210 A/D converter features and can be used as a presence test for the PARxCH in addition to initializing it. Advanced users wanting more control over the ADS1210 parameters can use the FullOpen function discussed later in the Advanced Functions – FullOpen section.

The DriverName parameter identifies the device driver and therefore the parallel port base I/O address. The XchModel parameter identifies which member of the PARxCH family is being used: the PAR1CH, PAR4CH, or PAR8CH. The PortMode parameter controls the parallel port mode that the parxch library will use in communicating with the board. Usually DriverName is “SrParXch0” and PortMode is 0 to 3 indicating PS2 style bi-directional, EPP, ECP/BPP, or ECP/EPP communications respectively. Use the defined constants in parxch.h to specify these parameters and refer to the installation chapter of this manual for more information about parallel port settings.

The Sps parameter is the desired number of samples per second. Because of how the A/D converters work, not all sampling rates can be achieved. Open selects the closest achievable rate and returns that value in the parameter ActualSps.

The Close function cleanly shutdowns the device driver. It should be the last library function called. In some cases, it may be called implicitly at program termination. However, you should not rely on this.

START AND STOP EXECUTION

C usage:

```
#include "parxch.h"

void ParXchStart( DEVHANDLE handle );

void ParXchStop( DEVHANDLE handle );
```

After successfully opening the device driver and initializing the PARxCH board, the next step is to start acquisition by calling Start. Until Start is called, the ADS1210 A/D converters are held in idle mode and do not sample their analog inputs. Calling Start signals the converters to begin sampling their analog inputs and performing sigma-delta conversion.

Once acquisition is started you can use the Ready function to identify when the PARxCH FIFO contains acquired data and the ReadData function to transfer that data over to the PC.

After you are done acquiring, call the Stop function to put the ADS1210 converters back into idle mode and halt their execution. Calling Stop is optional, but if you do not, the converters will continue executing until the Close function is called.

If you want to re-start acquisition once the A/D converters have been halted, you should re-initialize the PARxCH by first calling Close and Open before calling Start again. This ensures that the A/D converters on the board are in a known state so they will begin executing correctly.

READ DATA

```
C usage:
            #include "parxch.h"

            unsigned int ParXchReadData(
                  DEVHANDLE handle,
                  long *Values,
                  unsigned int Nvalues,
                  int *Error
                  );
```

ReadData copies data from the PARxCH FIFO to the PC where it is available for further processing. It also implicitly checks whether data is ready before reading and whether overflow or bad on board voltage has occurred. Because of these checks, ReadData may be the only function you need besides Open/Close and Start/Stop.

To use ReadData, you need an array to hold the data values and the dimension of that array, Nvalues. ReadData reads as many data points as possible into that array up to Nvalues. It returns the number of points actually read. This will always be a multiple of the number of analog channels (x), but may be zero if no complete data values are ready yet. To read a single point for each channel, just pass 1, 4, or 8 for Nvalues as appropriate.

The optional Error parameter will be filled with 0 if all is well and with an error code if something has gone wrong. See parxch.h for a list of the possible error codes. You can ignore this parameter by passing NULL.

How is the data returned by ReadData organized? Values are returned in multiplexed fashion. When using the PAR8CH, for example, 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]
```

Each 24 bit A/D value is stored as a long 32 bit integer. The format of this number is a standard 32 bit two's compliment signed value, unless you request an unsigned offset binary value by setting the df parameter in the advanced FullOpen function. See the Advanced Functions - FullOpen section for a discussion of the df parameter and Appendix B for more details on the possible data formats.

A typical code fragment for using ReadData would be:

```
/* Declare an array to contain acquired values. */
Values = (long *)malloc( Nvalues * sizeof( long ) );

/* Open driver and start execution ... */
...

/* Acquire and process data. */
while ( 1 ) {

      Nremain = Nvalues;
      NextPt  = 0;

      /* Get Nvalues worth of data. */
      while ( Nremain > 0 ) {
            Newpts = ParXchReadData(   handle,
                                       &Values[NextPt],
                                       Nremain,
                                       &Err );
            Nremain -= Newpts;
            NextPt  += Newpts;

            /* Check for FIFO overflow and other errors. */
            if ( Err != PARXCH_ERROR_NONE )
                  Error("Error %s\n",PARXCH_ERROR_MSG[Err]);
            }

      /* Do something with data. */
      SaveValuesToDisk( Values );

      }
```

See sample.c in the examples directory and simp.c in the simple directory for complete code listings showing how to use this function.

READ DATA WITH DIGITAL AND/OR GPS

C usage:

```
#include "parxch.h"

unsigned int ParXchReadDataWithDigital(
       DEVHANDLE handle,
       long *Values,
       unsigned int Nvalues,
       int *Error
       );

unsigned int ParXchReadDataWithGpsMark(
       DEVHANDLE handle,
       long *Values,
       unsigned int Nvalues,
       int *Error
       );

unsigned int ParXchReadDataAll(
       DEVHANDLE handle,
       long *Values,
       unsigned int Nvalues,
       int *Error
       );
```

In addition to the standard ReadData function which copies the acquired analog data from the PARxCH FIFO to the PC, there are three variants which provide additional data. While the argument list is identical for all the read data functions, the size of the Values array will differ and must be large enough to hold all the data being requested.

Because of additional firmware enhancements beyond those found in the 1 and 4 channel units, the PAR8CH can record the values of the digital inputs when each analog sample is ready. Requesting this digital data adds one additional channel beyond the 8 analog channels. The low 4 bits of this additional 32-bit integer channel contain the values of the 4 digital input lines sampled when the related analog data is ready.

Requesting GPS data adds one additional channel beyond the 1, 4, or 8 analog channels. This additional channel contains the PPS event number, often referred to as the GPS mark, which is used to correlate the PPS with its associated serial NMEA data. For the PAR8CH, requesting GPS data also adds a second additional channel containing the results of an on-board counter. This on-board counter records the number of 800ns counts between the time the PPS signal arrives and the time when the next analog data sample is ready. Because this

counter is unaffected by interrupt latencies in the OS, the PAR8CH provides a further improvement to the already accurate GPS time-stamping available with the 1 and 4 channel units.

If both digital and GPS data are requested for the PAR8CH, there will be 3 additional channels as described above. However, because the PARGPS PPS signal will be coming in on digital input 3, that line is no longer available for simple digital data. So the corresponding bit is zeroed out in the returned digital channel.

As with the standard ReadData, the optional Error parameter for all the variants will be filled with 0 if all is well and with an error code if something has gone wrong. See parxch.h for a list of the possible error codes. You can ignore this parameter by passing NULL.

DATA READY / OVERFLOW

C usage:

```
#include "parxch.h"

int ParXchReady( DEVHANDLE handle );

int ParXchOverflow( DEVHANDLE handle );
```

Once the PARxCH device driver is opened and execution has been started, you can use Ready and Overflow to check whether the PARxCH FIFO is empty or full. They can be polled in an infinite loop or with a timer tick. Generally polling via a timer tick is preferred because that makes better use of the PC's CPU.

When the FIFO is empty, Ready returns 0. Ready returns 1 as soon as any data is available in the FIFO. Once all the data has been read out of the FIFO and transferred to the PC by calling ReadData, Ready will again return 0.

When acquiring at sampling rates lower than about 40 Hz, it may be easiest to use the implicit ready check inside ReadData rather than calling Ready explicitly. The reason is that extra averaging, beyond that in the converters themselves, is done in the device driver software to ensure high resolution below 40 Hz. ReadData will return 0 until it is able to read a complete averaged data point, but Ready will return 1 as soon as a single raw data point is in the FIFO.

You may also want to check for the PARxCH FIFO overflowing. If you wait too long before calling ReadData, it is possible for the FIFO buffering to be exceeded and data lost. The PARxCH buffer is actually 2Mb in size (4Mb for the PAR8CH)! Depending on the sampling rate, you can wait a long time before an overflow will occur.

When the FIFO is empty or partially filled, all is ok and Overflow returns a 0. Once the FIFO becomes full, Overflow returns 1 and latches this information until the FIFO is reset. This means that once an overflow has occurred, you can call ReadData to download the valid data currently in the FIFO. However, any additional data arriving after the overflow would be corrupt, so it is discarded and Overflow continues to return 1 even though the FIFO may no longer be full. To re-initialize the PARxCH and reset the FIFO, call Stop, Close, Open and Start.

The programs meter in the meter directory and simp in the simple directory show how to use these functions while getting data from the PARxCH board.

GPS

```
C usage:
             #include "parxch.h"

             void ParXchAttachGps( DEVHANDLE handle
                                   DEVHANDLE GpsHandle,
                                   int *Error );

             void ParXchReleaseGps( DEVHANDLE handle );
```

The PARGPS board can be used to accurately time stamp data acquired by the PARxCH. For the two boards to work together in this fashion, both the boards and their drivers must be connected to one another. The physical link is made by connecting the PPS signal from the PARGPS to digital input 3 on the PARxCH. The software link between the drivers is made by calling the AttachGps function and passing in the handle to an opened PARGPS driver. The ReleaseGps function should be called to clean up this software when it is no longer needed.

USER DIGITAL I/O & USER LED

C usage:

```
#include "parxch.h"

void ParXchUserIoRd( DEVHANDLE handle, int *Value );
void ParXchUserIoWr( DEVHANDLE handle, int Value );

void ParXchUserLed( DEVHANDLE handle, int State );
```

The PARxCH board has user programmable digital I/O with 4 input bits and 4 output bits that can be used to communicate signals to or from external equipment. The physical signals are available on the 15 pin D-shell connector on the right front of the enclosure or on the upper right edge of the PARxCH board. An additional output bit is connected to the yellow LED.

To read the input bits, call UserIoRd. To program the output bits, call UserIoWr with the desired output value. Call UserLed with 1 to turn the yellow LED on and with 0 to turn it off. Before these functions will work, Open must be called to open the PARxCH device driver and initialize the parallel port.

In addition to digital input, the inverted value of bit 3 is mapped to the parallel port interrupt. Enable or disable this function with the InterruptEnable and InterruptDisable functions discussed below.

See diag.c in the diags directory for an example of how to use UserLed.

PC INTERRUPT

C usage:

```
#include "parxch.h"

void ParXchInterruptEnable( DEVHANDLE handle );

void ParXchInterruptDisable( DEVHANDLE handle );
```

The PC can be interrupted by sending a signal over the parallel port. These functions enable and disable this capability. Before these functions will work, Open must be called to open the PARxCH device driver and initialize the parallel port.

VOLTAGE GOOD

```
C usage:
       #include "parxch.h"

       int ParXchVoltageGood( DEVHANDLE handle );
```

VoltageGood returns 1 if the voltage on the PARxCH board is good (within 1% of 4.75 volts). It returns 0 otherwise. Open must be called to open the device driver and initialize the parallel port before this function will work.

The VoltageGood function reports the state of the voltage good signal. This signal is sent from the PARxCH board via the parallel port's Parallel Error pin and is active low. When the PARxCH is not properly receiving power from the wall transformer, it does not have any power available to drive the voltage good signal high to indicate the bad voltage. In this situation, it relies on the PC having a pull-up resistor for this pin so floating inputs are brought high. Most PC's include this pull-up, but some do not. You may wish to add a pull-up to your computer if that is the case.

LIBRARY REV

```
C usage:
       #include "parxch.h"

       void ParXchGetRev( int *Rev );
```

GetRev fills the Rev argument with an integer representing the current revision number of the parxch library. For example, a value of 201 indicates version 2.01.

ADVANCED FUNCTIONS - FULLOPEN

```
C usage:
       #include "parxch.h"

       DEVHANDLE ParXchFullOpen(
                           char* DriverName,
                           int XchModel,
                           int PortMode,

                           int Df,
                           int GainLog,
                           int TurboLog,
                           int Decimation,
                           int ExtraDecimation,
                           int Unused,

                           long *ADS1210_CrValue,
                           int *Error
                     );
```

FullOpen opens the PARxCH device driver and initializes the PARxCH board while allowing advanced users more control over the initialization parameters than the simpler Open. FullOpen returns a valid device handle on success and the value BAD_DEVHANDLE otherwise. If the optional Error argument is provided, it is filled with an error code giving more detail about why the open failed. FullOpen takes several parameters controlling various parallel port and ADS1210 A/D converter features and can be used as a presence test for the PARxCH in addition to initializing it.

The DriverName parameter identifies the device driver and therefore the parallel port base I/O address. The XchModel parameter identifies which model or member of the PARxCH family is being used: the PAR1CH, PAR4CH, or PAR8CH. The PortMode parameter controls the parallel port mode that the parxch library will use in communicating with the board. Usually PortName is “SrParXch0” and PortMode is 0 to 3 indicating PS2 style bi-directional, EPP, ECP/BPP, or ECP/EPP communications respectively. Use the defined constants in parxch.h to specify these parameters and refer to the installation chapter of this manual for more information about parallel port settings.

The next five parameters control the ADS1210 A/D converter chips by specifying their sampling rate and other basic features. Note that all four ADS1210 A/D converter chips are initialized and programmed in parallel with the same parameters. It is not possible to program an individual ADS1210 with parameters that are different than the others. The

defined constants in parxch.h provide an easily readable way to work with these parameters. The parameters are as follows:

Df stands for data format. It maps into the df bit in the ADS1210 control register and controls whether signed integer or unsigned offset data is returned by the A/Ds. The value of this parameter should be 0 for signed integer or 1 for unsigned offset.

GainLog controls the internal ADS1210 programmable gain amplifiers and maps into the gain bits in the ADS1210 control register. This is an encoded value and not the gain setting itself. Use the defined constants in parxch.h to set this value. While the ADS1210 does have programmable gain, it gives the best performance with a gain of 1 (GainLog = 0). Most applications are best served by using a gain of 1 and then shifting the digital values left to provide a software gain.

TurboLog maps into the turbo bits in the ADS1210 command register, controls the internal ADS1210 modulator and affects the resolution. Use the SpsGainToTde helper function to compute the values of turbolog, decimation, and extradecimation corresponding to a particular sampling rate. FullOpen will fail if you specify a sampling rate that is unacceptable for the ADS1210. Also, increasing the gain reduces the available turbo options since the product of gain and turbo is fixed. For best performance, use the maximum turbo value (TurboLog=4).

Decimation maps into the decimation bits in the ADS1210 control register, sets the decimation value for the A/D converters and affects the native sampling rate. Use the SpsGainToTde helper function to compute the values of turbolog, decimation, and extradecimation corresponding to a particular sampling rate. FullOpen will fail if you specify an effective sampling rate that is unacceptable for the ADS1210.

ExtraDecimation sets the amount of decimation done by the device driver in addition to the native decimation done in the A/D converters themselves and affects the effective sampling rate. Use the SpsGainToTde helper function to compute the values of turbolog, decimation, and extradecimation corresponding to a particular sampling rate. FullOpen will fail if you specify an effective sampling rate that is unacceptable for the ADS1210.

Unused is not used at this time, but is included to allow for future improvements without causing a change in the function prototype.

If ADS1210 converters are successfully initialized, the ADS1210_CrValue parameter will be filled with the readback value from the ADS1210 control register. Besides checking the FullOpen return, you can also check the bits in this value to make sure all ADS1210 control register values were written out and read back successfully. Pass NULL to ignore this parameter.

Besides returning a device handle, FullOpen also fills the Error parameter with an error code indicating success or why it failed. See parxch.h for a list of the possible error codes. Pass NULL to ignore this parameter.

A typical calling sequence for FullOpen would be:

```
#include "parxch.h"

  DriverName = PARXCH_0;
    XchModel = PARXCH_MODEL_PAR4CH;
    PortMode = PARXCH_PORT_MODE_EPP;
          Df = PARXCH_DF_SIGNED;
         Sps = 50.0;
     GainLog = PARXCH_GAIN_1;

ParXchSpsGainToTde(
                    Sps,
                    GainLog
                    &TurboLog,
                    &Decimation,
                    &ExtraDecimation
                    );

handle = ParXchFullOpen(
                    DriverName,
                    XchModel,
                    PortMode,
                    Df,
                    GainLog,
                    TurboLog,
                    Decimation,
                    ExtraDecimation,
                    NULL,
                    NULL
             );
if ( handle == BAD_DEVHANDLE )
       printf( "error, unable to initialize");
```

ADVANCED FUNCTIONS – SPS HELPERS

```
C usage:
       #include "parxch.h"

       void ParXchSpsGainToTde(
                           double Sps,
                           int GainLog,

                           int *TurboLog,
                           int *Decimation,
                           int *ExtraDecimation
                           );

       void ParXchTdeToSpsGain(
                           int TurboLog,
                           int Decimation,
                           int ExtraDecimation,

                           double *Sps,
                           int *GainLog,
                           );
```

The SpsGainToTde and TdeToSpsGain functions convert between the related parameters of sampling rate, gain, turbo rate, native decimation, and extra decimation. For a description of the different parameters, please refer to the Advanced Functions – FullOpen section above.

A typical calling sequence for SpsGainToTde and TdeToSpsGain would be:

```
       #include "parxch.h"

       Sps = 200.0;
       ParXchSpsGainToTde( Sps, PARXCH_GAIN_1, &tl, &dec, &exdec );
       ParXchTdeToSpsGain( tl, dec, exdec, &ActualSps, &ActualGL );

       printf( "Requested Sps = %f\n", Sps );
       printf( "   Actual Sps = %f\n", ActualSps );
```

LABVIEW SUPPORT

Support for National Instruments LabView is provided in the LabView LLB VI library parxch.llb in the labview directory. This LLB contains 3 application and 20 utility VIs. The application VIs include complete data acquisition programs that rely on the underlying utility VIs which are wrappers around the corresponding core parxch.dll DLL functions supplying the low level support for all SR PARxCH software.

To use the PARxCH LLB library, put \sr\parxch\lib on your path so Windows can find the core DLL at run time. You can then access the VIs by using the File>Open menu option to open \sr\parxch\labview\parxch.llb. If it is 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 parxch.llb into the user.lib subdirectory of your LabView directory. For further details, please refer to file \sr\parxch\labview\readme.txt.

The PARxCH 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 with their block diagrams included so you can modify them as needed to fit your application.

PARxCH LED Demo

This VI is so simple, it does not even start acquisition. It just initializes the PARxCH and allows you to toggle the on board yellow LED to visually verify that initialization was successful.

To run the demo, use the LabView Operating tool to set the Port Mode switch and the Port Name value as appropriate for your computer. Then press the Run Arrow on the toolbar. If PARxCH 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.

PARxCH Meter Demo

This VI samples the PARxCH at a slow rate and displays the results as digital readouts. After the front panel controls have been set, the PARxCH is initialized and acquisition is started. Ready is called to determine when a new sample for each channel is available. Then Read

Data is called to read them in. Since the desired sampling rates are slower than the minimum native rate that can be assigned to the ADS1210 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 approximate default values, but you will require different calibration values appropriate for each A/D chip before the volts display is accurate.

To run this demo, press the Run Arrow on the toolbar to enable the Driver Name control, the Xch Model control, the Port Mode switch, the Units switch, and the Sample Period dial. Once these controls are set 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.

PARxCH Scope Demo

The scope demo samples the PARxCH 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 simp.exe and scope.exe.

After the front panel controls have been set, the PARxCH is initialized, the header file is written, and acquisition is started. Because the sampling rate for this demo VI is much faster than for PARxCH 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, Ready is called to determine when one sample is available and Read Data is called repeatedly to read in a buffer worth of samples which are written out and displayed on the multi-plot graph. To reduce flashing as the graph is updated, check the "Use smooth updates" box from the Edit>Preferences>Front Panel dialog.

To run this demo, press the Run Arrow on the toolbar to enable the Driver Name control, the Xch Model control, the Port Mode switch, and the Sample Rate dial. Once these controls are set 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.

PARxCH Utility VIs

The following table lists the utility VIs in parxch.llb and their arguments. To allow complete flexibility in controlling the flow of execution, dummy inputs and/or outputs are provided for any function not having at least one input and output. PARxCH Start is an example.

Function	In/Out	Type	Description	Range
PARxCH Open	input	char	Driver name	SrParXch0, default
	input	int32	Xch Model	PAR1CH, PAR4CH,etc
	input	int32	Port mode	0 = BPP, 1 = EPP, 2 = ECP/BPP, 3 = ECP/EPP
	input	double	Sampling rate	2.44 to 15625, native
	output	int32	Device handle	BAD_DEVHANDLE = fail
	output	int32	ADS1210 CrValue	see BB spec sheet
	output	int32	Error code	0 to 21, see parxch.h
	output	double	True sample rate	
PARxCH Close	input	uint32	Device handle	from Open
	output	int32	Close result	1 = ok, 0 = fail
PARxCH Start	input	uint32	Device handle	from Open
	output	Int32	flow_out	Dummy
PARxCH Stop	input	uint32	Device handle	from Open
	output	int32	flow_out	Dummy
PARxCH Ready	input	uint32	Device handle	from Open
	output	int32	Ready result	1 = ready, 0 = not
PARxCH Overflow	input	uint32	Device handle	from Open
	output	int32	Overflow result	1 = overflow, 0 = not
PARxCH Read Data	input	uint32	Device handle	from Open
	input	int32*	Data values_in	
	input	uint32	Nvalues	max number to read
	input	int32	Error in	
	output	int32	Read Data result	# values read
	output	int32*	Data values	array of int32 values

	output	int32	Error	0 to 21, see parxch.h

PARxCH User Led	input	uint32	Device handle	from Open
	input	int32	User Led	0 or 1
	output	int32	flow_out	Dummy

PARxCH User IO Wr	input	uint32	Device handle	from Open
	input	int32	Value	0x0 to 0xF
	output	int32	flow_out	Dummy

PARxCH User IO Rd	input	uint32	Device handle	from Open
	input	int32	Value in	
	output	int32	Value	0x0 to 0xF

PARxCH Interrupt Enable	input	uint32	Device handle	from Open
	output	int32	flow_out	Dummy

PARxCH Interrupt Disable	input	uint32	Device handle	from Open
	output	int32	flow_out	Dummy

PARxCH Full Open	input	char	Driver Name	SrParXch0 default
	input	int32	Xch Model	PAR4CH, PAR8CH, etc
	input	int32	Port mode	0 = BPP, 1 = EPP, 2 = ECP/BPP, 3 = ECP/EPP
	input	double	Sampling rate	2.44 to 15625, native
	input	int32	Data Format	0 = signed, 1 = offset
	input	int32	Gain log	0 to 4 (for 1 to 16)
	input	int32	Turbo log	0 to 4 (for 1 to 16)
	input	int32	Decimation	19 to 8000
	input	int32	ExtraDecimation	1 default
	output	uint32	Device handle	BAD_DEVHANDLE = fail
	output	int32	ADS1210 CrValue	See BB spec sheet
	output	int32	Error code	0 to 21, see parxch.h
	output	double	True sample rate	

PARxCH Sps Gain To Tde	input	double	Samples / sec	2.44 to 15625, native
	input	int32	Gain log	0 to 4
	input	int32	Turbo log in	
	input	int32	Decimation in	
	input	int32	ExtraDecimation in	
	output	int32	Turbo log	0 to 4
	output	int32	Decimation	19 to 8000
	output	int32	ExtraDecimation	1 default

PARxCH Tde To Sps Gain	input	int32	Turbo log	0 to 4
	input	int32	Decimation	19 to 8000
	input	int32	ExtraDecimation	1 default
	input	double	Samples / sec in	
	input	int32	Gain log in	
	output	double	Samples / sec	2.44 to 15625, native
	output	int32	Gain log	0 to 4

PARxCH Voltage Good	input	uint32	Device handle	from Open
	output	int32	Voltage result	1 = good, 0 = bad

PARxCH Get Rev	input	int32	Rev in	
	output	int32	Rev result	201 means DLL rev 2.01

PARxCH Fifo Ready Size	input	uint32	Nvalues in	
	output	uint32	Nvalues	Currently always 1 * 4

PARxCH Write Header File	input	int32	Data format	0 = signed, 1 = offset
	input	int32	Word size	4
	input	int32	Record pts	Nvalues
	input	int32	Channels	1, 4, 8
	input	path	File name	Scope.hdr
	input	string	Id	SR PARxCH
	input	error	Error in	
	input	int32*	Channel array	0,1,2,3,4,5,6,7 default
	input	string*	Channel title	[1,4,8], 12 char

	input	float	Sample rate	2.44 to 15625, native
	input	float	Gain	0 => 2**0 to 4 => 2**4
	input	float	Filter coeff	0
	input	float	Filter scale	1
	output	error	Error out	

PARxCH 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

POWER SUPPLY REQUIREMENTS:

The PARxCH uses +5vdc to power the analog and digital circuitry. The on board voltages are generated by linear regulators from the off board power supply, usually a wall transformer, to provide quiet operation. The off board power supply must allow enough voltage headroom for the on board regulators to operate. Off board power of at least 10 volts is recommended for reliable operation.

The system requires approximately 120ma of current. This figure is approximate because the actual current draw depends on the sampling rate. Higher sampling rates result in more current draw. We have found 120ma to be a typical maximum figure.

A 9vdc 500ma wall transformer is supplied with the system. When loaded at 120ma, it will provide approximately 12vdc of output power, more than enough to adequately power the PARxCH.

Other off board power supplies are acceptable. AC power will be rectified by the on board circuitry and 9vac wall or chassis mount transformers are perfectly fine with no degradation in system noise or resolution.

MAXIMUM ANALOG INPUT VOLTAGE RANGE:

The absolute maximum analog input voltage range of the PARxCH is +/-10v. This is mapped by the on board circuitry to a 0 to 5v range that is compatible with the ADS1210 A/D converters. Although the PARxCH analog inputs are protected by resistive current limiting, do not exceed the recommended +/-10v input range, or you may damage the system. Static discharges near the analog input pins should also be avoided.

All of the PARxCH analog inputs are differential. This means that the absolute difference between the + and - inputs on any given channel must be less than 20 volts. In addition, for the system to work correctly, ***you must*** establish a common analog ground that is connected to one of the ground pins on the analog input D-shell connector. The inputs are differential only up to the point where the A/D converters are pinned. When analog ground floats, it can easily drift to absolute voltages that exceed the A/Ds common mode range even though the absolute difference is still within range. Connecting an analog ground reference avoids this.

APPENDIX B: DATA FORMATS

When programming the PARxCH, you can use the Data Format parameter, df, to choose the format of the acquired data. The two choices are signed and offset. The Open function defaults to the signed format.

When signed format is selected, the 24 bit data is sign extended and treated as signed 32 bit values. When offset format is selected, the data is treated as unsigned two's complement. Selected values for both formats are shown in the following table:

	Offset	**Signed**
Lowest value	`0x00000000`	`0xFF800000`
Mid-level, minus one	`0x007FFFFF`	`0xFFFFFFFF`
Mid-level, zero	`0x00800000`	`0x00000000`
Highest value	`0x00FFFFFF`	`0x007FFFFF`

APPENDIX C: OUTPUT FILE FORMATS

The acquisition programs simp and scope can output the acquired data to files in one of two formats, .out files and .dat files. Additional utility programs, located in the convert directory, are available for converting the binary .out and .dat data files into other formats such as ASCII text or IASPEI SUDS.

The .out file format is very simple and contains only the multiplexed data values stored as 32-bit integers. The data is organized exactly as described for the ReadData function. The first integer contains the data for point 0 of channel 0. The second has the data for point 0 of channel 1. The Nth integer has the data for point 0 of channel N, while integer N+1 has the data for point 1 of channel 0. And so on. All data is written out in the standard little-endian PC byte order, where the low byte is written first.

When .out files are requested, the simp and scope acquisition programs also write a single, separate header file with the .hdr extension. This file contains information such as sampling rate and number of channels that apply to all the .out data files. The format of this header file can be seen by looking at the HDR_data structure in the simp.h file in the simple directory.

The .dat file format is more complicated as each file contains a 4096 byte header followed by data records, possibly of many different types. The header begins with SrDatHdrLayout structure defined and described in the srdat.h file in the include directory. The remaining bytes of the header are padded with zero. Each data record is composed of a short 128 byte record tag structure identifying the format and size of the following data and then the data itself. The record tag structure and the currently defined data formats are described in the srdat.h file. The .dat file header and data records are written out in standard little-endian PC byte order.

Which data format is best? That depends on what you need. Most users will be happiest with the .dat format since that keeps all the header information tightly associated with the data and allows saving GPS PPS and serial data in addition to the analog and digital data. But, these extra features do have a cost in terms of speed and complexity. So, users with high sampling rates that only need analog and digital data may prefer using the simpler .out file format.

INDEX

SR PAR4CH BLOCK DIAGRAM OVERVIEW

PARALLEL PORT INTERFACE

SCHEMATIC PAGES 2-5
DB25 MALE CONNECTOR
IEEE 1284 BPP/EPP/ECP MODES

(100)

DIGITAL CONTROLLER

SCHEMATIC PAGES 6-8
CPLD/FPGA CUSTOM PROCESSOR
LOW POWER 3.3 volt OPERATION
2 Mb DRAM FIFO

(200)

DIGITAL IO

SCHEMATIC PAGES 9-11
DB15 MALE CONNECTOR
FOUR INPUTS AND FOUR OUTPUTS
TTL/CMOS 5 volt SIGNAL LEVELS

(300)

ANALOG INPUTS

SCHEMATIC PAGE 12-22
DB15 MALE CONNECTOR
FOUR INPUT CHANNELS
+/-10 volt DIFFERENTIAL INPUTS
24 BIT A/D CONVERTER PER CHANNEL

(400)

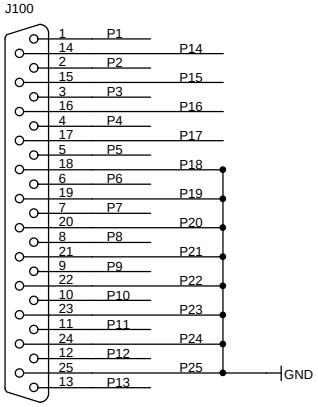
POWER SUPPLY

SCHEMATIC PAGES 23-31
9 TO 24 VAC/VDC ACCEPTABLE INPUTS
2.1MM WALL TRANSFORMER AND 4 PIN MOLEX CONNECTORS
ON BOARD REGULATED TO +5 AND +3.3

(900)

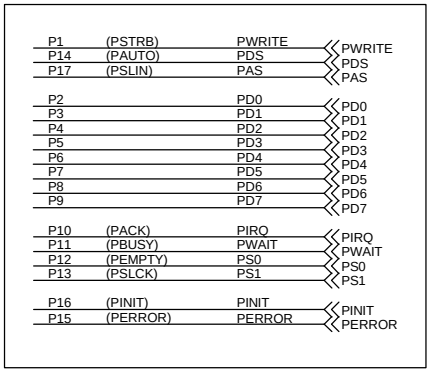
Title: OVERVIEW		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 1 of 31	

PC PARALLEL PORT CONNECTOR



CONNECTOR DB25

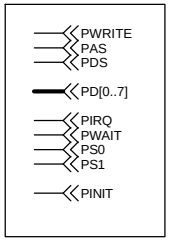
MALE DB25

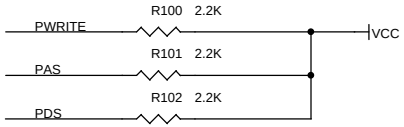


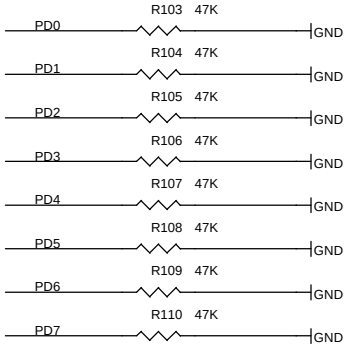
PIN TO SIGNAL NAME MAPPING

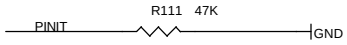
Title	DIG PAR CONNECTOR	
Size A	Document Number PAR4CH.DSN, (c) Symmetric Research, 2003	Rev E
Date: Tuesday, April 01, 2003	Sheet 2 of 31	

PC PARALLEL PORT PULL UP/DOWN

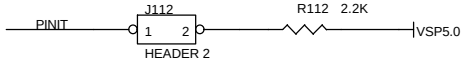






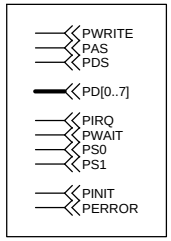


OPTIONAL PULLUP ON PINIT

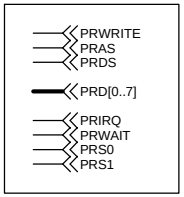


Title: DIG PAR PULL UP/DOWN		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 3 of 31	

PC PARALLEL PORT EDGE CONTROL AND CURRENT LIMITING



C120	NP [470pF]	GND	PWRITE	PRWRITE	R120 1K
C121	NP [470pF]	GND	PAS	PRAS	R121 1K
C122	NP [470pF]	GND	PDS	PRDS	R122 1K
C123	NP [2200pF]	GND	PD0	PRD0	R123 220
C124	NP [2200pF]	GND	PD1	PRD1	R124 220
C125	NP [2200pF]	GND	PD2	PRD2	R125 220
C126	NP [2200pF]	GND	PD3	PRD3	R126 220
C127	NP [2200pF]	GND	PD4	PRD4	R127 220
C128	NP [2200pF]	GND	PD5	PRD5	R128 220
C129	NP [2200pF]	GND	PD6	PRD6	R129 220
C130	NP [2200pF]	GND	PD7	PRD7	R130 220
C131	NP [470pF]	GND	PIRQ	PRIRQ	R131 47
C132	NP [470pF]	GND	PWAIT	PRWAIT	R132 47
C133	NP [470pF]	GND	PS0	PRS0	R133 47
C134	NP [470pF]	GND	PS1	PRS1	R134 47



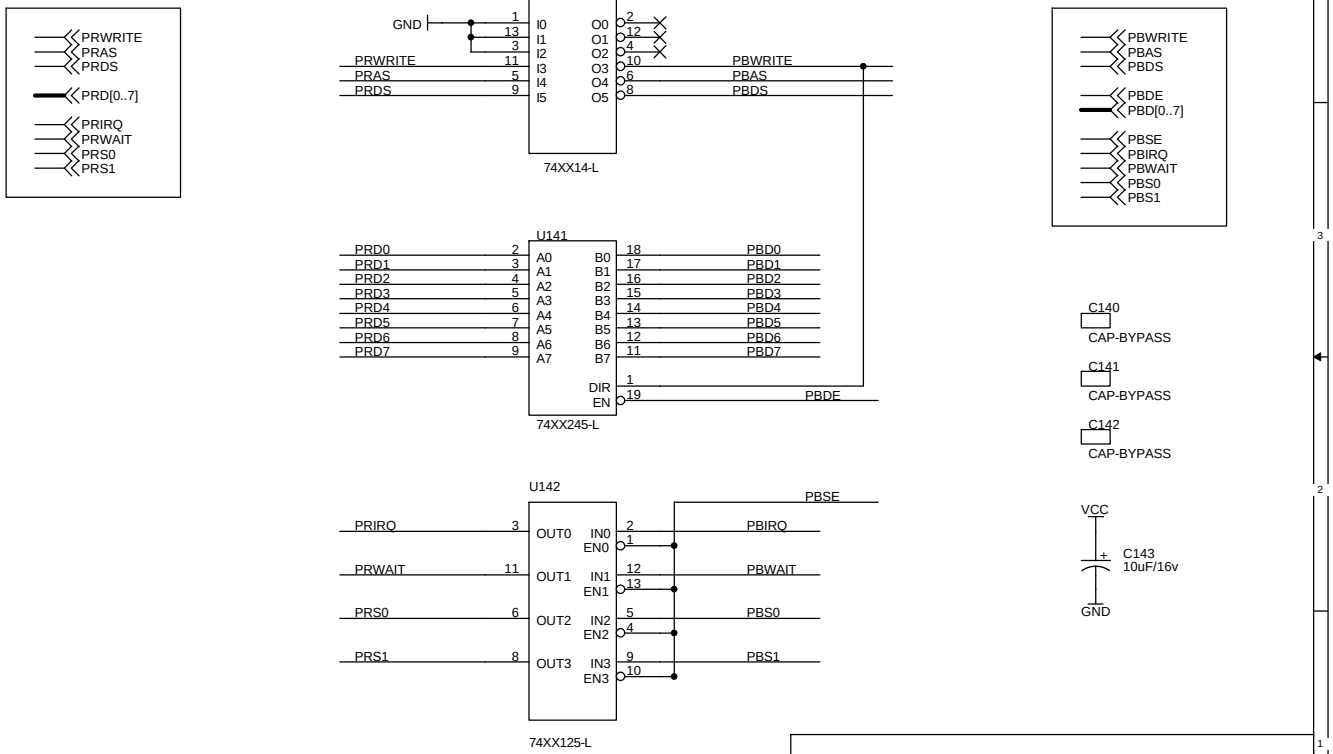
C135 NP [0.1uF] GND PINIT PRINIT R135 1K

PERROR PRERROR

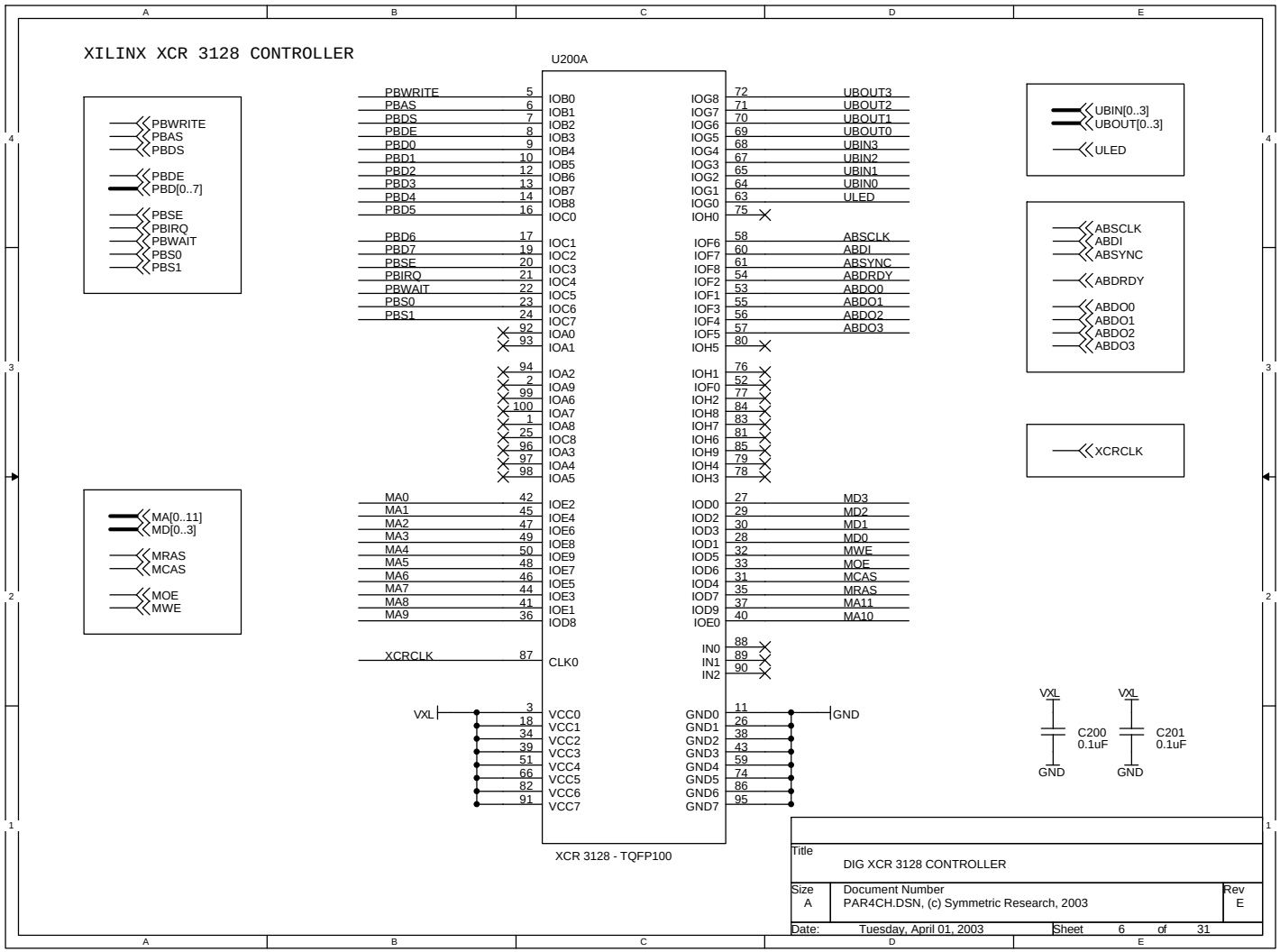
SEE PAGE POWER SUPPLY 5 FOR PERROR CONDITIONING

Title	DIG PAR FILTERS	
Size A	Document Number PAR4CH.DSN, (c) Symmetric Research, 2003	Rev E
Date: Tuesday, April 01, 2003	Sheet 4 of 31	

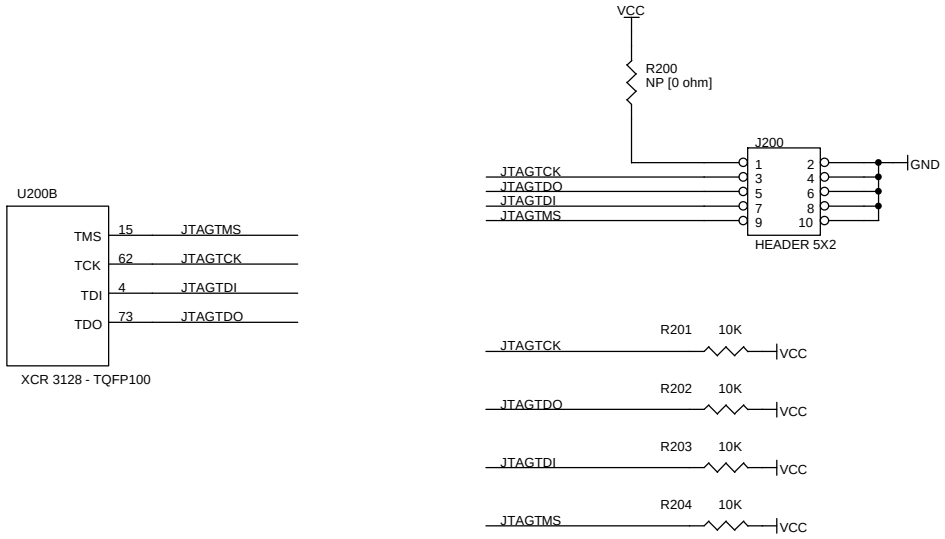
PC PARALLEL PORT BUFFERING



Title	DIG PAR BUFFERS	
Size A	Document Number PAR4CH.DSN, (c) Symmetric Research, 2003	Rev E
Date: Tuesday, April 01, 2003	Sheet 5 of 31	


XILINX XCR 3128 CONTROLLER
U200A
PBWRITE
PBAS
PBDS
PBDE
PBD[0..7]
PBSE
PBIRQ
PBWAIT
PBS0
PBS1
MA[0..11]
MD[0..3]
MRAS
MCAS
MOE
MWE
UBIN[0..3]
UBOUT[0..3]
ULED
ABSCLK
ABDI
ABSYNC
ABDRDY
ABDO0
ABDO1
ABDO2
ABDO3
XCRCLK
IOB0 IOB1 IOB2 IOB3 IOB4 IOB5 IOB6 IOB7 IOB8 IOC0
IOC1 IOC2 IOC3 IOC4 IOC5 IOC6 IOC7 IOA0 IOA1
IOA2 IOA9 IOA6 IOA7 IOA8 IOC8 IOA3 IOA4 IOA5
IOE2 IOE4 IOE6 IOE8 IOE9 IOE7 IOE5 IOE3 IOE1 IOD8
CLK0
VCC0 VCC1 VCC2 VCC3 VCC4 VCC5 VCC6 VCC7
IOG8 IOG7 IOG6 IOG5 IOG4 IOG3 IOG2 IOG1 IOG0 IOH0
IOF6 IOF7 IOF8 IOF2 IOF1 IOF3 IOF4 IOF5 IOH5
IOH1 IOF0 IOH2 IOH8 IOH7 IOH6 IOH9 IOH4 IOH3
IOD0 IOD2 IOD3 IOD1 IOD5 IOD6 IOD4 IOD7 IOD9 IOE0
IN0 IN1 IN2
GND0 GND1 GND2 GND3 GND4 GND5 GND6 GND7
XCR 3128 - TQFP100
VXL
GND
C200 0.1uF
C201 0.1uF
Title
DIG XCR 3128 CONTROLLER
Size A
Document Number
PAR4CH.DSN, (c) Symmetric Research, 2003
Rev E
Date: Tuesday, April 01, 2003
Sheet 6 of 31

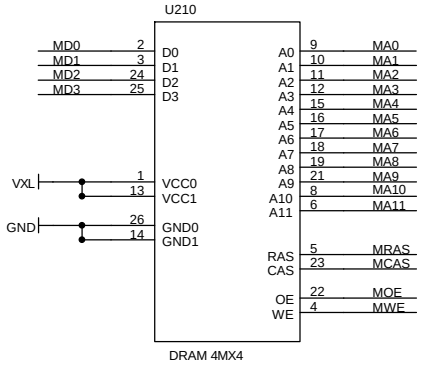
XILINX XCR 3128 JTAG ISP PROGRAMMING HEADER

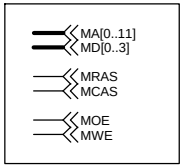


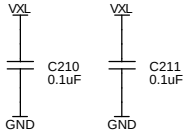
XCR 3128 PROGRAMMING HARDWARE INTERFACE ONLY

Title: DIG XCR 3128 PROGRAMMING		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 7 of 31	

DRAM FIFO/MEMORY BUFFER





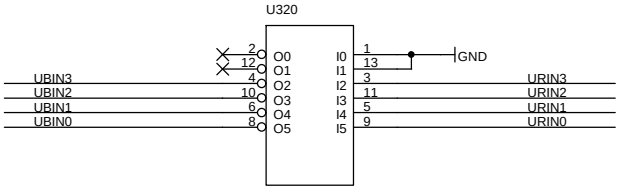


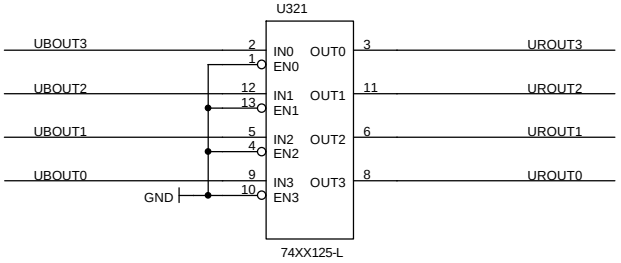
Title: DIG DRAM FIFO		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 8 of 31	

USER DIGITAL IO BUFFERS

UBIN[0..3]
UBOUT[0..3]

URIN[0..3]
UROUT[0..3]





C320
CAP-BYPASS

C321
CAP-BYPASS

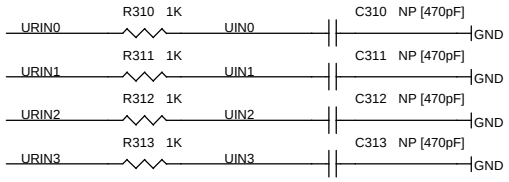
VCC
C322
10uF
GND

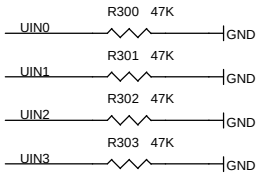
Title: DIG USER IO BUFFERS		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 9 of 31	

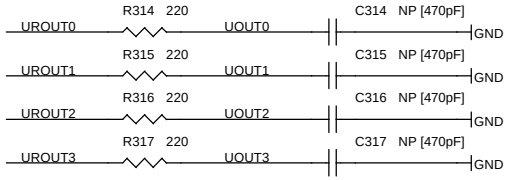
USER DIGITAL IO EDGE CONTROL AND CURRENT LIMITING

URIN[0..3]
UROUT[0..3]

UIN[0..3]
UOUT[0..3]





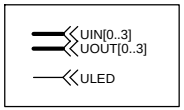


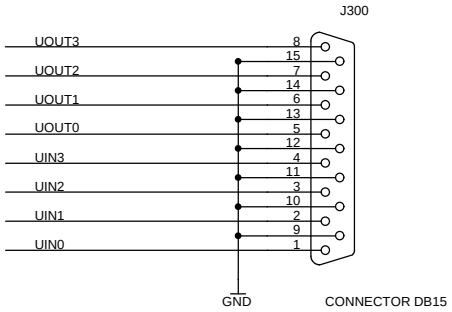
Title: DIG USER IO PULL UP/DOWN		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 10 of 31	

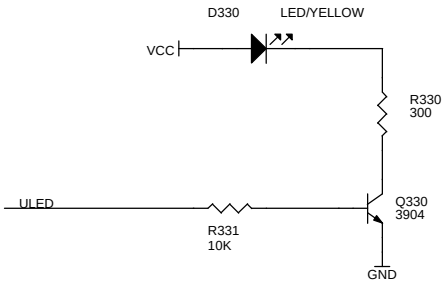
USER DIGITAL IO CONNECTOR AND LED

MALE DB15 CONNECTOR

(RIGHT HAND SIDE OF FRONT PANEL)



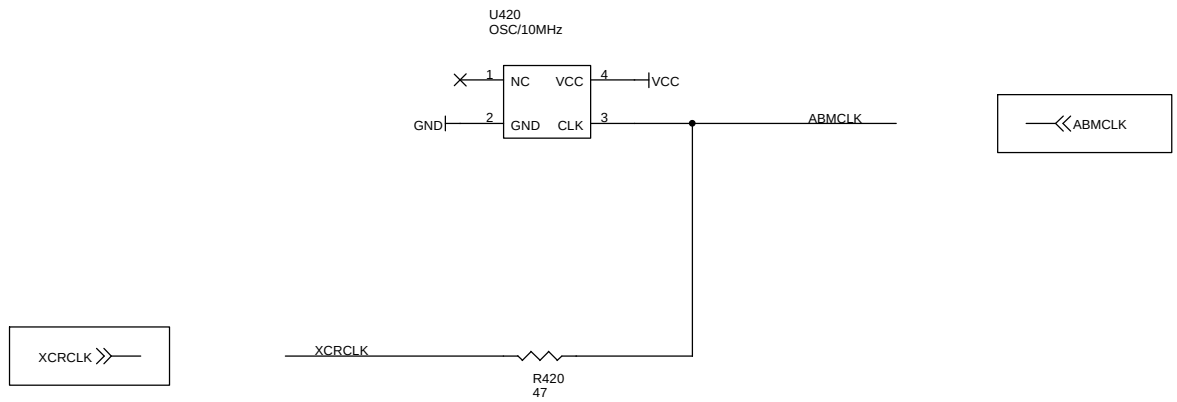


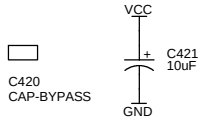


YELLOW FRONT PANEL LED

Title: DIG USER IO CONNECTOR		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 11 of 31	

MASTER CLOCK FOR A/D CONVERTERS AND XCR 3128 DIGITAL CONTROLLER





CLOCK PLACED ON ANALOG PLANES FOR MINIMUM SIGMA DELTA JITTER

Title: ANA MASTER CLOCK		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 12 of 31	

A/D CONVERTER DIGITAL BUFFERING

(INPUTS TO A/D)

XCR 3128

ABSCLK
ABDI
ABSYNC
ABMCLK

ABDRDY

ABDO0
ABDO1
ABDO2
ABDO3

ABSCLK	R421 47	ASCLK
ABDI	R422 1K	ADI
ABMCLK	R423 47	AMCLK
ABSYNC	R424 47	ASYNC

A/D CONVERTERS

ASCLK
ADI
ASYNC
AMCLK

ADRDY

ADO0
ADO1
ADO2
ADO3

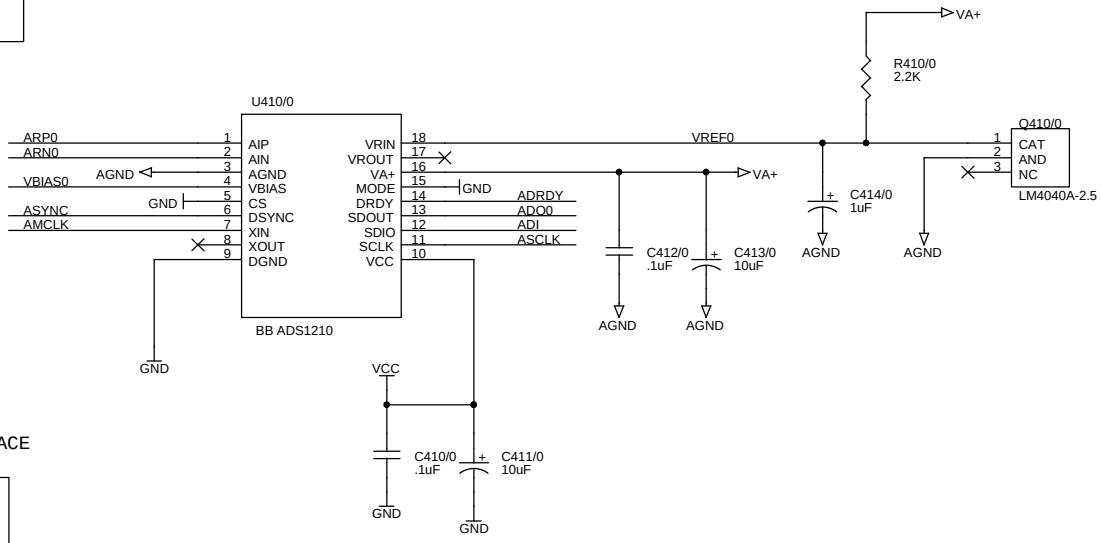
(OUTPUTS FROM A/D)

ABDRDY	ADRDY
ABDO0	ADO0
ABDO1	ADO1
ABDO2	ADO2
ABDO3	ADO3

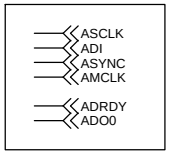
Title: ANA DIG BUFFERS		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 13 of 31	

A/D CONVERTER (4 IDENTICAL CHANNELS)

ANALOG INPUT FROM SIGNAL CONDITIONING



DIGITAL INTERFACE



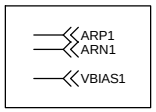
MODE WIRED LOW, 1 = MASTER, 0 = SLAVE

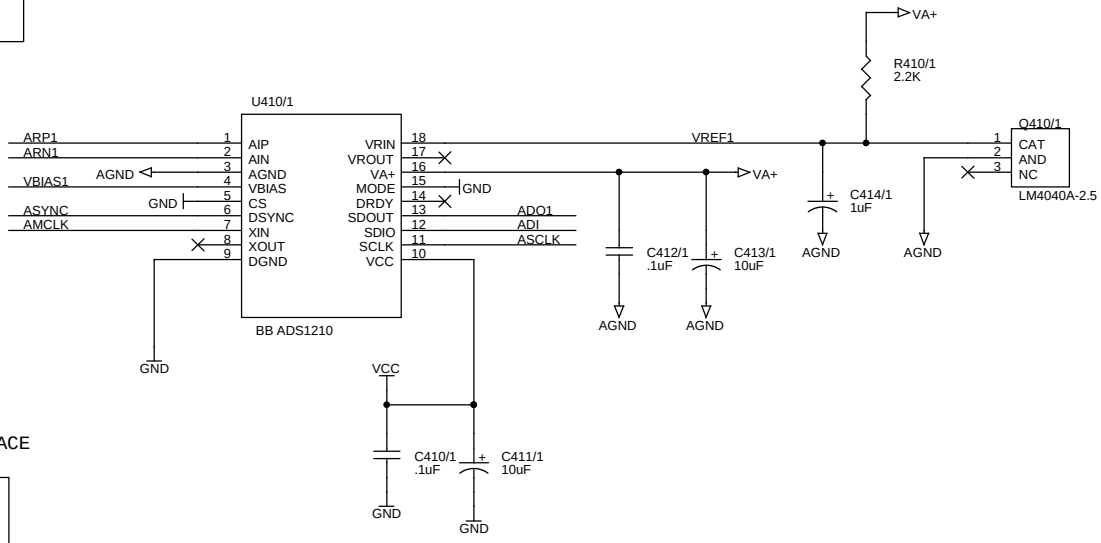
DRDY ONLY TAKEN FROM CHANNEL 0

Title: ANA A/D CONV 0		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 14 of 31	

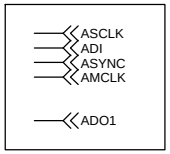
A/D CONVERTER (4 IDENTICAL CHANNELS)

ANALOG INPUT FROM SIGNAL CONDITIONING





DIGITAL INTERFACE



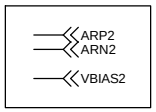
MODE WIRED LOW, 1 = MASTER, 0 = SLAVE

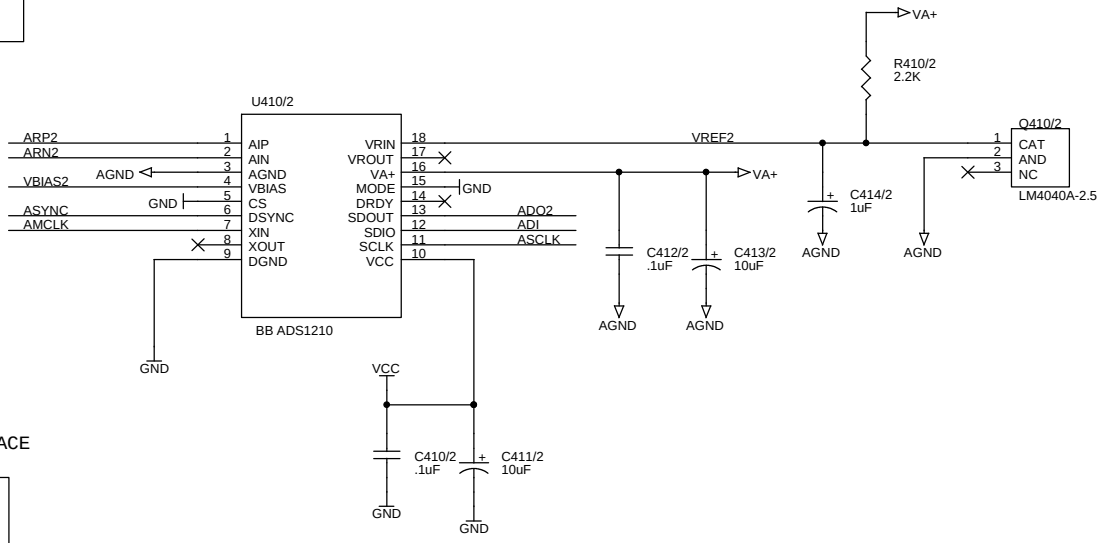
DRDY ONLY TAKEN FROM CHANNEL 0

Title: ANA A/D CONV 1		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 15 of 31	

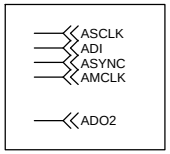
A/D CONVERTER (4 IDENTICAL CHANNELS)

ANALOG INPUT FROM SIGNAL CONDITIONING





DIGITAL INTERFACE



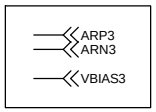
MODE WIRED LOW, 1 = MASTER, 0 = SLAVE

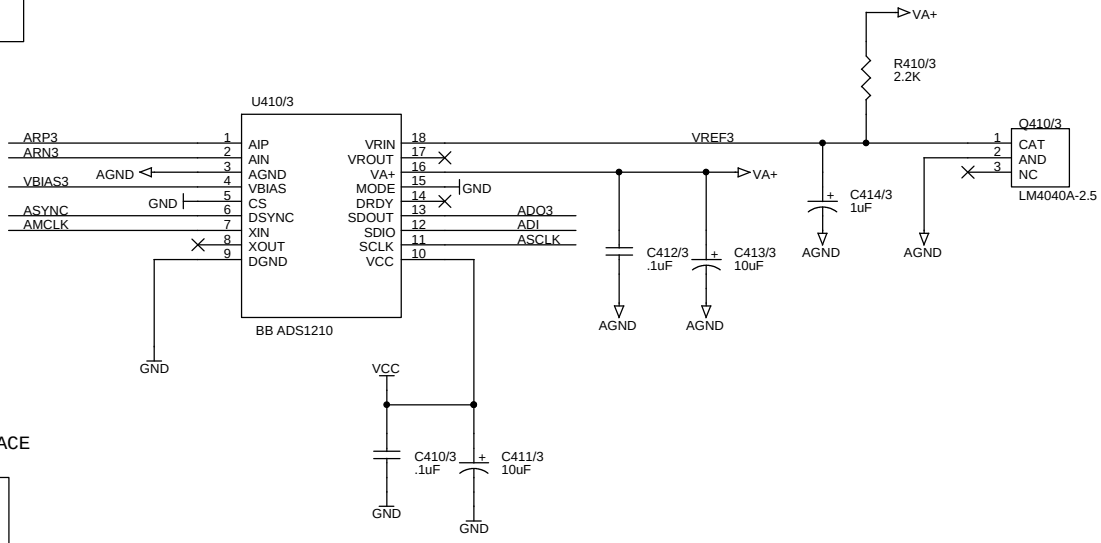
DRDY ONLY TAKEN FROM CHANNEL 0

Title: ANA A/D CONV 2		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 16 of 31	

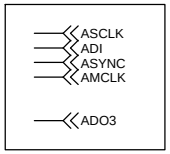
A/D CONVERTER (4 IDENTICAL CHANNELS)

ANALOG INPUT FROM SIGNAL CONDITIONING





DIGITAL INTERFACE



MODE WIRED LOW, 1 = MASTER, 0 = SLAVE

DRDY ONLY TAKEN FROM CHANNEL 0

Title: ANA A/D CONV 3		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 17 of 31	

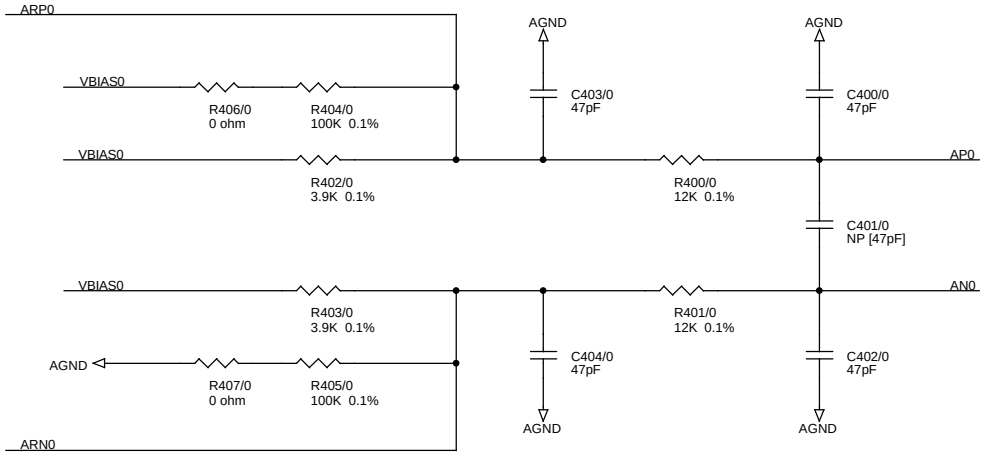
ANALOG INPUT SIGNAL CONDITIONING (4 IDENTICAL CHANNELS)

TO A/D CONVERTER

ARP0
ARN0
VBIAS0

FROM ANALOG INPUT CONNECTOR

AP0
AN0



Title: ANA SIG COND 0		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 18 of 31	

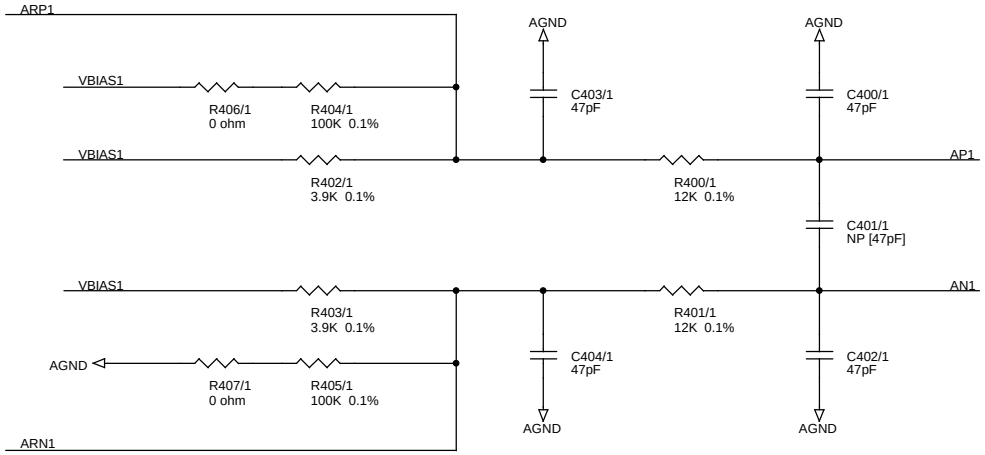
ANALOG INPUT SIGNAL CONDITIONING (4 IDENTICAL CHANNELS)

TO A/D CONVERTER

ARP1
ARN1
VBIAS1

FROM ANALOG INPUT CONNECTOR

AP1
AN1



Title: ANA SIG COND 1		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 19 of 31	

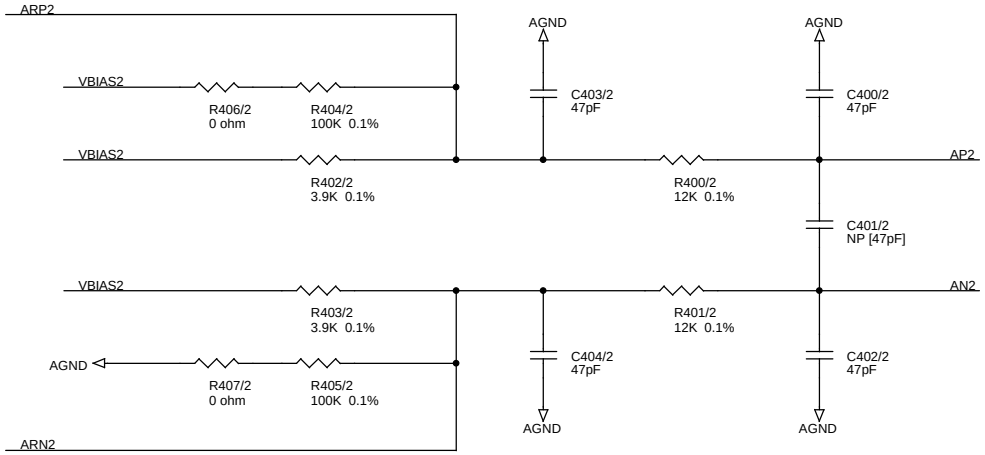
ANALOG INPUT SIGNAL CONDITIONING (4 IDENTICAL CHANNELS)

TO A/D CONVERTER

ARP2
ARN2
VBIAS2

FROM ANALOG INPUT CONNECTOR

AP2
AN2



Title: ANA SIG COND 2		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 20 of 31	

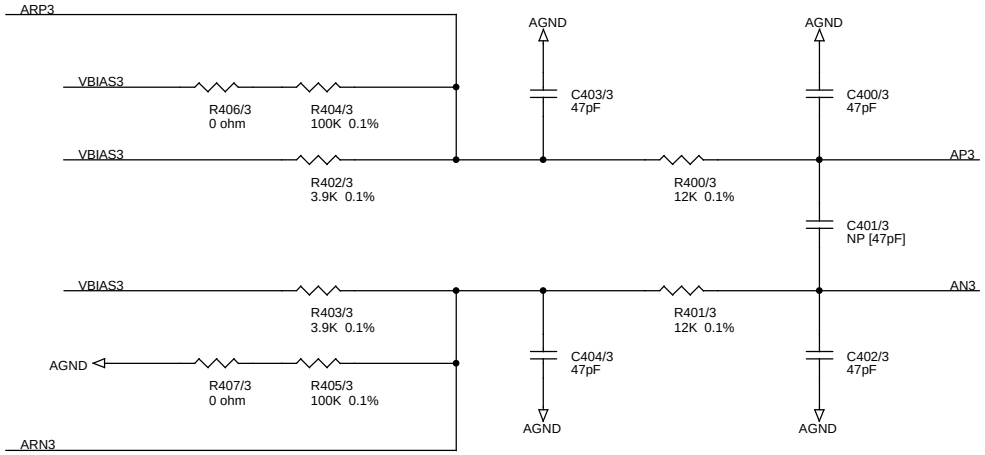
ANALOG INPUT SIGNAL CONDITIONING (4 IDENTICAL CHANNELS)

TO A/D CONVERTER

ARP3
ARN3
VBIAS3

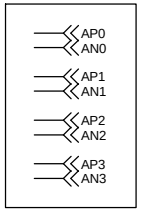
FROM ANALOG INPUT CONNECTOR

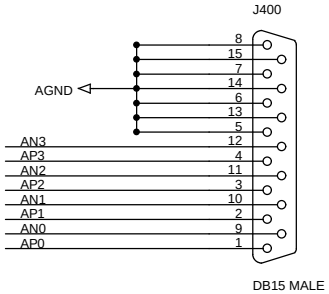
AP3
AN3



Title: ANA SIG COND 3		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 21 of 31	

ANALOG INPUT CONNECTOR





TO INPUT SIGNAL CONDITIONING

NOTES:

ALL INPUTS DIFFERENTIAL

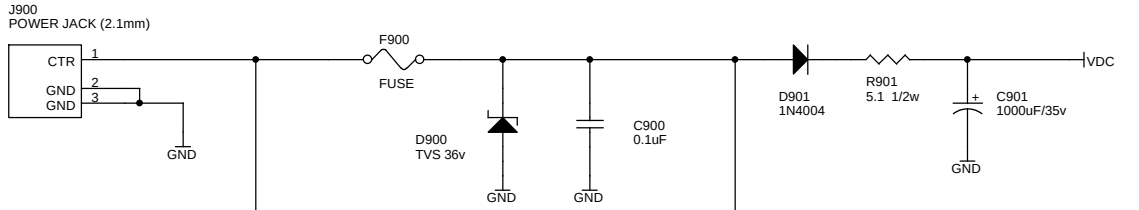
SEE USER MANUAL FOR TYPICAL CONNECTIONS, IMPROPER USAGE WILL LEAD TO SIGNAL CLIPPING

Title: ANA INPUT CONNECTOR		
Size: A	Document Number: PAR4CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 22 of 31	

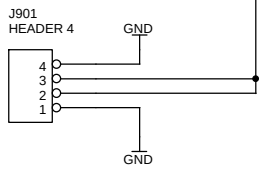
POWER SUPPLY INPUT CONNECTORS AND BASIC FILTERING

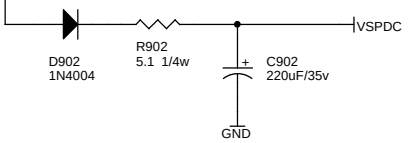
2.1MM WALL TRANS JACK

D900 = BIDIRECTIONAL P6KE36CA TVS



ALTERNATE HEADER POWER



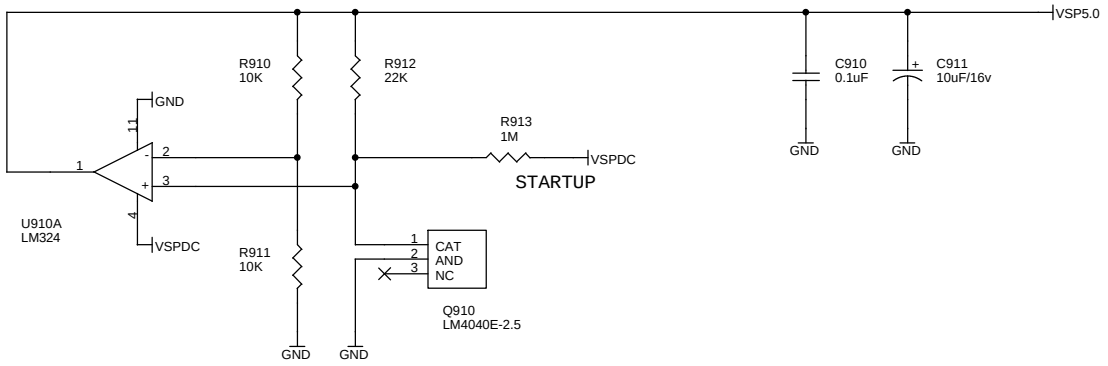


ACCEPTABLE OFF BOARD POWER SUPPLIES = 9 to 24 VAC or VDC

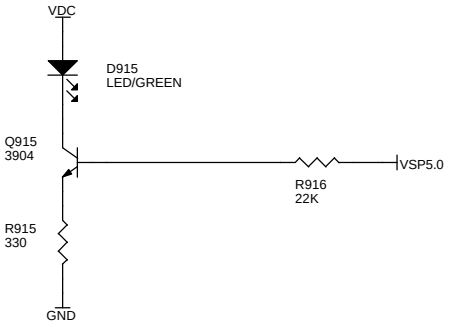
9 VDC @ 500ma PREFERRED

Title: PWR SUPPLY 0		
Size: A	Document Number: PAR1CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 23 of 31	

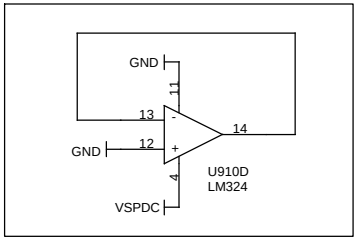
POWER SUPPLY REFERENCE AND SECONDARY 5v



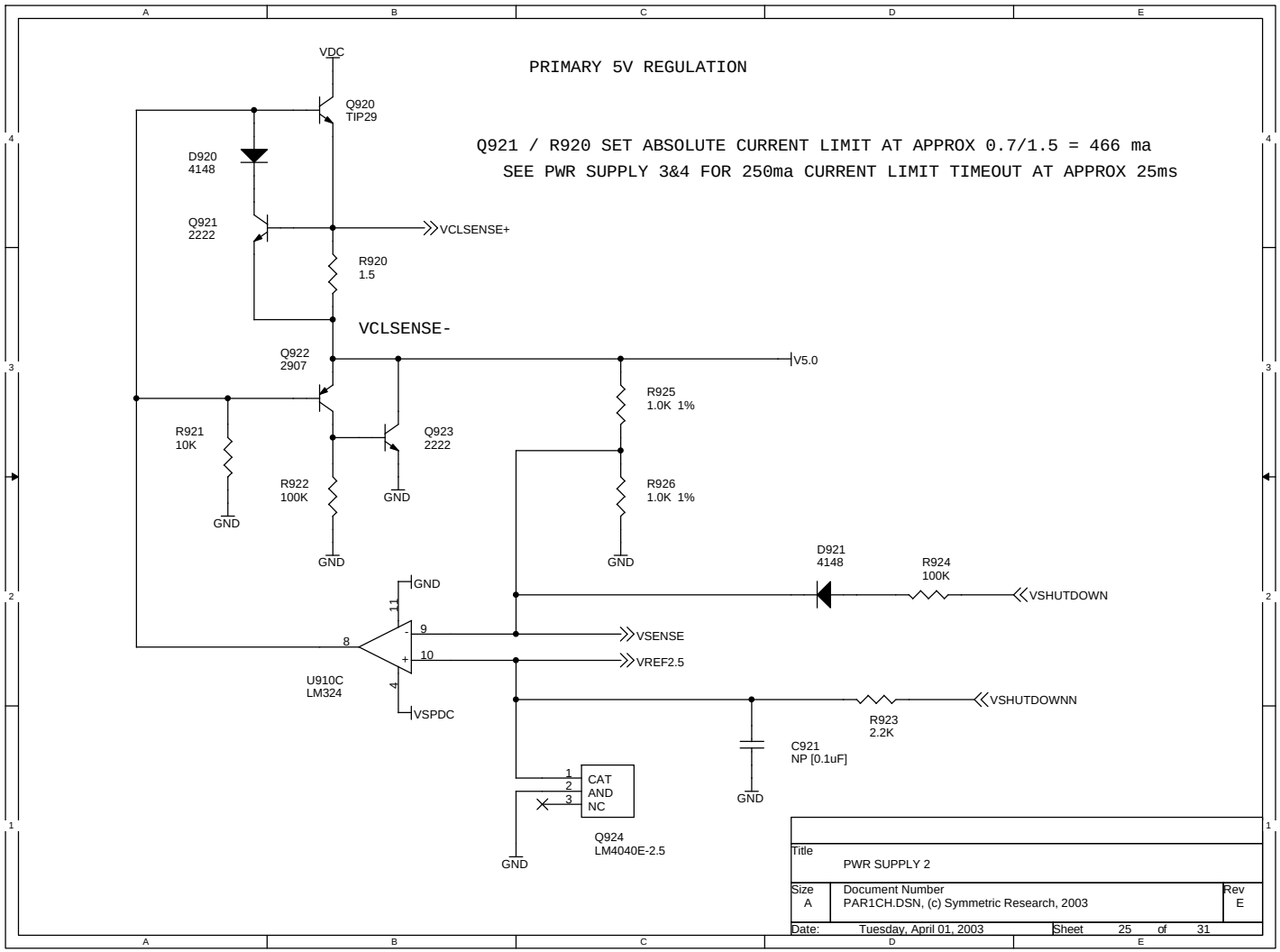
WALL TRANS LED



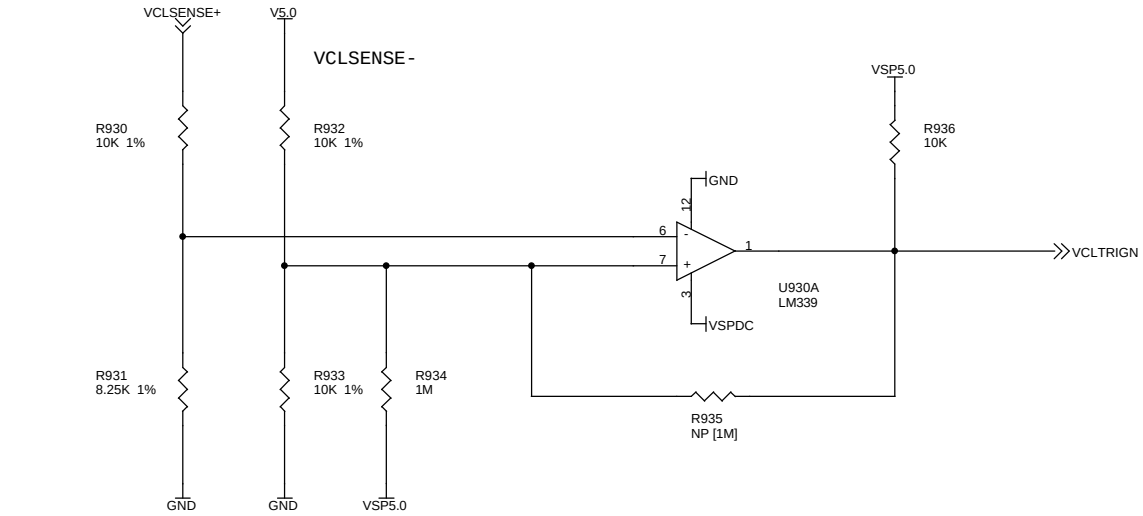
UNUSED OP AMP



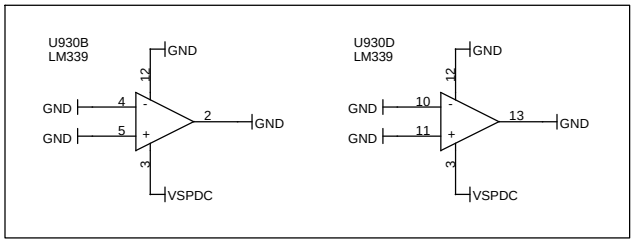
Title	PWR SUPPLY 1	
Size A	Document Number PAR1CH.DSN, (c) Symmetric Research, 2003	Rev E
Date: Tuesday, April 01, 2003	Sheet 24 of 31	


PRIMARY 5V REGULATION
Q921 / R920 SET ABSOLUTE CURRENT LIMIT AT APPROX 0.7/1.5 = 466 ma
SEE PWR SUPPLY 3&4 FOR 250ma CURRENT LIMIT TIMEOUT AT APPROX 25ms
VDC
Q920
TIP29
D920
4148
Q921
2222
VCLSENSE+
R920
1.5
VCLSENSE-
Q922
2907
V5.0
R925
1.0K 1%
R921
10K
Q923
2222
R922
100K
R926
1.0K 1%
GND
D921
4148
R924
100K
VSHUTDOWN
VSENSE
VREF2.5
VSHUTDOWNN
R923
2.2K
C921
NP [0.1uF]
U910C
LM324
VSPDC
CAT
AND
NC
Q924
LM4040E-2.5
Title
PWR SUPPLY 2
Size
A
Document Number
PAR1CH.DSN, (c) Symmetric Research, 2003
Rev
E
Date:
Tuesday, April 01, 2003
Sheet
25
of
31

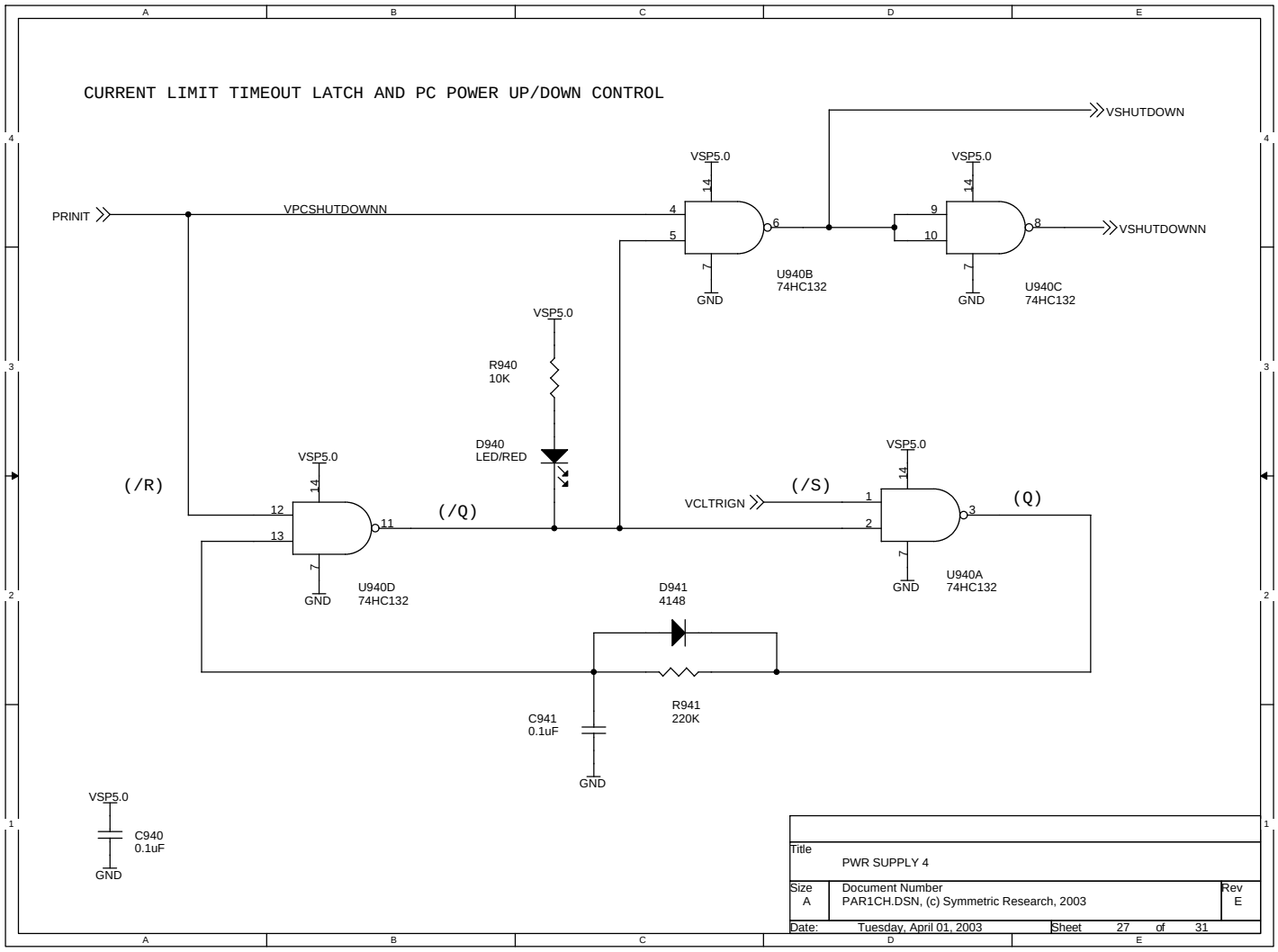
CURRENT LIMIT TIMEOUT COMPARATOR



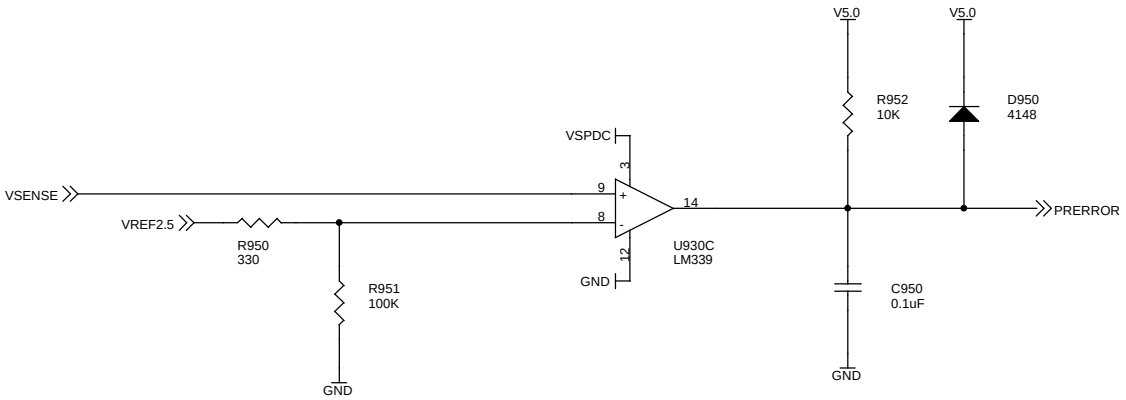
UNUSED COMPARATORS



Title	PWR SUPPLY 3	
Size A	Document Number PAR1CH.DSN, (c) Symmetric Research, 2003	Rev E
Date: Tuesday, April 01, 2003	Sheet 26 of 31	


CURRENT LIMIT TIMEOUT LATCH AND PC POWER UP/DOWN CONTROL
VSHUTDOWN
VSP5.0
VSP5.0
PRINIT
VPCSHUTDOWNN
U940B
74HC132
U940C
74HC132
VSHUTDOWNN
GND
GND
VSP5.0
R940
10K
D940
LED/RED
VSP5.0
VSP5.0
(/R)
(/Q)
(/S)
(Q)
VCLTRIGN
U940D
74HC132
U940A
74HC132
GND
GND
D941
4148
R941
220K
C941
0.1uF
GND
VSP5.0
C940
0.1uF
GND
Title
PWR SUPPLY 4
Size
A
Document Number
PAR1CH.DSN, (c) Symmetric Research, 2003
Rev
E
Date:
Tuesday, April 01, 2003
Sheet
27
of
31

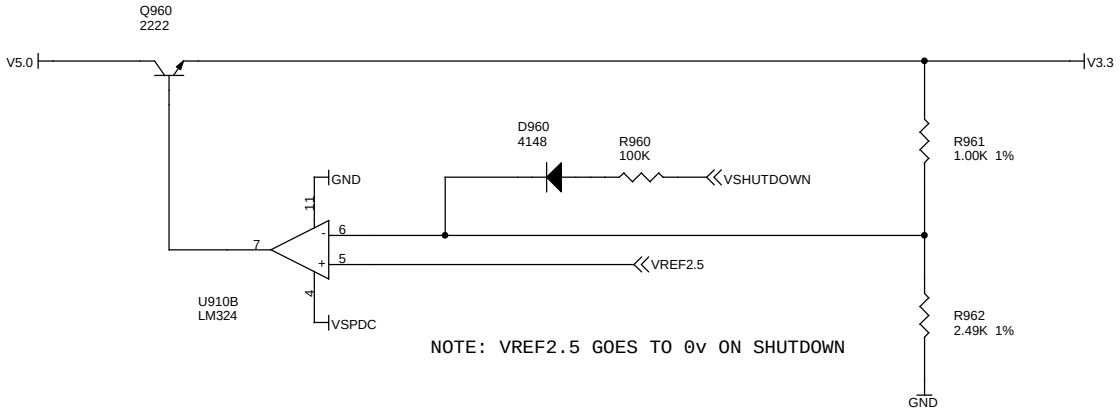
VOLTAGE GOOD/IN REGULATION STATUS



Title	PWR SUPPLY 5	
Size A	Document Number PAR1CH.DSN, (c) Symmetric Research, 2003	Rev E
Date: Tuesday, April 01, 2003	Sheet 28 of 31	

3V LOGIC SUPPLY

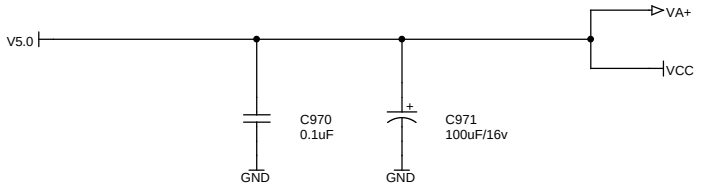
RATIO R960/R961 = 0.4, SELECTED FOR 3.5V POWER



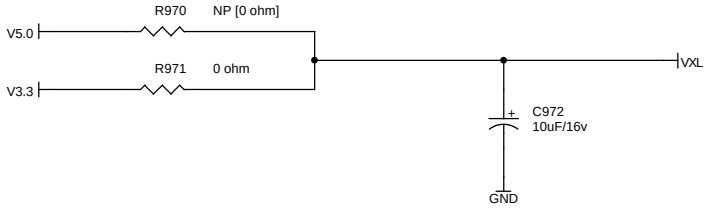
NOTE: VREF2.5 GOES TO 0v ON SHUTDOWN

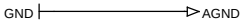
Title: PWR SUPPLY 6		
Size: A	Document Number: PAR1CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 29 of 31	

POWER DISTRIBUTION



OPTIONAL 5v SUPPORT FOR OLDER XCR5128 CPLD

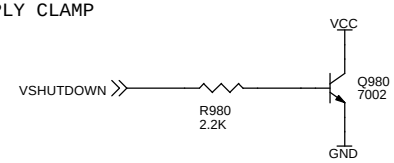




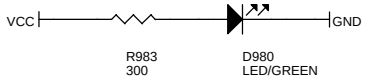
**** DO NOT CHANGE R970, R971 AS DELIVERED FROM FACTORY ****

Title: PWR SUPPLY 7		
Size: A	Document Number: PAR1CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 30 of 31	

HARD PC SHUTDOWN POWER SUPPLY CLAMP

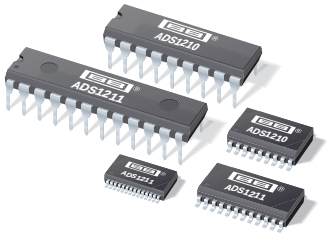


ON BOARD POWER INDICATOR



Title: PWR SUPPLY 8		
Size: A	Document Number: PAR1CH.DSN, (c) Symmetric Research, 2003	Rev: E
Date: Tuesday, April 01, 2003	Sheet 31 of 31	

BURR-BROWN®



ADS1210
ADS1211

24-Bit ANALOG-TO-DIGITAL CONVERTER

FEATURES

- DELTA-SIGMA A/D CONVERTER
- 23 BITS EFFECTIVE RESOLUTION AT 10Hz AND 20 BITS AT 1000Hz
- DIFFERENTIAL INPUTS
- PROGRAMMABLE GAIN AMPLIFIER
- FLEXIBLE SPI COMPATIBLE SSI INTERFACE WITH 2-WIRE MODE
- PROGRAMMABLE CUT-OFF FREQUENCY UP TO 15.6kHz
- INTERNAL/EXTERNAL REFERENCE
- ON CHIP SELF-CALIBRATION
- ADS1211 INCLUDES 4 CHANNEL MUX

APPLICATIONS

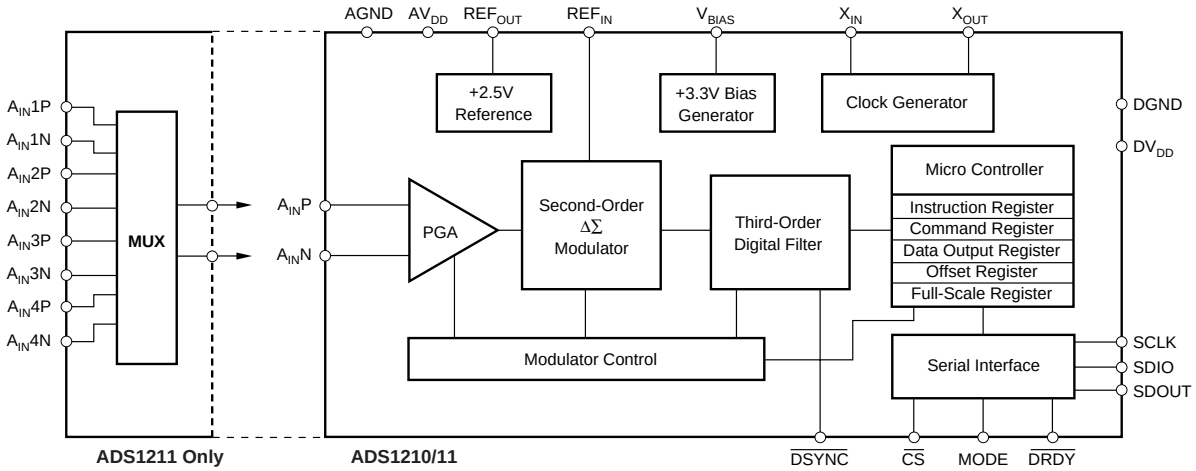
- INDUSTRIAL PROCESS CONTROL
- INSTRUMENTATION
- BLOOD ANALYSIS
- SMART TRANSMITTERS
- PORTABLE INSTRUMENTS
- WEIGH SCALES
- PRESSURE TRANSDUCERS

DESCRIPTION

The ADS1210 and ADS1211 are precision, wide dynamic range, delta-sigma analog-to-digital converters with 24-bit resolution operating from a single +5V supply. The differential inputs are ideal for direct connection to transducers or low level voltage signals. The delta-sigma architecture is used for wide dynamic range and to guarantee 22 bits of no missing code performance. An effective resolution of 23 bits is achieved through the use of a very low-noise input amplifier at conversion rates up to 10Hz. Effective resolutions of 20 bits can be maintained up to a sample rate of 1kHz through the use of the unique Turbo modulator mode of operation. The dynamic range of the converters is further increased by providing a low-noise programmable gain amplifier with a gain range of 1 to 16 in binary steps.

The ADS1210 and ADS1211 are designed for high resolution measurement applications in smart transmitters, industrial process control, weigh scales, chromatography and portable instrumentation. Both converters include a flexible synchronous serial interface which is SPI compatible and also offers a two-wire control mode for low cost isolation.

The ADS1210 is a single channel converter and is offered in both 18-pin DIP and 18-lead SOIC packages. The ADS1211 includes a 4 channel input multiplexer and is available in 24-pin DIP, 24-lead SOIC, and 28-lead SSOP packages.



International Airport Industrial Park • Mailing Address: PO Box 11400, Tucson, AZ 85734 • Street Address: 6730 S. Tucson Blvd., Tucson, AZ 85706 • Tel: (520) 746-1111
Twx: 910-952-1111 • Internet: http://www.burr-brown.com/ • Cable: BBRCORP • Telex: 066-6491 • FAX: (520) 889-1510 • Immediate Product Info: (800) 548-6132

PDS-1284E

Printed in U.S.A. May, 2000

SBAS034

SPECIFICATIONS

All specifications T_{MIN} to T_{MAX}, AV_{DD} = DV_{DD} = +5V, f_{XIN} = 10MHz, programmable gain amplifier setting of 1, Turbo Mode Rate of 1, REF_{OUT} disabled, V_{BIAS} disabled, and external 2.5V reference, unless otherwise specified.

PARAMETER	CONDITIONS	ADS1210U, P/ADS1211U, P, E			UNITS
		MIN	TYP	MAX	
ANALOG INPUT					
Input Voltage Range[1]		0		+5	V
	With V_{BIAS}[2]	–10		+10	V
Input Impedance	G = Gain, TMR = Turbo Mode Rate		4/(G • TMR)[3]		MΩ
Programmable Gain Amplifier	User Programmable: 1, 2, 4, 8, or 16	1		16	
Input Capacitance			8		pF
Input Leakage Current	At +25°C		5	50	pA
	At T_{MIN} to T_{MAX}			1	nA
SYSTEMS PERFORMANCE					
Resolution			24		Bits
No Missing Codes	f_{DATA} = 60Hz	22			Bits
Integral Linearity	f_{DATA} = 60Hz			±0.0015	%FSR
	f_{DATA} = 1000Hz, TMR of 16			±0.0015	%FSR
Unipolar Offset Error[4]			See Note 5		
Unipolar Offset Drift[6]			1		µV/°C
Gain Error[4]			See Note 5		
Gain Error Drift[6]			1		µV/°C
Common-Mode Rejection[9]	At DC, +25°C	100	115		dB
	At DC, T_{MIN} to T_{MAX}	90	115		dB
	50Hz, f_{DATA} = 50Hz[7]	160			dB
	60Hz, f_{DATA} = 60Hz[7]	160			dB
Normal-Mode Rejection	50Hz, f_{DATA} = 50Hz[7]	100			dB
	60Hz, f_{DATA} = 60Hz[7]	100			dB
Output Noise		See Typical Performance Curves			
Power Supply Rejection	DC, 50Hz, and 60Hz	65			dB
VOLTAGE REFERENCE					
Internal Reference (REF_{OUT})		2.4	2.5	2.6	V
Drift			25		ppm/°C
Noise			50		µVp-p
Load Current	Source or Sink			1	mA
Output Impedance			2		Ω
External Reference (REF_{IN})		2.0		3.0	V
Load Current				2.5	µA
V_{BIAS} Output	Using Internal Reference	3.15	3.3	3.45	V
Drift			50		ppm/°C
Load Current	Source or Sink			10mA	
DIGITAL INPUT/OUTPUT					
Logic Family			TTL Compatible CMOS		
Logic Level: (all except X_{IN})					
V_{IH}	I_{IH} = +5µA	2.0		DV_{DD} +0.3	V
V_{IL}	I_{IL} = +5µA	–0.3		0.8	V
V_{OH}	I_{OH} = 2 TTL Loads	2.4			V
V_{OL}	I_{OL} = 2 TTL Loads			0.4	V
X_{IN} Input Levels: V_{IH}		3.5		DV_{DD} +0.3	V
V_{IL}		–0.3		0.8	V
X_{IN} Frequency Range (f_{XIN})		0.5		10	MHz
Output Data Rate (f_{DATA})	User Programmable	2.4		15,625	Hz
	f_{XIN} = 500kHz	0.12		781	Hz
Data Format	User Programmable		Two's Complement or Offset Binary		
SYSTEM CALIBRATION					
Offset and Full-Scale Limits	V_{FS} = Full-Scale Differential Voltage[8]	0.7 • (2 • REF_{IN})/G			
V_{FS} – | V_{OS} |	V_{OS} = Offset Differential Voltage[8]			1.3 • (2 • REF_{IN})/G	

SPECIFICATIONS (CONT)

All specifications T_{MIN} to T_{MAX}, $AV_{DD} = DV_{DD} = +5V$, $f_{XIN} = 10MHz$, programmable gain amplifier setting of 1, Turbo Mode Rate of 1, REF_{OUT} disabled, V_{BIAS} disabled, and external 2.5V reference, unless otherwise specified.

PARAMETER	CONDITIONS	ADS1210U, P/ADS1211U, P, E			UNITS
		MIN	TYP	MAX	
POWER SUPPLY REQUIREMENTS					
Power Supply Voltage		4.75		5.25	V
Power Supply Current:					
Analog Current			2		mA
Digital Current			3.5		mA
Additional Analog Current with					
REF_{OUT} Enabled			1.6		mA
V_{BIAS} Enabled	No Load		1		mA
Power Dissipation			26	40	mW
	TMR of 16		37	60	mW
	f_{XIN} = 2.5MHz		17		mW
	f_{XIN} = 2.5MHz, TMR of 16		27		mW
	Sleep Mode		11		mW
TEMPERATURE RANGE					
Specified		–40		+85	°C
Storage		–60		+125	°C

NOTES: (1) In order to achieve the converter's full-scale range, the input must be fully differential ($A_{IN}N = 2 \cdot REF_{IN} – A_{IN}P$). If the input is single-ended ($A_{IN}N$ or $A_{IN}P$ is fixed), then the full scale range is one-half that of the differential range. (2) This range is set with external resistors and V_{BIAS} (as described in the text). Other ranges are possible. (3) Input impedance is higher with lower f_{XIN}. (4) Applies after calibration. (5) After system calibration, these errors will be of the order of the effective resolution of the converter. Refer to the Typical Performance Curves which apply to the desired mode of operation. (6) Recalibration can remove these errors. (7) The specification also applies at f_{DATA}/i, where i is 2, 3, 4, etc. (8) Voltages at the analog inputs must remain within AGND to AV_{DD}. (9) The common-mode rejection test is performed with a 100mV differential input.

ABSOLUTE MAXIMUM RATINGS

Analog Input: Current	±100mA, Momentary
	±10mA, Continuous
Voltage	AGND –0.3V to AV_{DD} +0.3V
AV_{DD} to DV_{DD}	–0.3V to 6V
AV_{DD} to AGND	–0.3V to 6V
DV_{DD} to DGND	–0.3V to 6V
AGND to DGND	±0.3V
REF_{IN} Voltage to AGND	–0.3V to AV_{DD} +0.3V
Digital Input Voltage to DGND	–0.3V to DV_{DD} +0.3V
Digital Output Voltage to DGND	–0.3V to DV_{DD} +0.3V
Lead Temperature (soldering, 10s)	+300°C
Power Dissipation (Any package)	500mW

PACKAGE/ORDERING INFORMATION

PRODUCT	PACKAGE	PACKAGE DRAWING NUMBER(1)	TEMPERATURE RANGE
ADS1210P	18-Pin Plastic DIP	218	–40°C to +85°C
ADS1210U	18-Lead SOIC	219	–40°C to +85°C
ADS1211P	24-Pin Plastic DIP	243	–40°C to +85°C
ADS1211U	24-Lead SOIC	239	–40°C to +85°C
ADS1211E	28-Lead SSOP	324	–40°C to +85°C

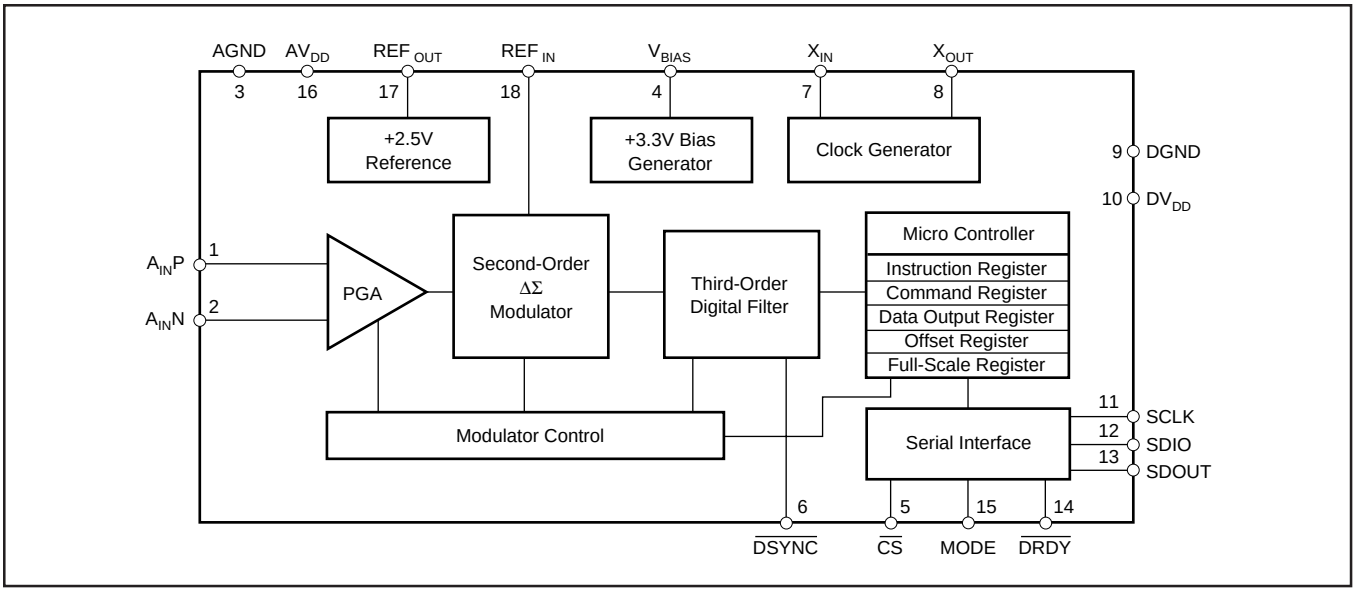
NOTE: (1) For detailed drawing and dimension table, please see end of data sheet, or Appendix C of Burr-Brown IC Data Book.

ELECTROSTATIC DISCHARGE SENSITIVITY

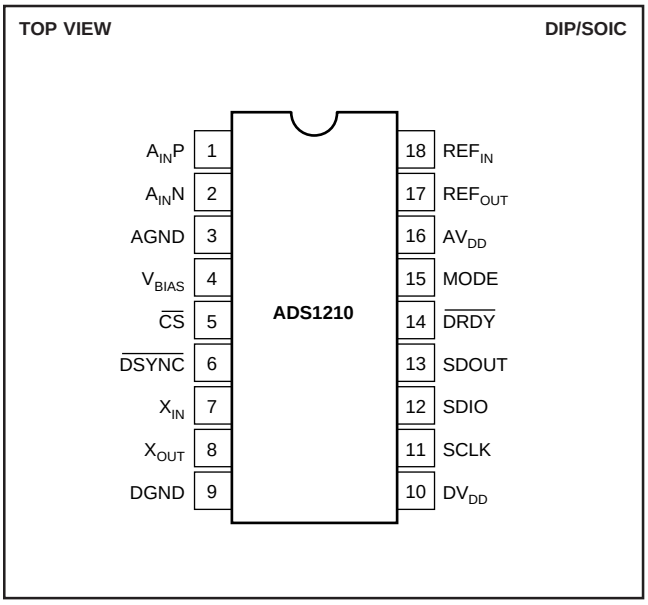
This integrated circuit can be damaged by ESD. Burr-Brown recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

Electrostatic discharge can cause damage ranging from performance degradation to complete device failure. Burr-Brown Corporation recommends that all integrated circuits be handled and stored using appropriate ESD protection methods.

ADS1210 SIMPLIFIED BLOCK DIAGRAM



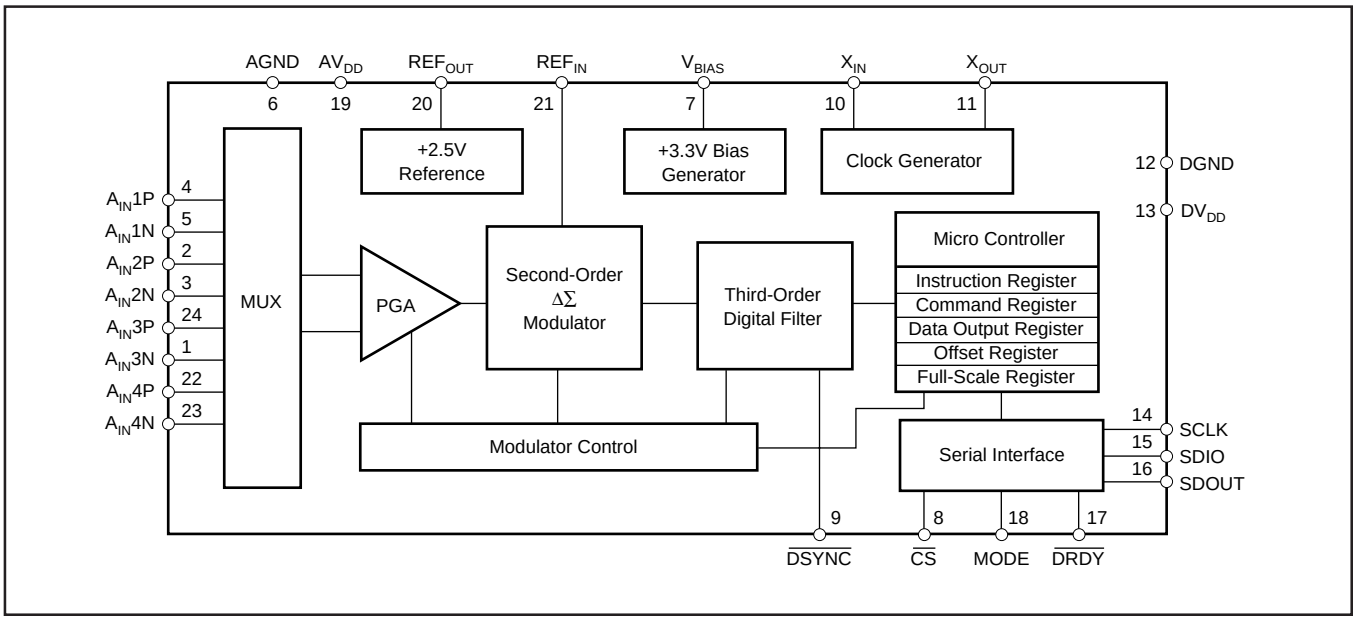
ADS1210 PIN CONFIGURATION



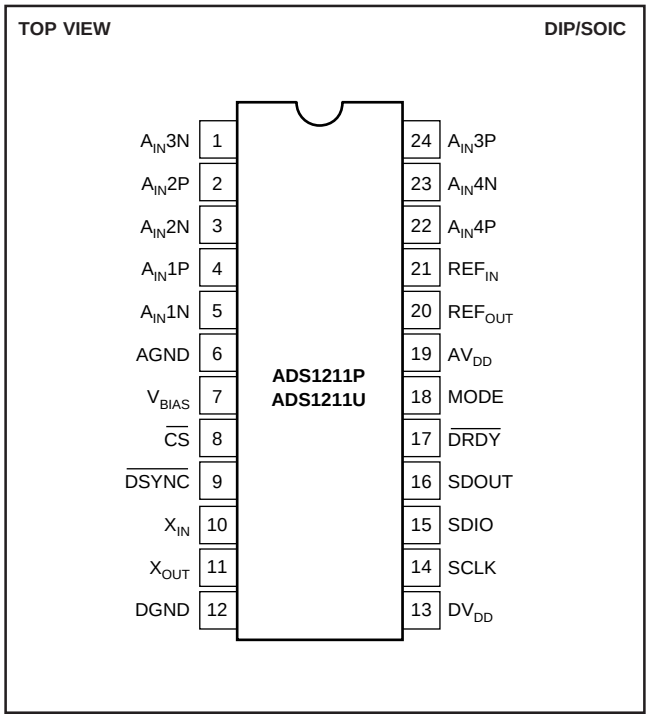
ADS1210 PIN DEFINITIONS

PIN NO	NAME	DESCRIPTION
1	$A_{IN}P$	Noninverting Input.
2	$A_{IN}N$	Inverting Input.
3	AGND	Analog Ground.
4	V_{BIAS}	Bias Voltage Output, +3.3V nominal.
5	$\overline{CS}$	Chip Select Input.
6	$\overline{DSYNC}$	Control Input to Synchronize Serial Output Data.
7	X_{IN}	System Clock Input.
8	X_{OUT}	System Clock Output (for Crystal or Resonator).
9	DGND	Digital Ground.
10	DV_{DD}	Digital Supply, +5V nominal.
11	SCLK	Clock Input/Output for serial data transfer.
12	SDIO	Serial Data Input (can also function as Serial Data Output).
13	SDOUT	Serial Data Output.
14	$\overline{DRDY}$	Data Ready.
15	MODE	SCLK Control Input (Master = 1, Slave = 0).
16	AV_{DD}	Analog Supply, +5V nominal.
17	REF_{OUT}	Reference Output, +2.5V nominal.
18	REF_{IN}	Reference Input.

ADS1211 SIMPLIFIED BLOCK DIAGRAM



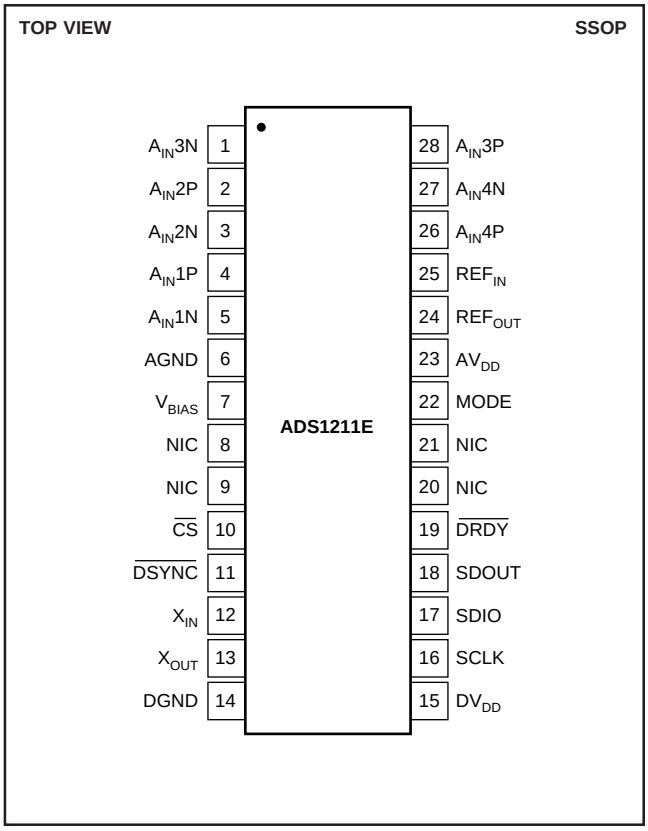
ADS1211P AND ADS1211U PIN CONFIGURATION



ADS1211P AND ADS1211U PIN DEFINITIONS

PIN NO	NAME	DESCRIPTION
1	$A_{IN}3N$	Inverting Input Channel 3.
2	$A_{IN}2P$	Noninverting Input Channel 2.
3	$A_{IN}2N$	Inverting Input Channel 2.
4	$A_{IN}1P$	Noninverting Input Channel 1.
5	$A_{IN}1N$	Inverting Input Channel 1.
6	AGND	Analog Ground.
7	V_{BIAS}	Bias Voltage Output, +3.3V nominal.
8	$\overline{CS}$	Chip Select Input.
9	$\overline{DSYNC}$	Control Input to Synchronize Serial Output Data.
10	X_{IN}	System Clock Input.
11	X_{OUT}	System Clock Output (for Crystal or Resonator).
12	DGND	Digital Ground.
13	DV_{DD}	Digital Supply, +5V nominal.
14	SCLK	Clock Input/Output for serial data transfer.
15	SDIO	Serial Data Input (can also function as Serial Data Output).
16	SDOUT	Serial Data Output.
17	$\overline{DRDY}$	Data Ready.
18	MODE	SCLK Control Input (Master = 1, Slave = 0).
19	AV_{DD}	Analog Supply, +5V nominal.
20	REF_{OUT}	Reference Output: +2.5V nominal.
21	REF_{IN}	Reference Input.
22	$A_{IN}4P$	Noninverting Input Channel 4.
23	$A_{IN}4N$	Inverting Input Channel 4.
24	$A_{IN}3P$	Noninverting Input Channel 3.

ADS1211E PIN CONFIGURATION



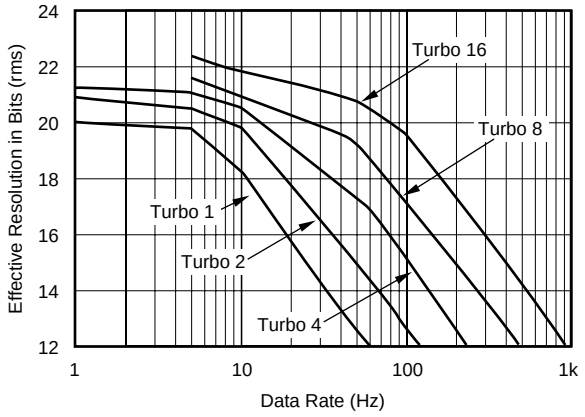
ADS1211E PIN DEFINITIONS

PIN NO	NAME	DESCRIPTION
1	$A_{IN}3N$	Inverting Input Channel 3.
2	$A_{IN}2P$	Noninverting Input Channel 2.
3	$A_{IN}2N$	Inverting Input Channel 2.
4	$A_{IN}1P$	Noninverting Input Channel 1.
5	$A_{IN}1N$	Inverting Input Channel 1.
6	AGND	Analog Ground.
7	V_{BIAS}	Bias Voltage Output, +3.3V nominal.
8	NIC	Not Internally Connected.
9	NIC	Not Internally Connected.
10	$\overline{CS}$	Chip Select Input.
11	DSYNC	Control Input to Synchronize Serial Output Data.
12	X_{IN}	System Clock Input.
13	X_{OUT}	System Clock Output (for Crystal or Resonator).
14	DGND	Digital Ground.
15	DV_{DD}	Digital Supply, +5V nominal.
16	SCLK	Clock Input/Output for serial data transfer.
17	SDIO	Serial Data Input (can also function as Serial Data Output).
18	SDOUT	Serial Data Output.
19	$\overline{DRDY}$	Data Ready.
20	NIC	Not Internally Connected.
21	NIC	Not Internally Connected.
22	MODE	SCLK Control Input (Master = 1, Slave = 0).
23	AV_{DD}	Analog Supply, +5V nominal.
24	REF_{OUT}	Reference Output: +2.5V nominal.
25	REF_{IN}	Reference Input.
26	$A_{IN}4P$	Noninverting Input Channel 4.
27	$A_{IN}4N$	Inverting Input Channel 4.
28	$A_{IN}3P$	Noninverting Input Channel 3.

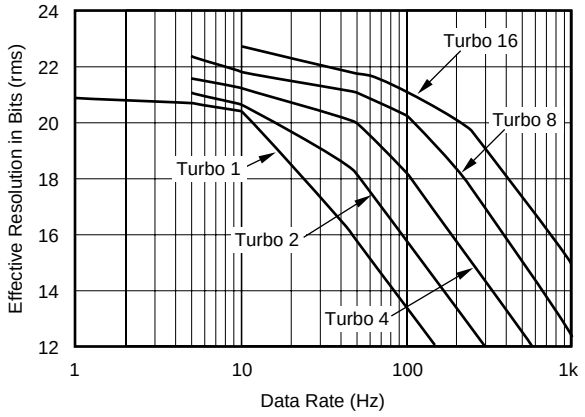
TYPICAL PERFORMANCE CURVES

At T_A = +25°C, AV_{DD} = DV_{DD} = +5V, f_{XIN} = 10MHz, programmable gain amplifier setting of 1, Turbo Mode Rate of one, REF_{OUT} disabled, V_{BIAS} disabled, and external 2.5V reference, unless otherwise noted.

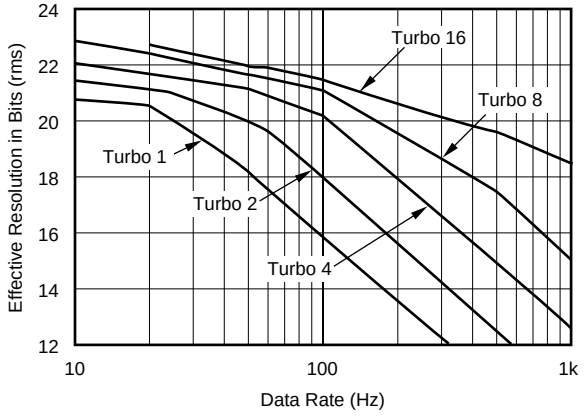
EFFECTIVE RESOLUTION vs DATA RATE
(1MHz Clock)



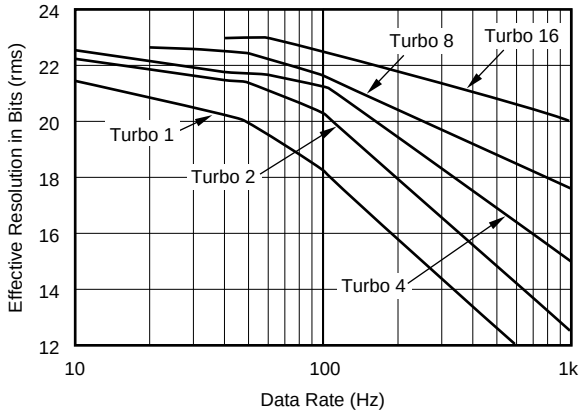
EFFECTIVE RESOLUTION vs DATA RATE
(2.5MHz Clock)



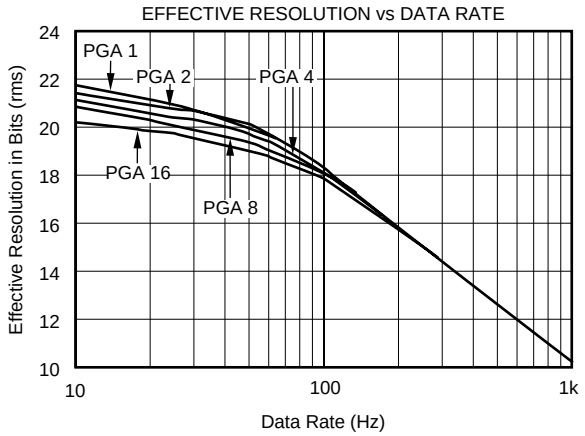
EFFECTIVE RESOLUTION vs DATA RATE
(5MHz Clock)



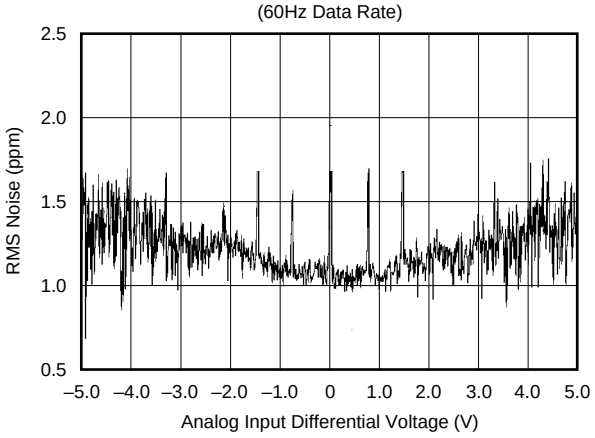
EFFECTIVE RESOLUTION vs DATA RATE
(10MHz Clock)



EFFECTIVE RESOLUTION vs DATA RATE



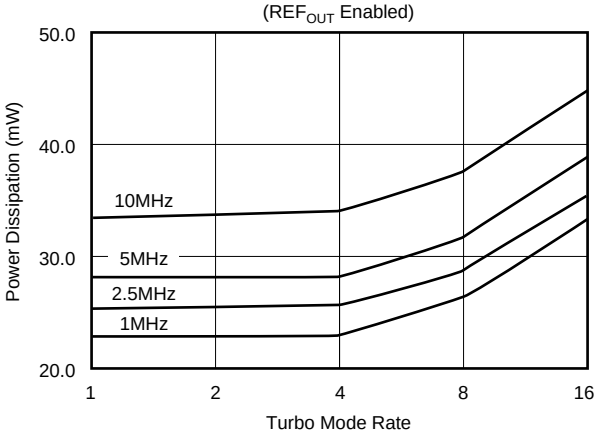
RMS NOISE vs INPUT VOLTAGE LEVEL
(60Hz Data Rate)



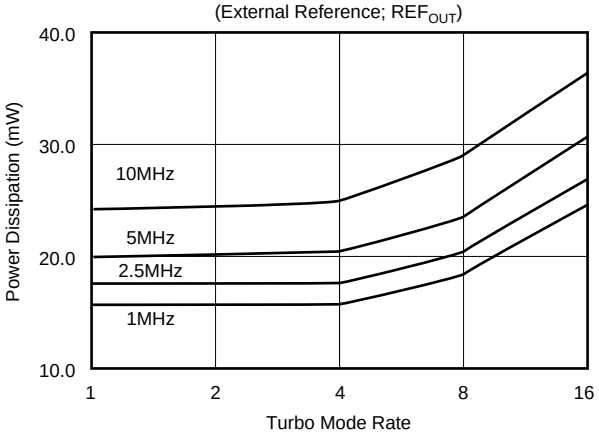
TYPICAL PERFORMANCE CURVES (CONT)

At T_A = +25°C, AV_{DD} = DV_{DD} = +5V, f_{XIN} = 10MHz, programmable gain amplifier setting of 1, Turbo Mode Rate of 1, REF_{OUT} disabled, V_{BIAS} disabled, and external 2.5V reference, unless otherwise noted.

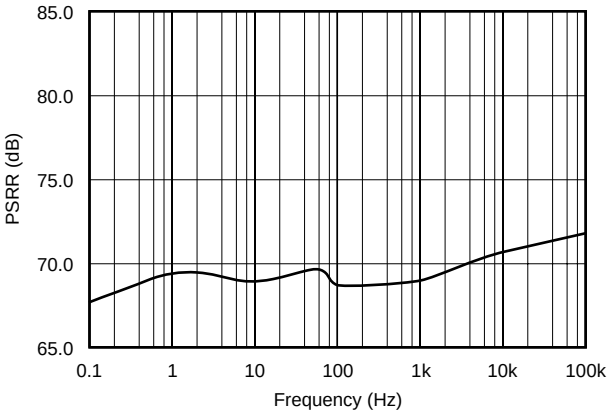
POWER DISSIPATION vs TURBO MODE RATE
(REF_{OUT} Enabled)



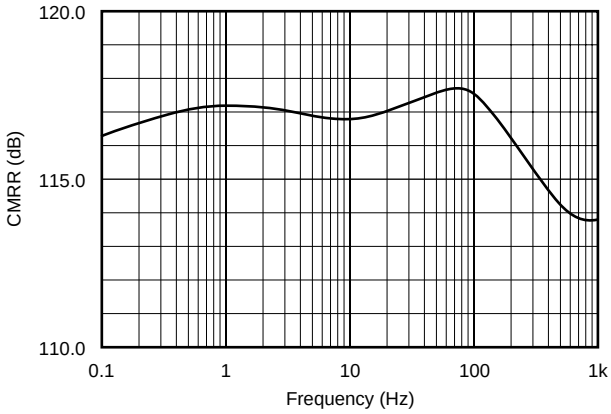
POWER DISSIPATION vs TURBO MODE RATE
(External Reference; REF_{OUT})



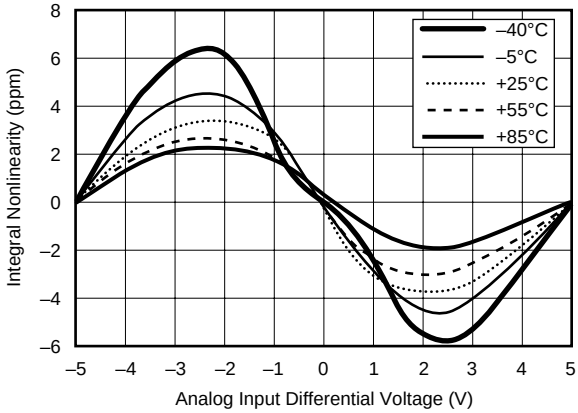
PSRR vs FREQUENCY



CMRR vs FREQUENCY



LINEARITY vs TEMPERATURE
(60Hz Data Rate)



THEORY OF OPERATION

The ADS1210 and ADS1211 are precision, high dynamic range, self-calibrating, 24-bit, delta-sigma A/D converters capable of achieving very high resolution digital results. Each contains a programmable gain amplifier (PGA); a second-order delta-sigma modulator; a programmable digital filter; a microcontroller including the Instruction, Command and Calibration registers; a serial interface; a clock generator circuit; and an internal 2.5V reference. The ADS1211 includes a 4-channel input multiplexer.

In order to provide low system noise, common-mode rejection of 115dB and excellent power supply rejection, the design topology is based on a fully differential switched capacitor architecture. Turbo Mode, a unique feature of the ADS1210/11, can be used to boost the sampling rate of the input capacitor, which is normally 19.5kHz with a 10MHz clock. By programming the Command Register, the sampling rate can be increased to 39kHz, 78kHz, 156kHz, or 312kHz. Each increase in sample rate results in an increase in performance when maintaining the same output data rate.

The programmable gain amplifier (PGA) of the ADS1210/11 can be set to a gain of 1, 2, 4, 8 or 16—substantially increasing the dynamic range of the converter and simplifying the interface to the more common transducers (see Table I). This gain is implemented by increasing the number of samples taken by the input capacitor from 19.5kHz for a gain of 1 to 312kHz for a gain of 16. Since the Turbo Mode and PGA functions are both implemented by varying the sampling frequency of the input capacitor, the combination of PGA gain and Turbo Mode Rate is limited to 16 (see Table II). For example, when using a Turbo Mode Rate of 8 (156kHz at 10MHz), the maximum PGA gain setting is 2.

GAIN SETTING	ANALOG INPUT[1]		ANALOG INPUT UTILIZING V_{BIAS}[1,2]	
	FULL-SCALE RANGE (V)	EXAMPLE VOLTAGE RANGE[3] (V)	FULL-SCALE RANGE (V)	EXAMPLE VOLTAGE RANGE[3] (V)
1	10	0 to 5	40	±10
2	5	1.25 to 3.75	20	±5
4	2.5	1.88 to 3.13	10	±2.5
8	1.25	2.19 to 2.81	5	±1.25
16	0.625	2.34 to 2.66	2.5	±0.625

NOTE: (1) With a 2.5V reference, such as the internal reference. (2) This example utilizes the circuit in Figure 12. Other input ranges are possible. (3) The ADS1210/11 allows common-mode voltage as long as the absolute input voltage on $A_{IN}P$ or $A_{IN}N$ does not go below AGND or above AV_{DD}.

TABLE I. Full-Scale Range vs PGA Setting.

TURBO MODE RATE	AVAILABLE PGA SETTINGS
1	1, 2, 4, 8, 16
2	1, 2, 4, 8
4	1, 2, 4
8	1, 2
16	1

TABLE II. Available PGA Settings vs Turbo Mode Rate.

The output data rate of the ADS1210/11 can be varied from a few hertz to as much as 15,625kHz, trading off lower resolution results for higher data rates. In addition, the data rate determines the first null of the digital filter and sets the –3dB point of the input bandwidth (see the Digital Filter section). Changing the data rate of the ADS1210/11 does not result in a change in the sampling rate of the input capacitor. The data rate effectively sets the number of samples which are used by the digital filter to obtain each conversion result. A lower data rate results in higher resolution, lower input bandwidth, and different notch frequencies than a higher data rate. It does not result in any change in input impedance or modulator frequency, or any appreciable change in power consumption.

The ADS1210/11 also includes complete on-board calibration that can correct for internal offset and gain errors or limited external system errors. Internal calibration can be run when needed, or automatically and continuously in the background. System calibration can be run as needed and the appropriate input voltages must be provided to the ADS1210/11. For this reason, there is no continuous System Calibration Mode. The calibration registers are fully readable and writable. This feature allows for switching between various configurations—different data rates, Turbo Mode Rates, and gain settings—without re-calibrating.

The various settings, rates, modes, and registers of the ADS1210/11 are read or written via a synchronous serial interface. This interface can operate in either a self-clocked mode (Master Mode) or an externally clocked mode (Slave Mode). In the Master Mode, the serial clock (SCLK) frequency is one-half of the ADS1210/11 X_{IN} clock frequency. This is an important consideration for many systems and may determine the maximum ADS1210/11 clock that can be used.

The high resolution and flexibility of the ADS1210/11 allow these converters to fill a wide variety of A/D conversion tasks. In order to ensure that a particular configuration will meet the design goals, there are several important items which must be considered. These include (but are certainly not limited to) the needed resolution, required linearity, desired input bandwidth, power consumption goal, and sensor output voltage.

The remainder of this data sheet discusses the operation of the ADS1210/11 in detail. In order to allow for easier comparison of different configurations, “effective resolution” is used as the figure of merit for most tables and graphs. For example, Table III shows a comparison between data rate (and –3dB input bandwidth) versus PGA setting at a Turbo Mode Rate of 1 and a clock rate of 10MHz. See the Definition of Terms section for a definition of effective resolution.

DATA RATE (HZ)	-3DB FREQUENCY (HZ)	EFFECTIVE RESOLUTION (BITS RMS)				
		G = 1	G = 2	G = 4	G = 8	G = 16
10	2.62	21.5	21.0	21.0	21.0	20.0
25	6.55	20.5	20.5	20.5	20.0	19.5
30	7.86	20.5	20.5	20.5	20.0	19.5
50	13.1	20.0	20.0	20.0	19.5	19.0
60	15.7	19.5	19.5	19.5	19.0	19.0
100	26.2	18.0	18.0	18.0	18.0	18.0
250	65.5	15.0	15.0	15.0	15.0	15.0
500	131	12.5	12.5	12.5	12.5	12.5
1000	262	10.0	10.5	10.0	10.0	10.0

TABLE III. Effective Resolution vs Data Rate and Gain Setting. (Turbo Mode Rate of 1 and a 10MHz clock.)

DEFINITION OF TERMS

An attempt has been made to be consistent with the terminology used in this data sheet. In that regard, the definition of each term is given as follows:

Analog Input Differential Voltage—For an analog signal that is fully differential, the voltage range can be compared to that of an instrumentation amplifier. For example, if both analog inputs of the ADS1210 are at 2.5V, then the differential voltage is 0V. If one is at 0V and the other at 5V, then the differential voltage magnitude is 5V. But, this is the case regardless of which input is at 0V and which is at 5V, while the digital output result is quite different.

The analog input differential voltage is given by the following equation: $A_{IN}P - A_{IN}N$. Thus, a positive digital output is produced whenever the analog input differential voltage is positive, while a negative digital output is produced whenever the differential is negative.

For example, when the converter is configured with a 2.5V reference and placed in a gain setting of 2, the positive full-scale output is produced when the analog input differential is 2.5V. The negative full-scale output is produced when the differential is –2.5V. In each case, the actual input voltages must remain within the AGND to AV_{DD} range (see Table I).

Actual Analog Input Voltage—The voltage at any one analog input relative to AGND.

Full-Scale Range (FSR)—As with most A/D converters, the full-scale range of the ADS1210/11 is defined as the "input" which produces the positive full-scale digital output minus the "input" which produces the negative full-scale digital output.

For example, when the converter is configured with a 2.5V reference and is placed in a gain setting of 2, the full-scale range is: [2.5V (positive full scale) minus –2.5V (negative full scale)] = 5V.

Typical Analog Input Voltage Range—This term describes the actual voltage range of the analog inputs which will cover the converter's full-scale range, assuming that each input has a common-mode voltage that is greater than REF_{IN}/PGA and smaller than ($AV_{DD} - REF_{IN}/PGA$).

For example, when the converter is configured with a 2.5V reference and placed in a gain setting of 2, the typical input voltage range is 1.25V to 3.75V. However, an input range of 0V to 2.5V or 2.5V to 5V would also cover the converter's full-scale range.

Voltage Span—This is simply the magnitude of the typical analog input voltage range. For example, when the converter is configured with a 2.5V reference and placed in a gain setting of 2, the input voltage span is 2.5V.

Least Significant Bit (LSB) Weight—This is the theoretical amount of voltage that the differential voltage at the analog input would have to change in order to observe a change in the output data of one least significant bit. It is computed as follows:

$$\text{LSB Weight} = \frac{\text{Full–Scale Range}}{2^N}$$

where N is the number of bits in the digital output.

Effective Resolution—The effective resolution of the ADS1210/11 in a particular configuration can be expressed in two different units: bits rms (referenced to output) and microvolts rms (referenced to input). Computed directly from the converter's output data, each is a statistical calculation based on a given number of results. Knowing one, the other can be computed as follows:

$$\text{ER in bits rms} = \frac{20 \bullet \log\left(\frac{\left(\frac{10\text{V}}{\text{PGA}}\right)}{\text{ER in Vrms}}\right) - 1.76}{6.02}$$

$$\text{ER in Vrms} = \frac{\left(\frac{10\text{V}}{\text{PGA}}\right)}{10^{\left(\frac{6.02 \bullet \text{ER in bits rms} + 1.76}{20}\right)}}$$

The 10V figure in each calculation represents the full-scale range of the ADS1210/11 in a gain setting of 1. This means that both units are absolute expressions of resolution—the performance in different configurations can be directly compared regardless of the units. Comparing the resolution of different gain settings expressed in bits rms requires accounting for the PGA setting.

Main Controller—A generic term for the external microcontroller, microprocessor, or digital signal processor which is controlling the operation of the ADS1210/11 and receiving the output data.

f_{XIN}—The frequency of the crystal oscillator or CMOS compatible input signal at the X_{IN} input of the ADS1210/11.

f_{MOD}—The frequency or speed at which the modulator of the ADS1210/11 is running, given by the following equation:

$$f_{MOD} = \frac{f_{XIN} \bullet \text{Turbo Mode}}{512}$$

f_{SAMP}—The frequency or switching speed of the input sampling capacitor. The value is given by the following equation:

$$f_{SAMP} = \frac{f_{XIN} \bullet \text{Turbo Mode} \bullet \text{Gain Setting}}{512}$$

f_{DATA}, t_{DATA}—The frequency of the digital output data produced by the ADS1210/11 or the inverse of this (the period), respectively, f_{DATA} is also referred to as the data rate.

$$f_{DATA} = \frac{f_{XIN} \bullet \text{Turbo Mode}}{512 \bullet (\text{Decimation Ratio} + 1)}, \quad t_{DATA} = \frac{1}{f_{DATA}}$$

Conversion Cycle—The term "conversion cycle" usually refers to a discrete A/D conversion operation, such as that performed by a successive approximation converter. As used here, a conversion cycle refers to the t_{DATA} time period. However, each digital output is actually based on the modulator results from the last three t_{DATA} time periods.

DIGITAL FILTER

The digital filter of the ADS1210/11 computes the output result based on the most recent results from the delta-sigma modulator. The number of modulator results that are used depend on the decimation ratio set in the Command Register. At the most basic level, the digital filter can be thought of as simply averaging the modulator results and presenting this average as the digital output.

While the decimation ratio determines the number of modulator results to use, the modulator runs faster at higher Turbo Modes. These two items, together with the ADS1210/11 clock frequency, determine the output data rate:

$$f_{DATA} = \frac{f_{XIN} \bullet \text{Turbo Mode}}{512 \bullet (\text{Decimation Ratio} + 1)}$$

Also, since the conversion result is essentially an average, the data rate determines where the resulting notches are in the digital filter. For example, if the output data rate is 1kHz, then a 1kHz input frequency will average to zero during the 1ms conversion cycle. Likewise, a 2kHz input frequency will average to zero, etc.

In this manner, the data rate can be used to set specific notch frequencies in the digital filter response (see Figure 1 for the normalized response of the digital filter). For example, if the rejection of power line frequencies is desired, then the data rate can simply be set to the power line frequency. Figures 2 and 3 show the digital filter response for a data rate of 50Hz and 60Hz, respectively.

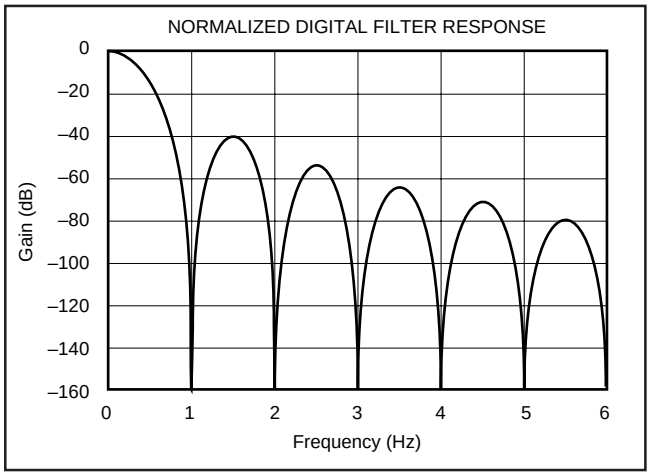


FIGURE 1. Normalized Digital Filter Response.

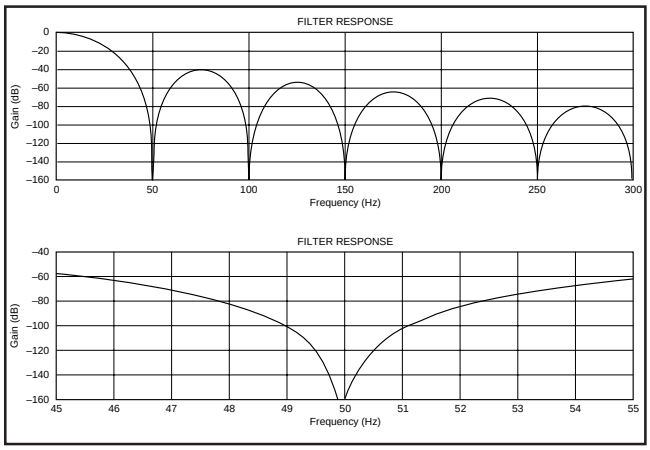


FIGURE 2. Digital Filter Response at a Data Rate of 50Hz.

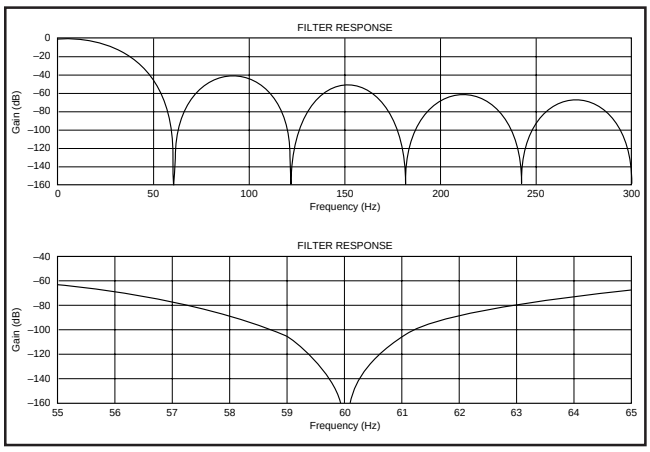


FIGURE 3. Digital Filter Response at a Data Rate of 60Hz.

If the effective resolution at a 50Hz or 60Hz data rate is not adequate for the particular application, then power line frequencies could still be rejected by operating the ADS1210/11 at 25/30Hz, 16.7/20Hz, 12.5/15Hz, etc. If a higher data rate is needed, then power line frequencies must either be rejected before conversion (with an analog notch filter) or after conversion (with a digital notch filter running on the main controller).

Filter Equation

The digital filter is described by the following transfer function:

$$|H(f)| = \left| \frac{\sin\left(\frac{\pi \bullet f \bullet N}{f_{MOD}}\right)}{N \bullet \sin\left(\frac{\pi \bullet f}{f_{MOD}}\right)} \right|^3$$

where N is the Decimation Ratio.

This filter has a $(\sin(x)/x)^3$ response and is referred to a $sinc^3$ filter. For the ADS1210/11, this type of filter allows the data rate to be changed over a very wide range (nearly four orders of magnitude). However, the –3dB point of the filter is 0.262 times the data rate. And, as can be seen in Figures 1 and 2, the rejection in the stopband (frequencies higher than the first notch frequency) may only be –40dB.

These factors must be considered in the overall system design. For example, with a 50Hz data rate, a significant signal at 75Hz may alias back into the passband at 25Hz. The analog front end can be designed to provide the needed attenuation to prevent aliasing, or the system may simply provide this inherently. Another possibility is increasing the data rate and then post filtering with a digital filter on the main controller.

Filter Settling

The number of modulator results used to compute each conversion result is three times the Decimation Ratio. This means that any step change (or any channel change for the ADS1211) will require at least three conversions to fully settle. However, if the change occurs asynchronously, then at least four conversions are required to ensure complete settling. For example, on the ADS1211, the fourth conversion result after a channel change will be valid (see Figure 4).

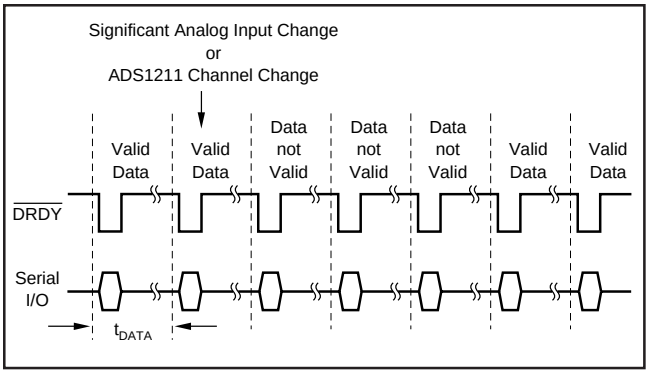


FIGURE 4. Asynchronous ADS1210/11 Analog Input Voltage Step or ADS1211 Channel Change to Fully Settled Output Data.

TURBO MODE

The ADS1210/11 offers a unique Turbo Mode feature which can be used to increase the modulator sampling rate by 2, 4, 8, or 16 times normal. With the increase of modulator sampling frequency, there can be a substantial increase in the effective resolution of the output data at a given data rate, but there is also an increase in power dissipation. For Turbo Mode Rates 2 and 4, the increase is slight. For rates 8 and 16, the increase is more substantial. See the Typical Performance Curves for more information.

In a Turbo Mode Rate of 16, the ADS1210/11 can offer 20 bits of effective resolution at a 1kHz data rate. A comparison of effective resolution versus Turbo Mode Rates and output data rates is shown in Table IV while Table V shows the corresponding noise level in μVrms.

Data Rate (Hz)	Effective Resolution (Bits rms)				
	Turbo Mode Rate 1	Turbo Mode Rate 2	Turbo Mode Rate 4	Turbo Mode Rate 8	Turbo Mode Rate 16
10	21.5	22.0	22.5		
20	21.0	22.0	22.0	22.5	
40	20.0	21.5	22.0	22.5	23.0
50	20.0	21.5	21.5	22.0	23.0
60	19.5	21.0	21.5	22.0	23.0
100	18.0	20.0	21.0	21.5	22.5
1000	10.0	12.5	15.0	17.5	20.0

TABLE IV. Effective Resolution vs Data Rate and Turbo Mode Rate. (Gain setting of 1 and 10MHz clock.)

DATA RATE (Hz)	NOISE LEVEL (μVrms)				
	TURBO MODE RATE 1	TURBO MODE RATE 2	TURBO MODE RATE 4	TURBO MODE RATE 8	TURBO MODE RATE 16
10	2.9	1.7	1.3		
20	4.3	2.1	1.7	1.3	
40	6.9	3.0	2.3	1.6	1.0
50	8.1	3.2	2.4	1.8	1.0
60	10.5	3.9	2.6	1.9	1.0
100	26.9	6.9	3.5	2.7	1.4
1000	6909.7	1354.5	238.4	46.6	7.8

TABLE V. Noise Level vs Data Rate and Turbo Mode Rate. (Gain setting of 1 and 10MHz clock.)

The Turbo Mode feature allows trade-offs to be made between the ADS1210/11 X_{IN} clock frequency, power dissipation, and effective resolution. If a 5MHz clock is available but a 10MHz clock is needed to achieve the desired performance, a Turbo Mode Rate of 2X will result in the same effective resolution. Table VI provides a comparison of effective resolution at various clock frequencies, data rates, and Turbo Mode Rates.

DATA RATE (Hz)	X_{IN} CLOCK FREQUENCY (MHz)	TURBO MODE RATE	EFFECTIVE RESOLUTION (Bits rms)
60	10	1	19.5
60	5	2	19.5
60	2.5	4	19.5
60	1.25	8	19.5
60	0.625	16	19.5
100	10	1	18.0
100	5	2	18.0
100	2.5	4	18.0
100	1.25	8	18.0
100	0.625	16	18.0

TABLE VI. Effective Resolution vs Data Rate, Clock Frequency, and Turbo Mode Rate. (Gain setting of 1.)

The Turbo Mode Rate (TMR) is programmed via the Sampling Frequency bits of the Command Register. Due to the increase in input capacitor sampling frequency, higher Turbo Mode settings result in lower analog input impedance;

$$A_{IN} \text{ Impedance } (\Omega) = (10MHz/f_{XIN}) \bullet 4.3E6/(G \bullet TMR)$$

where G is the gain setting. Because the modulator rate also changes in direct relation to the Turbo Mode setting, higher values result in a lower impedance for the REF_{IN} input:

$$REF_{IN} \text{ Impedance } (\Omega) = (10MHz/f_{XIN}) \bullet 1E6/TMR$$

The Turbo Mode Rate can be set to 1, 2, 4, 8, or 16. Consult the graphs shown in the Typical Performance Curves for full details on the performance of the ADS1210/11 operating in different Turbo Mode Rates. Keep in mind that higher Turbo Mode Rates result in fewer available gain settings as shown in Table II.

PROGRAMMABLE GAIN AMPLIFIER

The programmable gain amplifier gain setting is programmed via the PGA Gain bits of the Command Register. Changes in the gain setting (G) of the programmable gain amplifier results in an increase in the input capacitor sampling frequency. Thus, higher gain settings result in a lower analog input impedance:

$$A_{IN} \text{ Impedance } (\Omega) = (10MHz/f_{XIN}) \bullet 4.3E6/(G \bullet TMR)$$

where TMR is the Turbo Mode Rate. Because the modulator speed does not depend on the gain setting, the input impedance seen at REF_{IN} does not change.

The PGA can be set to gains of 1, 2, 4, 8, or 16. These gain settings with their resulting full-scale range and typical voltage range are shown in Table I. Keep in mind that higher Turbo Mode Rates result in fewer available gain settings as shown in Table II.

SOFTWARE GAIN

The excellent performance, flexibility, and low cost of the ADS1210/11 allow the converter to be considered for designs which would not normally need a 24-bit ADC. For example, many designs utilize a 12-bit converter and a high-gain INA or PGA for digitizing low amplitude signals. For some of these cases, the ADS1210/11 by itself may be a solution, even though the maximum gain is limited to 16.

To get around the gain limitation, the digital result can simply be shifted up by "n" bits in the main controller—resulting in a gain of "n" times G, where G is the gain setting. While this type of manipulation of the output data is obvious, it is easy to miss how much the gain can be increased in this manner on a 24-bit converter.

For example, shifting the result up by three bits when the ADS1210/11 is set to a gain of 16 results in an effective gain of 128. At lower data rates, the converter can easily provide more than 12 bits of resolution. Even higher gains are possible. The limitation is a combination of the needed data rate, desired noise performance, and desired linearity.

CALIBRATION

The ADS1210/11 offers several different types of calibration, and the particular calibration desired is programmed via the Command Register. In the case of Background Calibration, the calibration will repeat at regular intervals indefinitely. For all others, the calibration is performed once and then normal operation is resumed.

Each type of calibration is covered in detail in their respective section. In general, calibration is recommended immediately after power-on and whenever there is a "significant" change in the operating environment. The amount of change which should cause a re-calibration is dependent on the application, effective resolution, etc. Where high accuracy is important, re-calibration should be done on changes in temperature and power supply. In all cases, re-calibration should be done when the gain, Turbo Mode, or data rate is changed.

After a calibration has been accomplished, the Offset Calibration Register and the Full-Scale Calibration Register contain the results of the calibration. The data in these registers are accurate to the effective resolution of the ADS1210/11's mode of operation during the calibration. Thus, these values will show a variation (or noise) equivalent to a regular conversion result.

For those cases where this error must be reduced, it is tempting to consider running the calibration at a slower data rate and then increasing the converter's data rate after the calibration is complete. Unfortunately, this will not work as expected. The reason is that the results calculated at the slower data rate would not be valid for the higher data rate. Instead, the calibration should be done repeatedly. After each calibration, the results can be read and stored. After the desired number of calibrations, the main controller can compute an average and write this value into the calibration registers. The resulting error in the calibration values will be reduced by the square root of the number of calibrations which were averaged.

The calibration registers can also be used to provide system offset and gain corrections separate from those computed by the ADS1210/11. For example, these might be burned into E^2PROM during final product testing. On power-on, the main controller would load these values into the calibration registers. A further possibility is a look-up table based on the current temperature.

Note that the values in the calibration registers will vary from configuration to configuration and from part to part. There is no method of reliably computing what a particular calibration register should be to correct for a given amount of system error. It is possible to present the ADS1210/11 with a known amount of error, perform a calibration, read the desired calibration register, change the error value, perform another calibration, read the new value and use these values to interpolate an intermediate value.

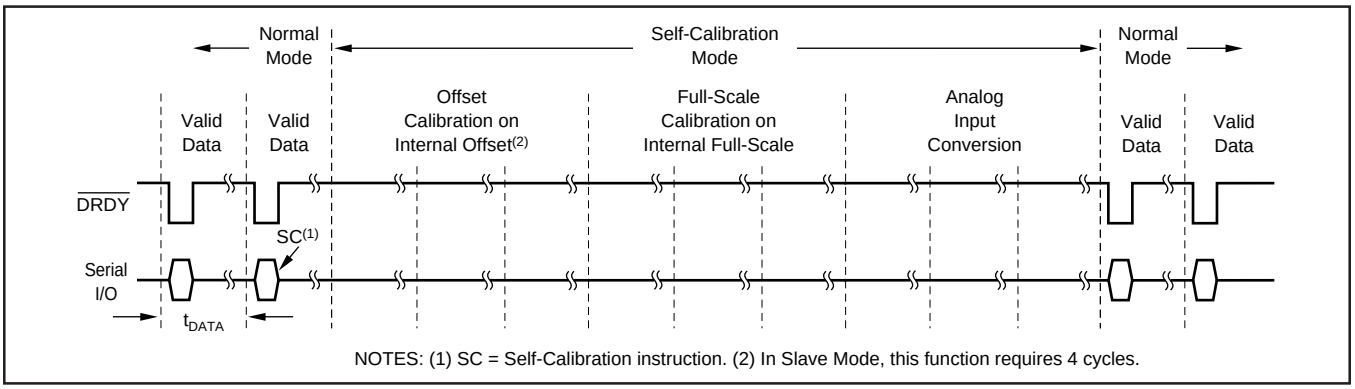


FIGURE 5. Self-Calibration Timing.

Self-Calibration

A self-calibration is performed after the bits 001 have been written to the Command Register Operation Mode bits (MD2 through MD0). This initiates the following sequence at the start of the next conversion cycle (see Figure 5). The $\overline{DRDY}$ signal will not go LOW but will remain HIGH and will continue to remain HIGH throughout the calibration sequence. The inputs to the sampling capacitor are disconnected from the converter's analog inputs and are shorted together. An offset calibration is performed over the next three conversion periods (four in Slave Mode). Then, the input to the sampling capacitor is connected across REF_{IN}, and a full-scale calibration is performed over the next three conversions.

After this, the Operation Mode bits are reset to 000 (normal mode) and the input capacitor is reconnected to the input. Conversions proceed as usual over the next three cycles in order to fill the digital filter. $\overline{DRDY}$ remains HIGH during this time. On the start of the fourth cycle, $\overline{DRDY}$ goes LOW indicating valid data and resumption of normal operation.

System Offset Calibration

A system offset calibration is performed after the bits 010 have been written to the Command Register Operation Mode bits (MD2 through MD0). This initiates the following sequence (see Figure 6). At the start of the next conversion cycle, the $\overline{DRDY}$ signal will not go LOW but will remain HIGH and will continue to remain HIGH throughout the calibration sequence. The offset calibration will be performed on the differential input voltage present at the converter's input over the next three conversion periods (four in Slave Mode). When this is done, the Operation Mode bits are reset to 000 (Normal Mode). A single conversion is done with $\overline{DRDY}$ HIGH. After this conversion, the $\overline{DRDY}$ signal goes LOW indicating resumption of normal operation.

Normal operation returns within a single conversion cycle because it is assumed that the input voltage at the converter's input is not removed immediately after the offset calibration is performed. In this case, the digital filter already contains a valid result.

For full system calibration, offset calibration must be performed first and then full-scale calibration. In addition, the offset calibration error will be the rms sum of the conversion error and the noise on the system offset voltage. See the System Calibration Limits section for information regarding the limits on the magnitude of the system offset voltage.

System Full-Scale Calibration

A system full-scale calibration is performed after the bits 011 have been written to the Command Register Operation Mode bits (MD2 through MD0). This initiates the following sequence (see Figure 7). At the start of the next conversion cycle, the $\overline{DRDY}$ signal will not go LOW but will remain HIGH and will continue to remain HIGH throughout the calibration sequence. The full-scale calibration will be performed on the differential input voltage ($2 \cdot REF_{IN}/G$) present at the converter's input over the next three conversion periods (four in Slave Mode). When this is done, the Operation Mode bits are reset to 000 (Normal Mode). A single conversion is done with $\overline{DRDY}$ HIGH. After this conversion, the $\overline{DRDY}$ signal goes LOW indicating resumption of normal operation.

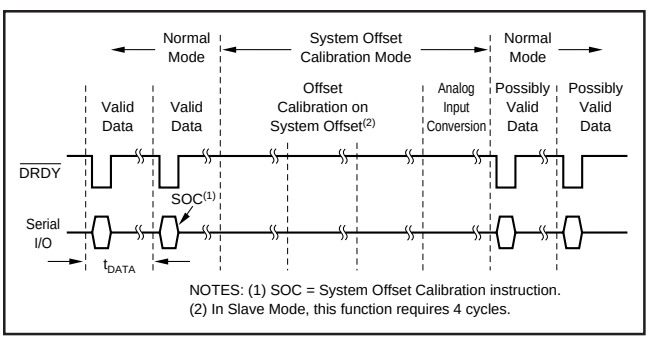


FIGURE 6. System Offset Calibration Timing.

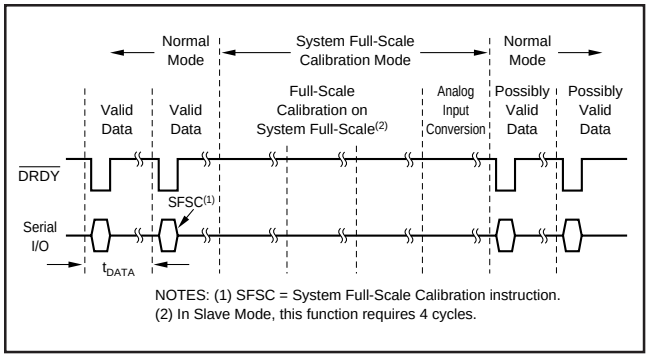


FIGURE 7. System Full-Scale Calibration Timing.

Normal operation returns within a single conversion cycle because it is assumed that the input voltage at the converter's input is not removed immediately after the full-scale calibration is performed. In this case, the digital filter already contains a valid result.

For full system calibration, offset calibration must be performed first and then full-scale calibration. The calibration error will be a sum of the rms noise on the conversion result and the input signal noise. See the System Calibration Limits section for information regarding the limits on the magnitude of the system full-scale voltage.

Pseudo System Calibration

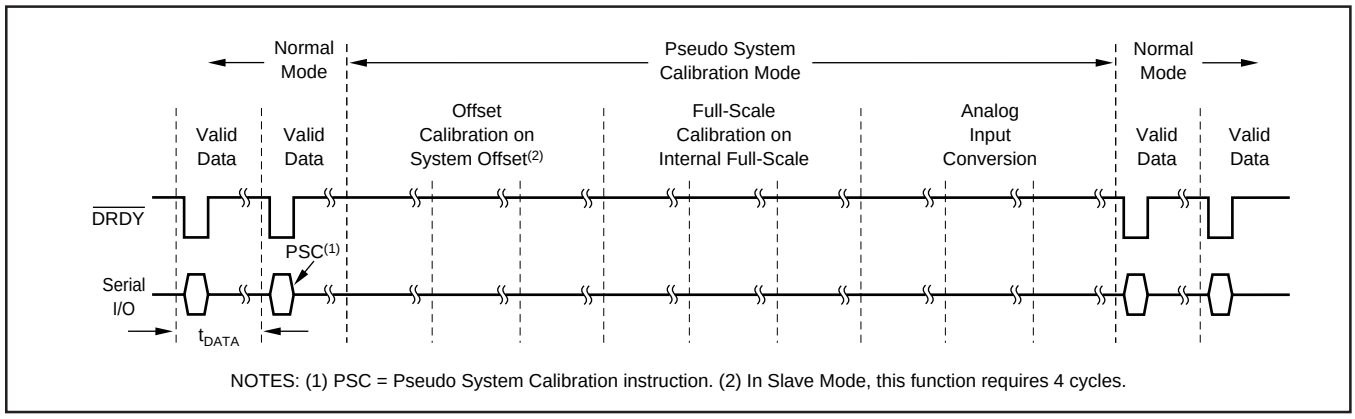
The Pseudo System Calibration is performed after the bits 100 have been written to the Command Register Operation Mode bits (MD2 through MD0). This initiates the following sequence (see Figure 8). At the start of the next conversion cycle, the $\overline{DRDY}$ signal will not go LOW but will remain HIGH and will continue to remain HIGH throughout the calibration sequence. The offset calibration will be performed on the differential input voltage present at the converter's input over the next three conversion periods (four in Slave Mode). Then, the input to the sampling capacitor is disconnected from the converter's analog input and connected across REF_{IN}. A gain calibration is performed over the next three conversions.

After this, the Operation Mode bits are reset to 000 (normal mode) and the input capacitor is then reconnected to the input. Conversions proceed as usual over the next three cycles in order to fill the digital filter. $\overline{DRDY}$ remains HIGH during this time. On the next cycle, the $\overline{DRDY}$ signal goes LOW indicating valid data and resumption of normal operation.

The system offset calibration range of the ADS1210/11 is limited and is listed in the Specifications Table. For more information on how to use these specifications, see the System Calibration Limits section. To calculate V_{OS}, use 2 • REF_{IN}/GAIN for V_{FS}.

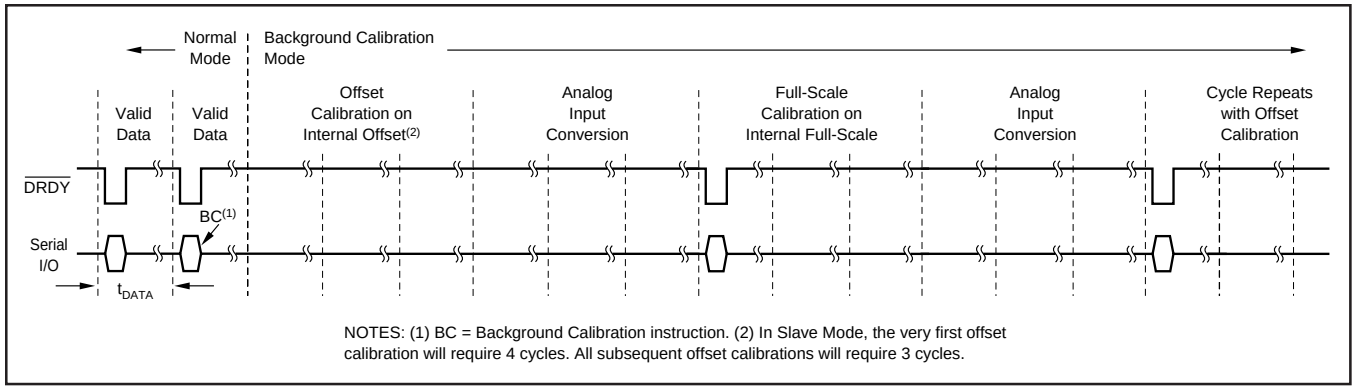
Background Calibration

The Background Calibration Mode is entered after the bits 101 have been written to the Command Register Operation Mode bits (MD2 through MD0). This initiates the following continuous sequence (see Figure 9). At the start of the next conversion cycle, the $\overline{DRDY}$ signal will not go LOW but will remain HIGH. The inputs to the sampling capacitor are disconnected from the converter's analog input and shorted together. An offset calibration is performed over the next three conversion periods (in Slave Mode, the very first offset calibration requires four periods and all subsequent offset calibrations require three periods). Then, the input capacitor is reconnected to the input. Conversions proceed as usual over the next three cycles in order to fill the digital filter. $\overline{DRDY}$ remains HIGH during this time. On the next cycle, the $\overline{DRDY}$ signal goes LOW indicating valid data.



NOTES: (1) PSC = Pseudo System Calibration instruction. (2) In Slave Mode, this function requires 4 cycles.

FIGURE 8. Pseudo System Calibration Timing.



NOTES: (1) BC = Background Calibration instruction. (2) In Slave Mode, the very first offset calibration will require 4 cycles. All subsequent offset calibrations will require 3 cycles.

FIGURE 9. Background Calibration Timing.

Also, during this cycle, the sampling capacitor is disconnected from the converter's analog input and is connected across REF_{IN}. A gain calibration is initiated and proceeds over the next three conversions. After this, the input capacitor is once again connected to the analog input. Conversions proceed as usual over the next three cycles in order to fill the digital filter. $\overline{DRDY}$ remains HIGH during this time. On the next cycle, the $\overline{DRDY}$ signal goes LOW indicating valid data, the input to the sampling capacitor is shorted, and an offset calibration is initiated. At this point, the Background Calibration sequence repeats.

In essence, the Background Calibration Mode performs continuous self-calibration where the offset and gain calibrations are interleaved with regular conversions. Thus, the data rate is reduced by a factor of 6. The advantage is that the converter is continuously adjusting to environmental changes such as ambient or component temperature (due to airflow variations).

The ADS1210/11 will remain in the Background Calibration Mode indefinitely. To move to any other mode, the Command Register Operation Mode bits (MD2 through MD0) must be set to the appropriate values.

System Calibration Offset and Full-Scale Calibration Limits

The System Offset and Full-Scale Calibration range of the ADS1210/11 is limited and is listed in the Specifications Table. The range is specified as:

$$(V_{FS} - |V_{OS}|) < 1.3 \bullet (2 \bullet REF_{IN})/GAIN$$
$$(V_{FS} - |V_{OS}|) > 0.7 \bullet (2 \bullet REF_{IN})/GAIN$$

where V_{FS} is the system full-scale voltage and $|V_{OS}|$ is the absolute value of the system offset voltage. In the following discussion, keep in mind that these voltages are differential voltages.

For example, with the internal reference (2.5V) and a gain of two, the previous equations become (after some manipulation):

$$V_{FS} - 3.25 < V_{OS} < V_{FS} - 1.75$$

If V_{FS} is perfect at 2.5V (positive full-scale), then V_{OS} must be greater than –0.75V and less than 0.75V. Thus, when offset calibration is performed, the positive input can be no more than 0.75V below or above the negative input. If this range is exceeded, the ADS1210/11 may not calibrate properly.

This calculation method works for all gains other than one. For a gain of one and the internal reference (2.5V), the equation becomes:

$$V_{FS} - 6.5 < V_{OS} < V_{FS} - 3.5$$

With a 5V positive full-scale input, V_{OS} must be greater than –1.5V and less than 1.5V. Since the offset represents a common-mode voltage and the input voltage range in a gain of one is 0V to 5V, a common-mode voltage will cause the actual input voltage to possibly go below 0V or above 5V. The specifications also show that for the specifications to be valid, the input voltage must not go below AGND by more than 30mV or above AV_{DD} by more than 30mV.

This will be an important consideration in many systems which use a 2.5V or greater reference, as the input range is constrained by the expected power supply variations. In addition, the expected full-scale voltage will impact the allowable offset voltage (and vice-versa) as the combination of the two must remain within the power supply and ground potentials, regardless of the results obtained via the range calculation shown previously.

There are only two solutions to this constraint: either the system design must ensure that the full-scale and offset voltage variations will remain within the power supply and ground potentials, or the part must be used in a gain of 2 or greater.

SLEEP MODE

The Sleep Mode is entered after the bits 110 have been written to the Command Register Operation Mode bits (MD2 through MD0). This mode is exited by entering a new mode into the MD2-MD0 bits.

The Sleep Mode causes the analog section and a good deal of the digital section to power down. For full analog power down, the V_{BIAS} generator and the internal reference must also be powered down by setting the BIAS and REFO bits in the Command Register accordingly. The power dissipation shown in the Specifications Table is with the internal reference and the V_{BIAS} generator disabled.

To initiate serial communication with the converter while it is in Sleep Mode, one of the following procedures must be used: If $\overline{CS}$ is being used, simply taking $\overline{CS}$ LOW will enable serial communication to proceed normally. If $\overline{CS}$ is not being used (tied LOW) and the ADS1210/11 is in the Master Mode, then a falling edge must be produced on the SDIO line. If SDIO is LOW, the SDIO line must be taken HIGH for $2 \bullet t_{XIN}$ periods (minimum) and then taken LOW. Alternatively, SDIO can be forced HIGH after putting the ADS1210/11 to "sleep" and then taken LOW when the Sleep Mode is to be exited. Finally, if $\overline{CS}$ is not being used (tied LOW) and the ADS1210/11 is in the Slave Mode, then simply sending a normal Instruction Register command will re-establish communication.

Once serial communication is resumed, the Sleep Mode is exited by changing the MD2-MD0 bits to any other mode. When a new mode (other than Sleep) has been entered, the ADS1210/11 will execute a very brief internal power-up sequence of the analog and digital circuitry. Once this has been done, one normal conversion cycle is performed before the new mode is actually entered. At the end of this conversion cycle, the new mode takes effect and the converter will respond accordingly. The $\overline{DRDY}$ signal will remain HIGH through the first conversion cycle. It will also remain HIGH through the second, even if the new mode is the Normal Mode.

If the V_{BIAS} generator and/or the internal reference have been disabled, then they must be manually re-enabled via the appropriate bits in the Command Register. In addition, the internal reference will have to charge the external bypass capacitor(s) and possibly other circuitry. There may also be

considerations associated with V_{BIAS} and the settling of external circuitry. All of these must be taken into account when determining the amount of time required to resume normal operation. The timing diagram shown in Figure 10 does not take into account the settling of external circuitry.

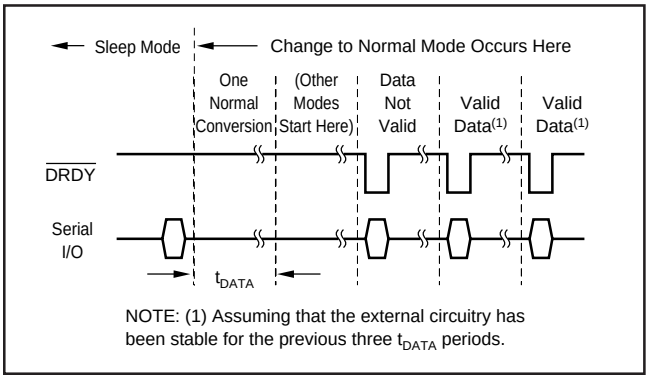


FIGURE 10. Sleep Mode to Normal Mode Timing.

ANALOG OPERATION

ANALOG INPUT

The input impedance of the analog input changes with ADS1210/11 clock frequency (f_{XIN}), gain (G), and Turbo Mode Rate (TMR). The relationship is:

$$A_{IN} \text{ Impedance } (\Omega) = (10MHz/f_{XIN}) \bullet 4.3E6/(G \bullet TMR)$$

Figure 11 shows the basic input structure of the ADS1210. The ADS1211 includes an input multiplexer, but this has little impact on the analysis of the input structure. The impedance is directly related to the sampling frequency of the input capacitor. The X_{IN} clock rate sets the basic sampling rate in a gain of 1 and Turbo Mode Rate of 1. Higher gains and higher Turbo Mode Rates result in an increase of the sampling rate, while slower clock (X_{IN}) frequencies result in a decrease.

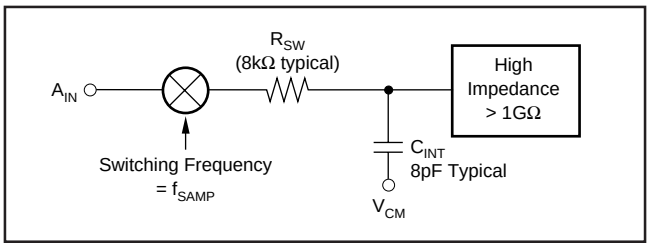


FIGURE 11. Analog Input Structure.

This input impedance can become a major point of consideration in some designs. If the source impedance of the input signal is significant or if there is passive filtering prior to the ADS1210/11, then a significant portion of the signal can be lost across this external impedance. How significant this effect is depends on the desired system performance.

There are two restrictions on the analog input signal to the ADS1210/11. Under no conditions should the current into or out of the analog inputs exceed 10mA. In addition, while the analog signal must reside within this range, the linearity of the ADS1210/11 is only guaranteed when the actual analog input voltage resides within a range defined by AGND –30mV and AV_{DD} +30mV. This is due to leakage paths which occur within the part when AGND and AV_{DD} are exceeded.

For this reason, the 0V to 5V input range (gain of 1 with a 2.5V reference) must be used with caution. Should AV_{DD} be 4.75V, the analog input signal would swing outside of the guaranteed specifications of the device. Designs utilizing this mode of operation should consider limiting the span to a slightly smaller range. Common-mode voltages are also a significant concern in this mode and must be carefully analyzed.

An input voltage range of 0.75V to 4.25V is the smallest span that is allowed if a full system calibration will be performed (see the Calibration section for more details). This also assumes an offset error of zero. A better choice would be 0.5V to 4.5V (a full-scale range of 9V). This span would allow some offset error, gain error, power supply drift, and common-mode voltage while still providing full system calibration over reasonable variation in each of these parameters.

The actual input voltage exceeding AGND or AV_{DD} should not be a concern in higher gain settings as the input voltage range will reside well within 0V to 5V. This is true unless the common-mode voltage is large enough to place positive full-scale or negative full-scale outside of the AGND to AV_{DD} range.

REFERENCE INPUT

The input impedance of the REF_{IN} input changes with clock frequency (f_{XIN}) and Turbo Mode Rate (TMR). The relationship is:

$$REF_{IN} \text{ Impedance } (\Omega) = (10MHz/f_{XIN}) \bullet 1E6/TMR$$

Unlike the analog input, the reference input impedance has a negligible dependency on the PGA gain setting.

The reference input voltage can vary between 2V and 3V. A nominal voltage of 2.5V appears at REF_{OUT}, and this can be directly connected to REF_{IN}. Higher reference voltages will cause the full-scale range to increase while the internal circuit noise of the converter remains approximately the same. This will increase the LSB weight but not the internal noise, resulting in increased signal-to-noise ratio and effective resolution. Likewise, lower reference voltages will decrease the signal-to-noise ratio and effective resolution.

REFERENCE OUTPUT

The ADS1210/11 contains an internal +2.5V reference. Tolerances, drift, noise, and other specifications for this reference are given in the Specification Table. Note that it is not designed to sink or to source more than 1mA of current. In addition, loading the reference with a dynamic or variable load is not recommended. This can result in small changes in reference voltage as the load changes. Finally, for designs approaching or exceeding 20 bits of effective resolution, a low-noise external reference is recommended as the internal reference may not provide adequate performance.

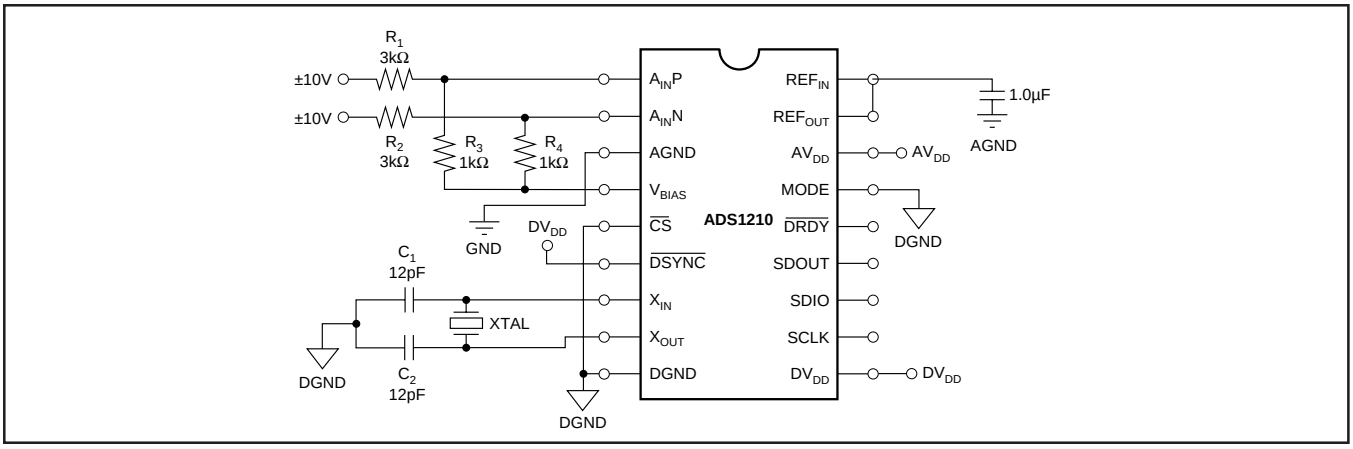


FIGURE 12. ±10V Input Configuration Using V_{BIAS}.

The circuitry which generates the +2.5V reference can be disabled via the Command Register and will result in a lower power dissipation. The reference circuitry consumes a little over 1.6mA of current with no external load. When the ADS1210/11 is in its default state, the internal reference is enabled.

V_{BIAS}

The V_{BIAS} output voltage is dependent on the reference input (REF_{IN}) voltage and is approximately 1.33 times as great. This output is used to bias input signals such that bipolar signals with spans of greater than 5V can be scaled to match the input range of the ADS1210/11. Figure 12 shows a connection diagram which will allow the ADS1210/11 to accept a ±10V input signal (40V full-scale range).

This method of scaling and offsetting the ±20V differential input signal will be a concern for those requiring minimum power dissipation. V_{BIAS} will supply 1.68mA for every channel connected as shown. For the ADS1211, the current draw is within the specifications for V_{BIAS}, but, at 12mW, the power dissipation is significant. If this is a concern, resistors R_1 and R_2 can be set to 9kΩ and R_3 and R_4 to 3kΩ. This will reduce power dissipation by one-third. In addition, these resistors can also be set to values which will provide any arbitrary input range. In all cases, the maximum current into or out of V_{BIAS} should not exceed its specification of 10mA.

Note that the connection diagram shown in Figure 12 causes a constant amount of current to be sourced by V_{BIAS}. This will be very important in higher resolution designs as the voltage at V_{BIAS} will not change with loading, as the load is constant. However, if the input signal is single-ended and one side of the input is grounded, the load will not be constant and V_{BIAS} will change slightly with the input signal. Also, in all cases, note that noise on V_{BIAS} introduces a common-mode error signal which is rejected by the converter.

The 3k resistors should not be used as part of an anti-alias filter with a capacitor across the inputs. The ADS1210 samples charge from the capacitor which has the effect of introducing an offset in the measurement. This might be acceptable for relative differential measurements.

The circuitry to generate V_{BIAS} is disabled when the ADS1210/11 is in its default state, and it must be enabled, via the Command Register, in order for the V_{BIAS} voltage to be present. When enabled, the V_{BIAS} circuitry consumes approximately 1mA with no external load.

On power-up, external signals may be present before V_{BIAS} is enabled. This can create a situation in which a negative voltage is applied to the analog inputs (–2.5V for the circuit shown in Figure 12), reverse biasing the negative input protection diode. This situation should not be a problem as long as the resistors R_1 and R_2 limit the current being sourced by each analog input to under 10mA (a potential of 0V at the analog input pin should be used in the calculation).

DIGITAL OPERATION

SYSTEM CONFIGURATION

The Micro Controller (MC) consists of an ALU and a register bank. The MC has two states: power-on reset and convert. In the power-on reset state, the MC resets all the registers to their default state, sets up the modulator to a stable state, and performs self-calibration at a 850Hz data rate. After this, it enters the convert mode, which is the normal mode of operation for the ADS1210/11.

The ADS1210/11 has 5 internal registers, as shown in Table VII. Two of these, the Instruction Register and the Command Register, control the operation of the converter. The Data Output Register (DOR) contains the result from the most recent conversion. The Offset and Full-Scale Calibration Registers (OCR and FCR) contain data used for correcting the internal conversion result before it is placed into the DOR. The data in these two registers may be the result of a calibration routine, or they may be values which have been written directly via the serial interface.

INSR	Instruction Register	8 Bits
DOR	Data Output Register	24 Bits
CMR	Command Register	32 Bits
OCR	Offset Calibration Register	24 Bits
FCR	Full-Scale Calibration Register	24 Bits

TABLE VII. ADS1210/11 Registers.

Communication with the ADS1210/11 is controlled via the Instruction Register (INSR). Under normal operation, the INSR is written as the first part of each serial communication. The instruction that is sent determines what type of communication will occur next. It is not possible to read the INSR.

The Command Register (CMR) controls all of the ADS1210/11's options and operating modes. These include the PGA gain setting, the Turbo Mode Rate, the output data rate (decimation ratio), etc. The CMR is the only 32-bit register within the ADS1210/11. It, and all the remaining registers, may be read from or written to.

Instruction Register (INSR)

The INSR is an 8-bit register which commands the serial interface either to read or to write "n" bytes beginning at the specified register location. Table VIII shows the format for the INSR.

MSB							LSB
R/$\overline{W}$	MB1	MB0	0	A3	A2	A1	A0

TABLE VIII. Instruction Register.

R/$\overline{W}$ (Read/$\overline{Write}$) Bit—For a write operation to occur, this bit of the INSR must be 0. For a read, this bit must be 1, as follows:

R/$\overline{W}$	
0	Write
1	Read

MB1, MB0 (Multiple Bytes) Bits—These two bits are used to control the word length (number of bytes) of the read or write operation, as follows:

MB1	MB0	
0	0	1 Byte
0	1	2 Bytes
1	0	3 Bytes
1	1	4 Bytes

A3-A0 (Address) Bits—These four bits select the beginning register location which will be read from or written to, as shown in Table IX. Each subsequent byte will be read from or written to the next higher location. (If the BD bit in the Command Register is set, each subsequent byte will be read from the next lower location. This bit does not affect the write operation.) If the next location is not defined in Table IX, then the results are unknown. Reading or writing continues until the number of bytes specified by MB1 and MB0 have been transferred.

A3	A2	A1	A0	REGISTER BYTE
0	0	0	0	Data Output Register Byte 2 (MSB)
0	0	0	1	Data Output Register Byte 1
0	0	1	0	Data Output Register Byte 0 (LSB)
0	1	0	0	Command Register Byte 3 (MSB)
0	1	0	1	Command Register Byte 2
0	1	1	0	Command Register Byte 1
0	1	1	1	Command Register Byte 0 (LSB)
1	0	0	0	Offset Cal Register Byte 2 (MSB)
1	0	0	1	Offset Cal Register Byte 1
1	0	1	0	Offset Cal Register Byte 0 (LSB)
1	1	0	0	Full-Scale Cal Register Byte 2 (MSB)
1	1	0	1	Full-Scale Cal Register Byte 1
1	1	1	0	Full-Scale Cal Register Byte 0 (LSB)
Note: MSB = Most Significant Byte, LSB = Least Significant Byte				

TABLE IX. A3-A0 Addressing.

Each serial communication starts with the 8-bits of the INSR being sent to the ADS1210/11. This directs the remainder of the communication cycle, which consists of n bytes being read from or written to the ADS1210/11. The read/write bit, the number of bytes n, and the starting register address are defined, as shown in Table VIII. When the n bytes have been transferred, the INSR is complete. A new communication cycle is initiated by sending a new INSR (under restrictions outlined in the Interfacing section).

Command Register (CMR)

The CMR controls all of the functionality of the ADS1210/11. The new configuration takes effect on the negative transition of SCLK for the last bit in each byte of data being written to the command register. The organization of the CMR is shown in Table X.

Most Significant Bit — **Byte 3**

BIAS	REFO	DF	U/$\overline{B}$	BD	MSB	SDL	DSYNC(1) / $\overline{DRDY}$	
0 Off	1 On	0 Two's	0 Biplr	0 MSByte	0 MSB	0 SDIO	0	Defaults

NOTE: (1) DSYNC is Write only, $\overline{DRDY}$ is Read only.

Byte 2

MD2 MD1 MD0	G2 G1 G0	CH1 CH0	
000 Normal Mode	000 Gain 1	00 Channel 1	Defaults

Byte 1

SF2 SF1 SF0	DR12 DR11 DR10 DR9 DR8	
000 Turbo Mode Rate of 1	00000	Defaults

Byte 0 Least Significant Bit

DR7 DR6 DR5 DR4 DR3 DR2 DR1 DR0	
(00000) 0001 0111 (23) Data Rate of 814Hz	Defaults

TABLE X. Organization of the Command Register and Default Status.

BIAS (Bias Voltage) Bit—The BIAS bit controls the V_{BIAS} output state—either on (1.33 • REF_{IN}) or off (disabled), as follows:

BIAS	V_{BIAS} GENERATOR	V_{BIAS} STATUS	
0	Off	Disabled	Default
1	On	1.33•REF_{IN}	

The V_{BIAS} circuitry consumes approximately 1mA of steady state current with no external load. See the V_{BIAS} section for full details. When the internal reference (REF_{OUT}) is connected to the reference input (REF_{IN}), V_{BIAS} is 3.3V, nominal.

REFO (Reference Output) Bit—The REFO bit controls the internal reference (REF_{OUT}) state, either on (2.5V) or off (disabled), as follows:

REFO	INTERNAL REFERENCE	REF_{OUT} STATUS	
0	Off	High Impedance	
1	On	2.5V	Default

The internal reference circuitry consumes approximately 1.6mA of steady state current with no external load. See the Reference Output section for full details on the internal reference.

DF (Data Format) Bit—The DF bit controls the format of the output data, either Two's Complement or Offset Binary, as follows:

DF	FORMAT	ANALOG INPUT	DIGITAL OUTPUT	
0	Two's Complement	+Full-Scale Zero –Full Scale	$7FFFFF_H$ 000000_H 800000_H	Default
1	Offset Binary	+Full-Scale Zero –Full-scale	$FFFFFF_H$ 800000_H 000000_H	

These two formats are the same for all bits except the most significant, which is simply inverted in one format vs the other. This bit only applies to the Data Output Register—it has no effect on the other registers.

U/$\overline{B}$ (Unipolar) Bit—The U/$\overline{B}$ bit controls the limits imposed on the output data, as follows:

U/$\overline{B}$	MODE	LIMITS	
0	Bipolar	None	Default
1	Unipolar	Zero to +Full-Scale only	

The particular mode has no effect on the actual full-scale range of the ADS1210/11, data format, or data format vs input voltage. In the bipolar mode, the ADS1210/11 operates normally. In the unipolar mode, the conversion result is limited to positive values only (zero included).

This bit only controls what is placed in the Data Output Register. It has no effect on internal data. When cleared, the very next conversion will produce a valid bipolar result.

BD (Byte Order) Bit—The BD bit controls the order in which bytes of data are read, either most significant byte first or least significant byte, as follows:

BD	BYTE ACCESS ORDER	
0	Most Significant to Least Significant Byte	Default
1	Least Significant to Most Significant Byte	

Note that when BD is clear and a multi-byte read is initiated, A3-A0 of the Instruction Register is the address of the most significant byte and subsequent bytes reside at higher addresses. If BD is set, then A3-A0 is the address of the least significant byte and subsequent bytes reside at lower addresses. The BD bit only affects read operations, it has no affect on write operations.

MSB (Bit Order) Bit—The MSB bit controls the order in which bits within a byte of data are read, either most significant bit first or least significant bit, as follows:

MSB	BIT ORDER	
0	Most Significant Bit First	Default
1	Least Significant Bit First	

The MSB bit only affects read operations, it has no affect on write operations.

SDL (Serial Data Line) Bit—The SDL bit controls which pin on the ADS1210/11 will be used as the serial data output pin, either SDIO or SDOUT, as follows:

SDL	SERIAL DATA OUTPUT PIN	
0	SDIO	Default
1	SDOUT	

If SDL is LOW, then SDIO will be used for both input and output of serial data—see the Timing section for more details on how the SDIO pin transitions between these two states. In addition, SDOUT will remain in a tri-state condition at all times.

Important Note: Since the default condition is SDL LOW, SDIO has the potential of becoming an output once every data output cycle if the ADS1210/11 is in the Master Mode. This will occur until the Command Register can be written and the SDL bit set HIGH. See the Interfacing section for more information.

$\overline{DRDY}$ (Data Ready) Bit—The $\overline{DRDY}$ bit is a read only bit which reflects the state of the ADS1210/11's $\overline{DRDY}$ output pin, as follows:

$\overline{DRDY}$	MEANING
0	Data Ready
1	Data Not Ready

DSYNC (Data Synchronization) Bit—The DSYNC bit is a write only bit which occupies the same location as $\overline{DRDY}$. When a 'one' is written to this location, the affect on the ADS1210/11 is the same as if the $\overline{DSYNC}$ input pin had been taken LOW and returned HIGH. That is, the modulator count for the current conversion cycle will be reset to zero.

DSYNC	MEANING
0	No Change in Modulator Count
1	Modulator Count Reset to Zero

The DSYNC bit is provided in order to reduce the number of interface signals that are needed between the ADS1210/11 and the main controller. Consult "Making Use of DSYNC" in the Serial Interface section for more information.

MD2-MD0 (Operating Mode) Bits—The MD2-MD0 bits initiate or enable the various calibration sequences, as follows:

MD2	MD1	MD0	OPERATING MODE
0	0	0	Normal Mode
0	0	1	Self-Calibration
0	1	0	System Offset Calibration
0	1	1	System Full-Scale Calibration
1	0	0	Pseudo System Calibration
1	0	1	Background Calibration
1	1	0	Sleep
1	1	1	Reserved

The Normal Mode, Background Calibration Mode, and Sleep Mode are permanent modes and the ADS1210/11 will remain in these modes indefinitely. All other modes are temporary and will revert to Normal Mode once the appropriate actions are complete. See the Calibration and Sleep Mode sections for more information.

Data Rate (Hz)	Decimation Ratio	DR12	DR11	DR10	DR9	DR8	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
1000	19	0	0	0	0	0	0	0	0	1	0	0	1	1
500	38	0	0	0	0	0	0	0	1	0	0	1	1	0
250	77	0	0	0	0	0	0	1	0	0	1	1	0	1
100	194	0	0	0	0	0	1	1	0	0	0	0	1	0
60	325	0	0	0	0	1	0	1	0	0	0	1	0	1
50	390	0	0	0	0	1	1	0	0	0	0	1	1	0
20	976	0	0	0	1	1	1	1	0	1	0	0	0	0
10	1952	0	0	1	1	1	1	0	1	0	0	0	0	0

Table XI. Decimation Ratios vs Data Rates (Turbo Mode rate of 1 and 10MHz clock).

G2-G0 (PGA Control) Bits—The G2-G0 bits control the gain setting of the PGA, as follows:

G2	G1	G0	GAIN SETTING	AVAILABLE TURBO MODE RATES	
0	0	0	1	1, 2, 4, 8, 16	Default
0	0	1	2	1, 2, 4, 8	
0	1	0	4	1, 2, 4	
0	1	1	8	1, 2	
1	0	0	16	1	

The gain is partially implemented by increasing the input capacitor sampling frequency, which is given by the following equation:

$$f_{SAMP} = G \bullet TMR \bullet f_{XIN}/512$$

where G is the gain setting and TMR is the Turbo Mode Rate. The product of G and TMR cannot exceed 16. The sampling frequency of the input capacitor directly relates to the analog input impedance. See the Programmable Gain Amplifier and Analog Input sections for more details.

CH1-CH0 (Channel Selection) Bits—The CH1 and CH0 bits control the input multiplexer on the ADS1211, as follows:

CH1	CH0	ACTIVE INPUT	
0	0	Channel 1	Default
0	1	Channel 2	
1	0	Channel 3	
1	1	Channel 4	

(For the ADS1210, CH1 and CH0 must always be zero.) The channel change takes effect when the last bit of byte 2 has been written to the Command Register. Output data will not be valid for the next three conversions despite the $\overline{DRDY}$ signal indicating that data is ready. On the fourth time that $\overline{DRDY}$ goes LOW after a channel change has been written to the Command Register, valid data will be present in the Data Output Register (see Figure 4).

SF2-SF0 (Turbo Mode Rate) Bits—The SF2-SF0 bits control the input capacitor sampling frequency and modulator rate, as follows:

SF2	SF1	SF0	TURBO MODE RATE	AVAILABLE PGA SETTINGS	
0	0	0	1	1, 2, 4, 8, 16	Default
0	0	1	2	1, 2, 4, 8	
0	1	0	4	1, 2, 4	
0	1	1	8	1, 2	
1	0	0	16	1	

The input capacitor sampling frequency and modulator rate can be calculated from the following equations:

$$f_{SAMP} = G \bullet TMR \bullet f_{XIN}/512$$

$$f_{MOD} = TMR \bullet f_{XIN}/512$$

where G is the gain setting and TMR is the Turbo Mode Rate. The sampling frequency of the input capacitor directly relates to the analog input impedance. The modulator rate relates to the power consumption of the ADS1210/11 and the output data rate. See the Turbo Mode, Analog Input, and Reference Input sections for more details.

DR12-DR0 (Decimation Ratio) Bits—The DR12-DR0 bits control the decimation ratio of the ADS1210/11. In essence, these bits set the number of modulator results which are used in the digital filter to compute each individual conversion result. Since the modulator rate depends on both the ADS1210/11 clock frequency and the Turbo Mode Rate, the actual output data rate is given by the following equation:

$$f_{DATA} = f_{XIN} \bullet TMR/(512 \bullet (\text{Decimation Ratio} + 1))$$

where TMR is the Turbo Mode Rate. Table XI shows various data rates and corresponding decimation ratios (with a 10MHz clock). Valid decimation ratios are from 19 to 8000. Outside of this range, the digital filter will compute results incorrectly due to inadequate or too much data.

Data Output Register (DOR)

The DOR is a 24-bit register which contains the most recent conversion result (see Table XII). This register is updated with a new result just prior to $\overline{DRDY}$ going LOW. If the contents of the DOR are not read within a period of time defined by $1/f_{DATA} - 12 \bullet (1/f_{XIN})$, then a new conversion result will overwrite the old. ($\overline{DRDY}$ is forced HIGH prior to the DOR update, unless a read is in progress).

Most Significant Bit			**Byte 2**				
DOR23	DOR22	DOR21	DOR20	DOR19	DOR18	DOR17	DOR16
Byte 1							
DOR15	DOR14	DOR13	DOR12	DOR11	DOR10	DOR9	DOR8
			Byte 0			Least Significant Bit	
DOR7	DOR6	DOR5	DOR4	DOR3	DOR2	DOR1	DOR0

TABLE XII. Data Output Register.

The contents of the DOR can be in Two's Complement or Offset Binary format. This is controlled by the DF bit of the Command Register. In addition, the contents can be limited to unipolar data only with the $\overline{U}$/B bit of the Command Register.

Offset Calibration Register (OCR)

The OCR is a 24-bit register which contains the offset correction factor that is applied to the conversion result before it is placed in the Data Output Register (see Table XIII). In most applications, the contents of this register will be the result of either a self-calibration or a system calibration.

The OCR is both readable and writeable via the serial interface. For applications requiring a more accurate offset calibration, multiple calibrations can be performed, each resulting OCR value read, the results averaged, and a more precise offset calibration value written back to the OCR.

The actual OCR value will change from part-to-part and with configuration, temperature, and power supply. Thus, the actual OCR value for any arbitrary situation cannot be accurately predicted. That is, a given system offset could not be corrected simply by measuring the error externally, computing a correction factor, and writing that value to the OCR. In addition, be aware that the contents of the OCR are not used to directly correct the conversion result. Rather, the correction is a function of the OCR value. This function is linear and two known points can be used as a basis for interpolating intermediate values for the OCR. Consult the Calibration section for more details.

Most Significant Bit			**Byte 2**				
OCR23	OCR22	OCR21	OCR20	OCR19	OCR18	OCR17	OCR16
			Byte 1				
OCR15	OCR14	OCR13	OCR12	OCR11	OCR10	OCR9	OCR8
			Byte 0		Least Significant Bit		
OCR7	OCR6	OCR5	OCR4	OCR3	OCR2	OCR1	OCR0

TABLE XIII. Offset Calibration Register.

The contents of the OCR are in Two's Complement format. This is not affected by the DF bit in the Command Register.

Full-Scale Calibration Register (FCR)

The FCR is a 24-bit register which contains the full-scale correction factor that is applied to the conversion result before it is placed in the Data Output Register (see Table XIV). In most applications, the contents of this register will be the result of either a self-calibration or a system calibration.

Most Significant Bit			**Byte 2**				
FSR23	FSR22	FSR21	FSR20	FSR19	FSR18	FSR17	FSR16
			Byte 1				
FSR15	FSR14	FSR13	FSR12	FSR11	FSR10	FSR9	FSR8
			Byte 0		Least Significant Bit		
FSR7	FSR6	FSR5	FSR4	FSR3	FSR2	FSR1	FSR0

TABLE XIV. Full-Scale Calibration Register.

The FCR is both readable and writable via the serial interface. For applications requiring a more accurate full-scale calibration, multiple calibrations can be performed, each resulting FCR value read, the results averaged, and a more precise calibration value written back to the FCR.

The actual FCR value will change from part-to-part and with configuration, temperature, and power supply. Thus, the actual FCR value for any arbitrary situation cannot be accurately predicted. That is, a given system full-scale error cannot be corrected simply by measuring the error externally, computing a correction factor, and writing that value to the FCR. In addition, be aware that the contents of the FCR are not used to directly correct the conversion result. Rather, the correction is a function of the FCR value. This function is linear and two known points can be used as a basis for interpolating intermediate values for the FCR. Consult the Calibration section for more details. The contents of the FCR are in unsigned binary format. This is not affected by the DF bit in the Command Register.

TIMING

Table XV and Figures 13 through 21 define the basic digital timing characteristics of the ADS1210/11. Figure 13 and the associated timing symbols apply to the X_{IN} input signal. Figures 14 through 20 and associated timing symbols apply to the serial interface signals (SCLK, SDIO, SDOUT, and $\overline{CS}$) and their relationship to $\overline{DRDY}$. The serial interface is discussed in detail in the Serial Interface section. Figure 21 and the associated timing symbols apply to the maximum $\overline{DRDY}$ rise and fall times.

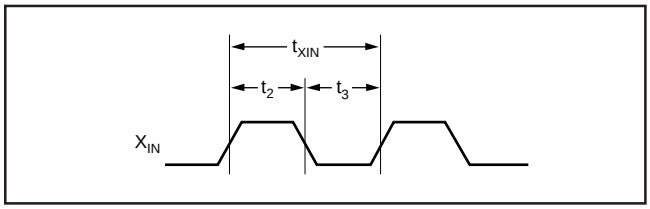


FIGURE 13. X_{IN} Clock Timing.

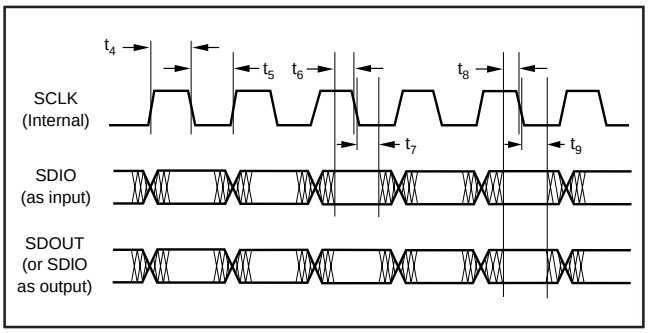


FIGURE 14. Serial Input/Output Timing, Master Mode.

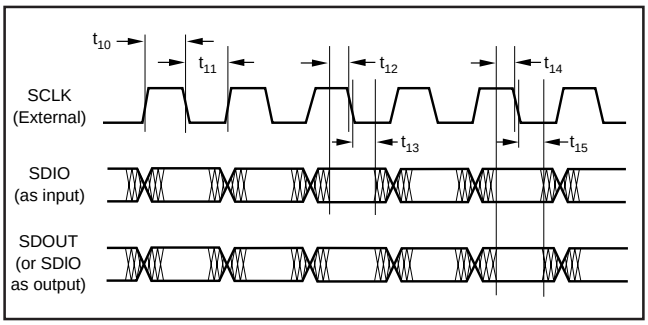


FIGURE 15. Serial Input/Output Timing, Slave Mode.

SYMBOL	DESCRIPTION	MIN	NOM	MAX	UNITS
f_{XIN}	X_{IN} Clock Frequency	0.5		10	MHz
t_{XIN}	X_{IN} Clock Period	100		2000	ns
t_2	X_{IN} Clock High	0.4 • t_{XIN}			ns
t_3	X_{IN} Clock LOW	0.4 • t_{XIN}			ns
t_4	Internal Serial Clock HIGH		t_{XIN}		ns
t_5	Internal Serial Clock LOW		t_{XIN}		ns
t_6	Data In Valid to Internal SCLK Falling Edge (Setup)	40			ns
t_7	Internal SCLK Falling Edge to Data In Not Valid (Hold)	20			ns
t_8	Data Out Valid to Internal SCLK Falling Edge (Setup)	t_{XIN} –25			ns
t_9	Internal SCLK Falling Edge to Data Out Not Valid (Hold)	t_{XIN}			ns
t_{10}	External Serial Clock HIGH	2.5 • t_{XIN}			ns
t_{11}	External Serial Clock LOW	2.5 • t_{XIN}			ns
t_{12}	Data In Valid to External SCLK Falling Edge (Setup)	40			ns
t_{13}	External SCLK Falling Edge to Data In Not Valid (Hold)	20			ns
t_{14}	Data Out Valid to External SCLK Falling Edge (Setup)	t_{XIN} –40			ns
t_{15}	External SCLK Falling Edge to Data Out Not Valid (Hold)	1.5 • t_{XIN}			ns
t_{16}	Falling Edge of $\overline{DRDY}$ to First SCLK Rising Edge (Master Mode, $\overline{CS}$ Tied LOW)		6 • t_{XIN}		ns
t_{17}	Falling Edge of Last SCLK for INSR to Rising Edge of First SCLK for Register Data (Master Mode)		5 • t_{XIN}		ns
t_{18}	Falling Edge of Last SCLK for Register Data to Rising Edge of $\overline{DRDY}$ (Master Mode)		3 • t_{XIN}		ns
t_{19}	Falling Edge of Last SCLK for INSR to Rising Edge of First SCLK for Register Data (Slave Mode)	5.5 • t_{XIN}			ns ns
t_{20}	Falling Edge of Last SCLK for Register Data to Rising Edge of $\overline{DRDY}$ (Slave Mode)	4 • t_{XIN}		5 • t_{XIN}	ns
t_{21}	Falling Edge of $\overline{DRDY}$ to Falling Edge of $\overline{CS}$ (Master and Slave Mode)	0.5 • t_{XIN}			ns
t_{22}	Falling Edge of $\overline{CS}$ to Rising Edge of SCLK (Master Mode)	5 • t_{XIN}		6 • t_{XIN}	ns
t_{23}	Rising Edge of $\overline{DRDY}$ to Rising Edge of $\overline{CS}$ (Master and Slave Mode)	10			ns
t_{24}	Falling Edge of $\overline{CS}$ to Rising Edge of SCLK (Slave Mode)	5.5 • t_{XIN}			ns
t_{25}	Falling Edge of Last SCLK for INSR to SDIO Tri-state (Master Mode)		2 • t_{XIN}		ns
t_{26}	SDIO as Output to Rising Edge of First SCLK for Register Data (Master and Slave Modes)		2 • t_{XIN}		ns
t_{27}	Falling Edge of Last SCLK for INSR to SDIO Tri-state (Slave Mode)	3 • t_{XIN}		4 • t_{XIN}	ns
t_{28}	SDIO Tri-state Time (Master and Slave Modes)		t_{XIN}		ns
t_{29}	Falling Edge of Last SCLK for Register Data to SDIO Tri-State (Master Mode)		t_{XIN}		ns
t_{30}	Falling Edge of Last SCLK for Register Data to SDIO Tri-state (Slave Mode)	2 • t_{XIN}		3 • t_{XIN}	ns
t_{31}	$\overline{DRDY}$ Fall Time			30	ns
t_{32}	$\overline{DRDY}$ Rise Time			30	ns
t_{33}	Minimum $\overline{DSYNC}$ LOW Time	10.5 • t_{XIN}			ns
t_{34}	$\overline{DSYNC}$ Valid HIGH to Falling Edge of X_{IN} (for Exact Synchronization of Multiple Converters only)	10			ns
t_{35}	Falling Edge of X_{IN} to $\overline{DSYNC}$ Not Valid LOW (for Exact Synchronization of Multiple Converters only)	10			ns
t_{36}	Falling Edge of Last SCLK for Register Data to Rising Edge of First SCLK of next INSR (Slave Mode, $\overline{CS}$ Tied LOW)	20.5 • t_{XIN}			ns
t_{37}	Rising Edge of $\overline{CS}$ to Falling Edge of $\overline{CS}$ (Slave Mode, Using $\overline{CS}$)	10.5 • t_{XIN}			ns
t_{38}	Falling Edge of $\overline{DRDY}$ to First SCLK Rising Edge (Slave Mode, $\overline{CS}$ Tied LOW)	5.5 • t_{XIN}			ns

TABLE XV. Digital Timing Characteristics.

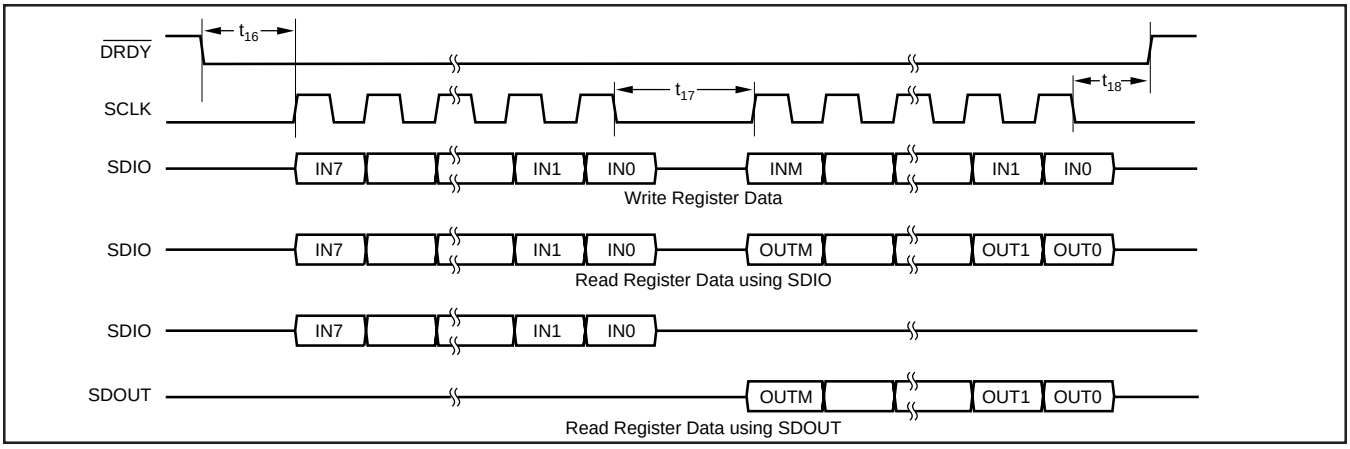


FIGURE 16. Serial Interface Timing ($\overline{CS}$ LOW), Master Mode.

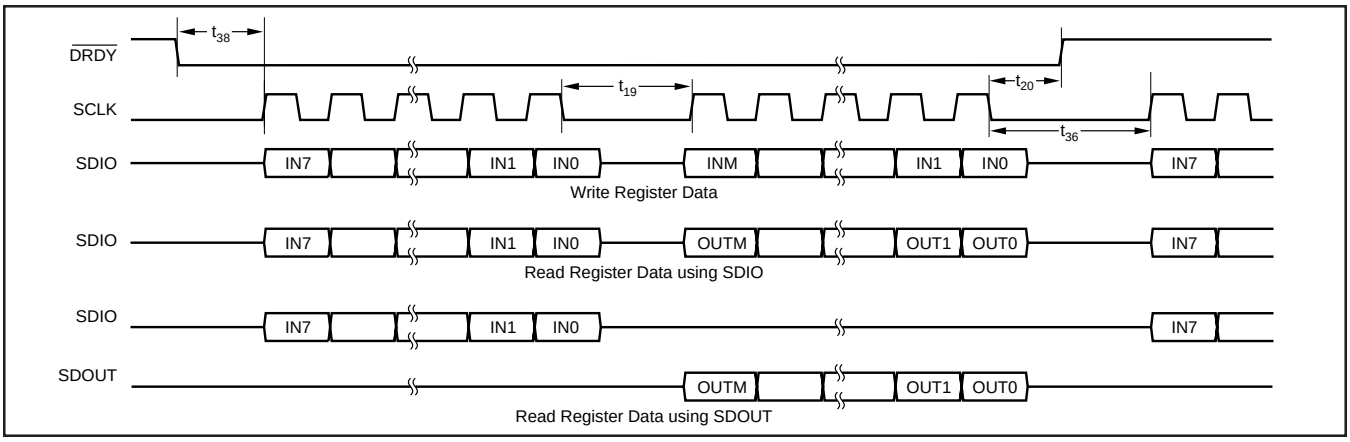


FIGURE 17. Serial Interface Timing ($\overline{CS}$ LOW), Slave Mode.

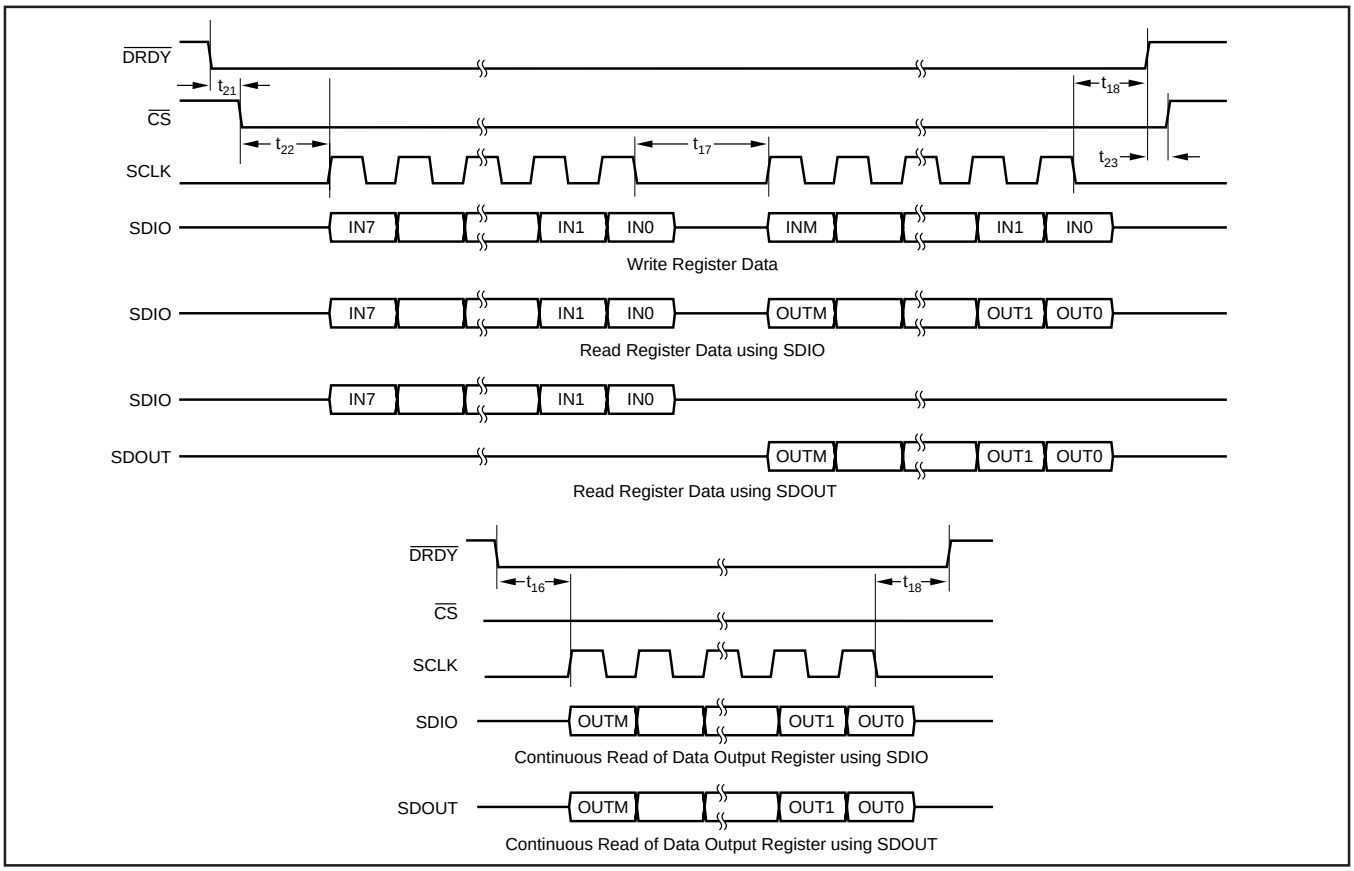


FIGURE 18. Serial Interface Timing (Using $\overline{CS}$), Master Mode.

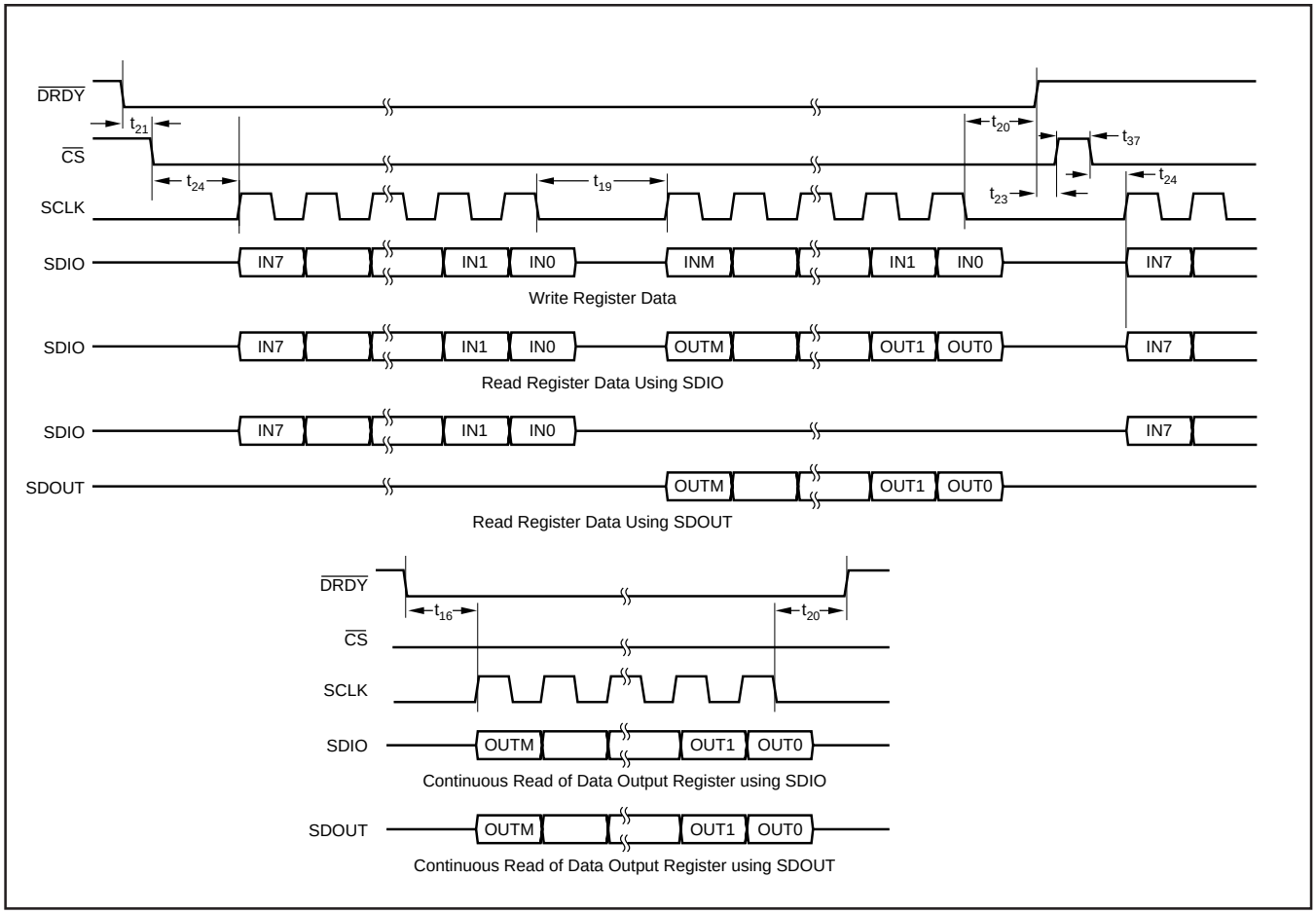


FIGURE 19. Serial Interface Timing (Using $\overline{CS}$), Slave Mode.

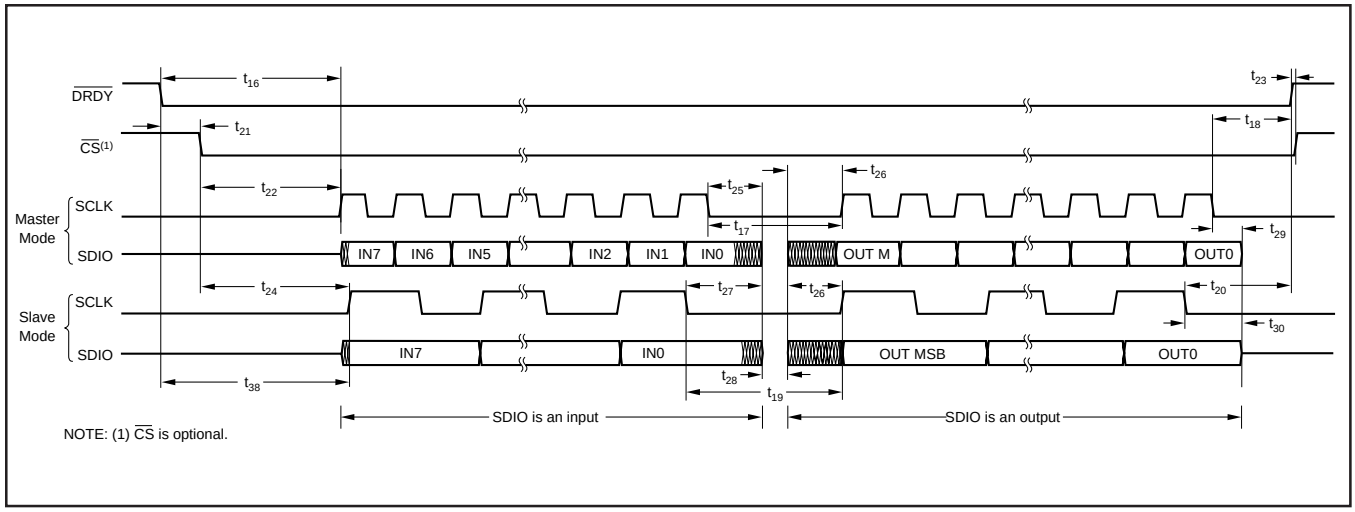


FIGURE 20. SDIO Input to Output Transition Timing.

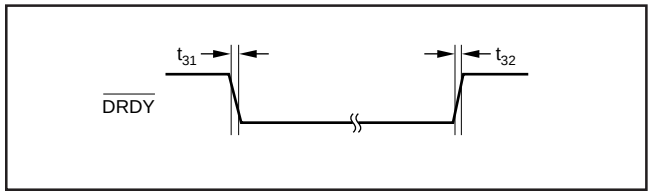


FIGURE 21. $\overline{DRDY}$ Rise and Fall Time.

Synchronizing Multiple Converters

A negative going pulse on $\overline{DSYNC}$ can be used to synchronize multiple ADS1210/11s. This assumes that each ADS1210 is driven from the same master clock and is set to the same Decimation Ratio and Turbo Mode Rate. The affect that this signal has on data output timing in general is discussed in the Serial Interface section.

The concern here is what happens if the $\overline{DSYNC}$ input is completely asynchronous to this master clock. If the $\overline{DSYNC}$ input rises at a critical point in relation to the master clock input, then some ADS1210/11s may start-up one X_{IN} clock cycle before the others. Thus, the output data will be synchronized, but only to within one X_{IN} clock cycle.

For many applications, this will be more than adequate. In these cases, the timing symbols which relate the $\overline{DSYNC}$ signal to the X_{IN} signal can be ignored. For other multiple-converter applications, this one X_{IN} clock cycle difference could be a problem. These types of applications would include using the $\overline{DRDY}$ and/or the SCLK output from one ADS1210/11 as the "master" signal for all converters.

To ensure exact synchronization to the same X_{IN} edge, the timing relationship between the $\overline{DSYNC}$ and X_{IN} signals, as shown in Figure 22, must be observed. Figure 23 shows a simple circuit which can be used to clock multiple ADS1210/11s from one ADS1210/11, as well as to ensure that an asynchronous $\overline{DSYNC}$ signal will exactly synchronize all the converters.

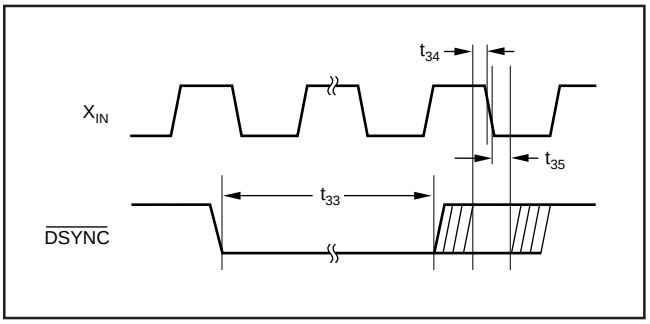


FIGURE 22. $\overline{DSYNC}$ to X_{IN} Timing for Synchronizing Mutliple ADS1210/11s.

SERIAL INTERFACE

The ADS1210/11 includes a flexible serial interface which can be connected to microcontrollers and digital signal processors in a variety of ways. Along with this flexibility, there is also a good deal of complexity. This section describes the trade-offs between the different types of interfacing methods in a top-down approach—starting with the overall flow and control of serial data, moving to specific interface examples, and then providing information on various issues related to the serial interface.

Multiple Instructions

The general timing diagrams which appear throughout this data sheet show serial communication to and from the ADS1210/11 occurring during the $\overline{DRDY}$ LOW period (see Figures 4 through 10 and Figure 36). This communication represents one instruction that is executed by the ADS1210/11, resulting in a single read or write of register data.

However, more than one instruction can be executed by the ADS1210/11 during any given conversion period (see Figure 24). Note that $\overline{DRDY}$ remains HIGH during the subsequent instructions. There are several important restrictions on how and when multiple instructions can be issued during any one conversion period.

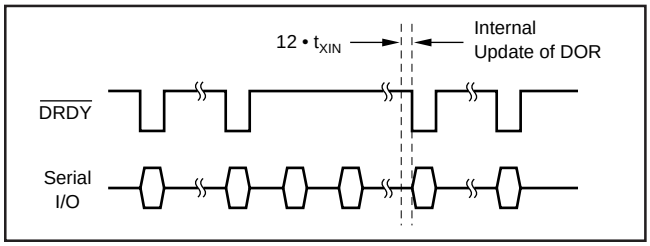


FIGURE 24. Timing of Data Output Register Update.

The first restriction is that the converter must be in the Slave Mode. There is no provision for multiple instructions when the ADS1210/11 is operating in the Master Mode. The second is that some instructions will produce invalid results if started at the end of one conversion period and carried into the start of the next conversion period.

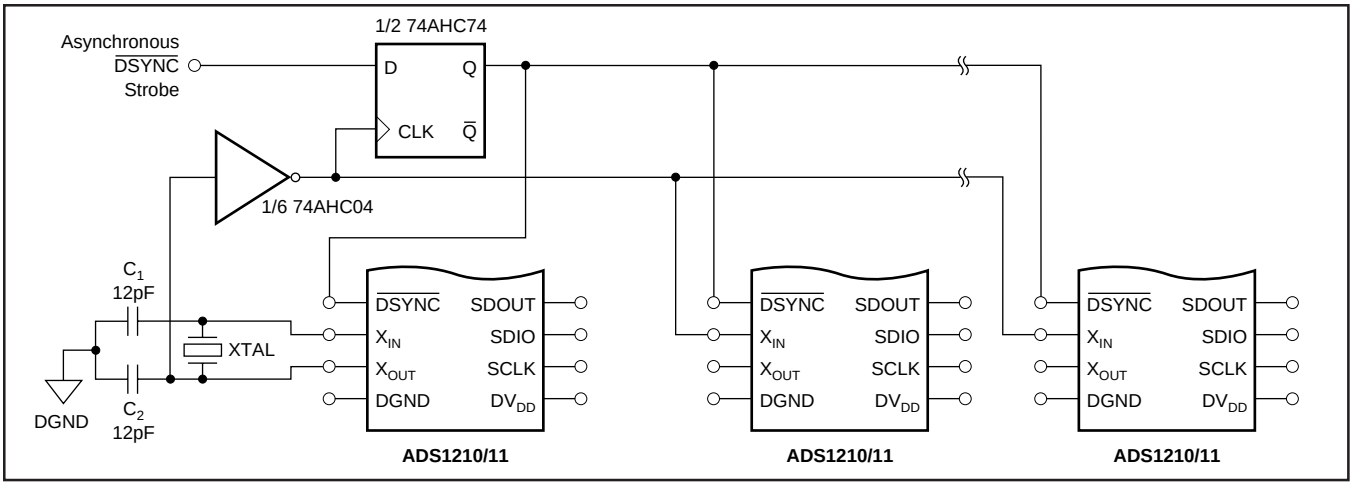


FIGURE 23. Exactly Synchronizing Multiple ADS1210/11s to an Asynchronous $\overline{DSYNC}$ Signal.

For example, Figure 24 shows that just prior to the $\overline{DRDY}$ signal going LOW, the internal Data Output Register (DOR) is updated. This update involves the Offset Calibration Register (OCR) and the Full-Scale Register (FSR). If the OCR or FSR are being written, their final value may not be correct, and the result placed into the DOR will certainly not be valid. Problems can also arise if certain bits of the Command Register are being changed.

Note that reading the Data Output Register is an exception. If the DOR is being read when the internal update is initiated, the update is blocked. The old output data will remain in the DOR and the new data will be lost. The old data will remain valid until the read operation has completed. In general, multiple instructions may be issued, but the last one in any conversion period should be complete within 12 • X_{IN} clock periods of the next $\overline{DRDY}$ LOW time. In this usage, "complete" refers to the point where $\overline{DRDY}$ rises in Figures 17 and 19 (in the Timing Section). Consult Figures 25 and 26 for the flow of serial data during any one conversion period.

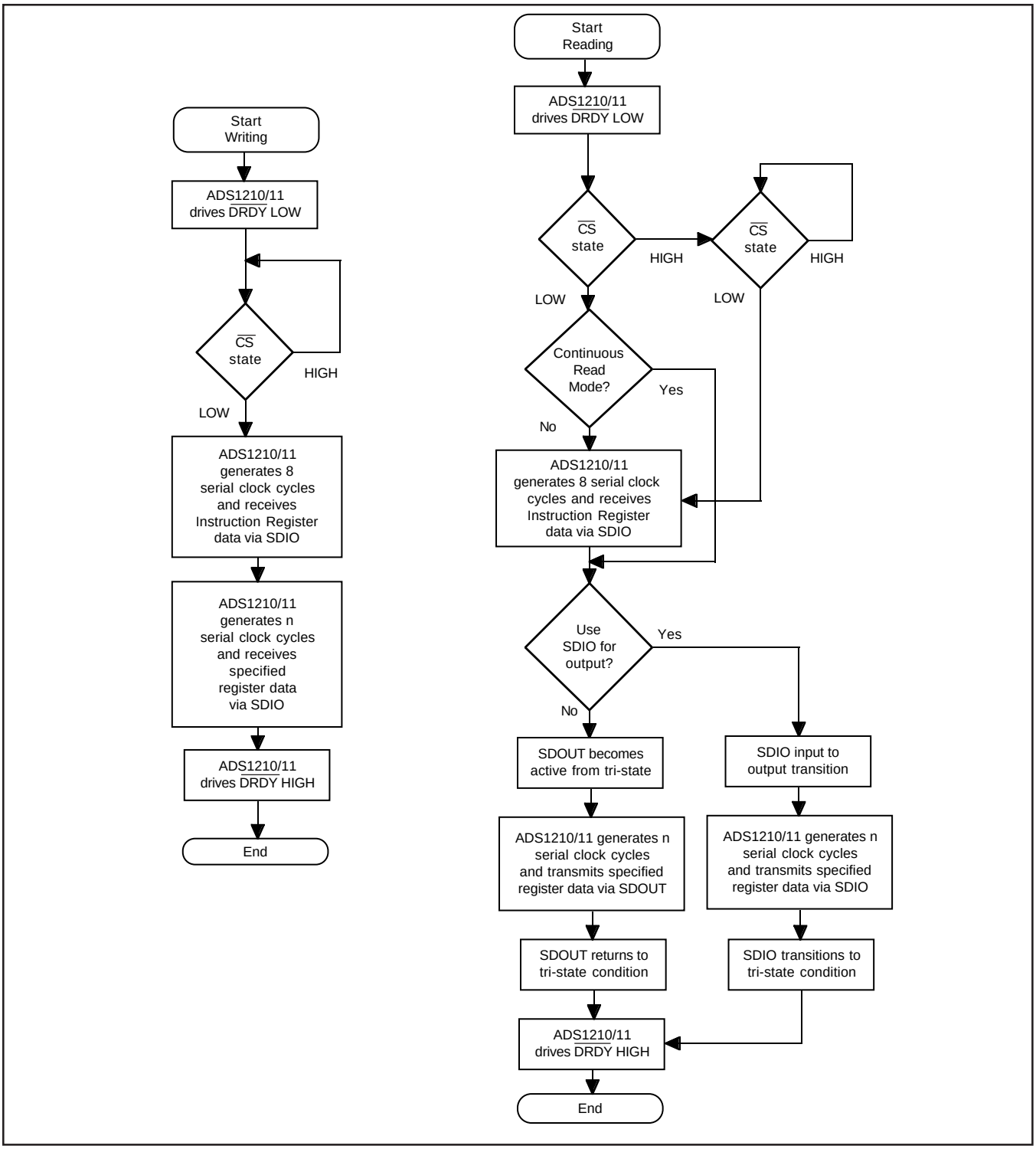


FIGURE 25. Flowchart for Writing and Reading Register Data, Master Mode.

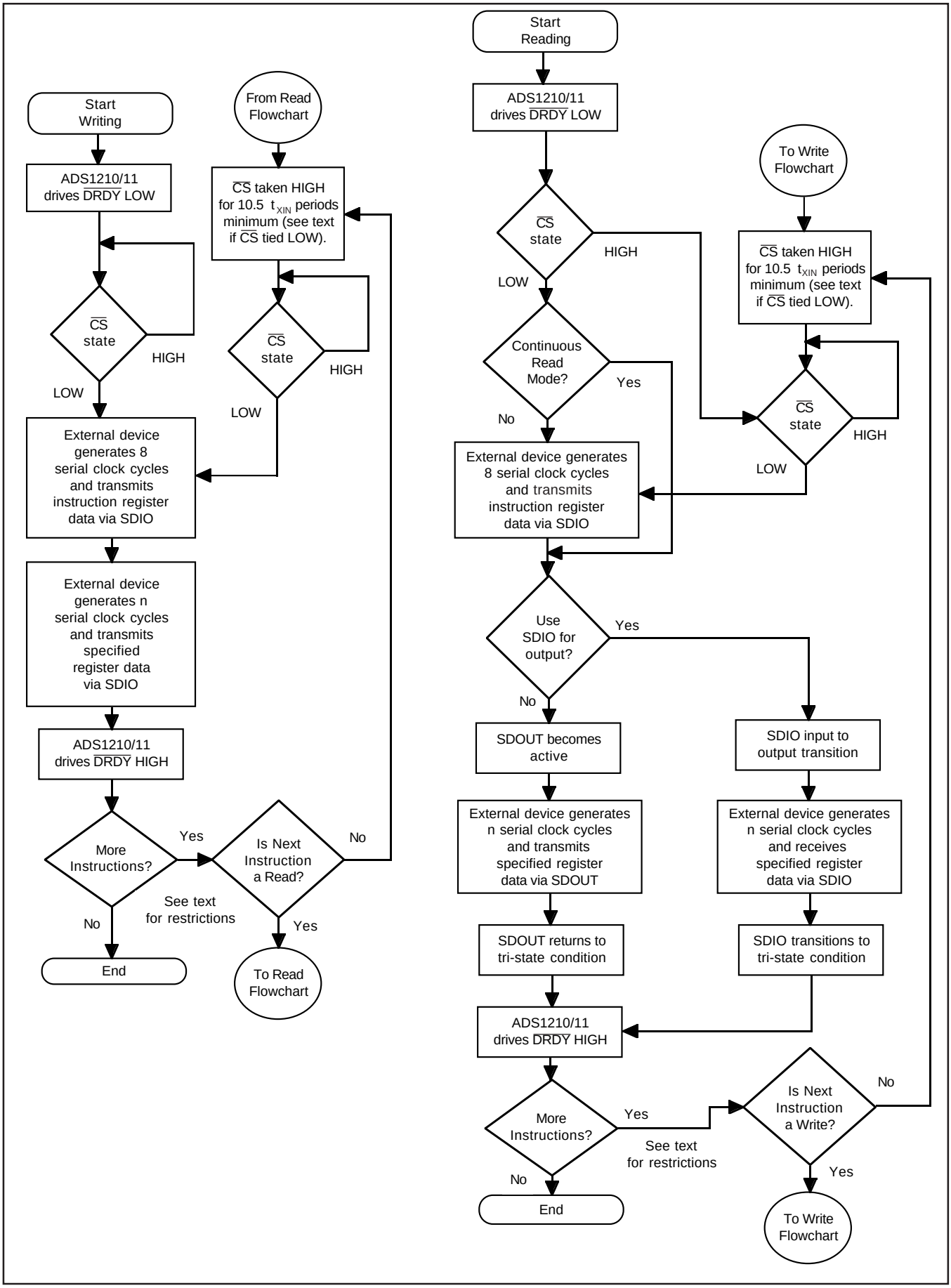


FIGURE 26. Flowchart for Writing and Reading Register Data, Slave Mode.

Using $\overline{CS}$ and Continuous Read Mode

The serial interface may make use of the $\overline{CS}$ signal, or this input may simply be tied LOW. There are several issues associated with choosing to do one or the other.

The $\overline{CS}$ signal does not directly control the tri-state condition of the SDOUT or SDIO output. These signals are normally in the tri-state condition. They only become active when serial data is being transmitted from the ADS1210/11. If the ADS1210/11 is in the middle of a serial transfer and SDOUT or SDIO is an output, taking $\overline{CS}$ HIGH will not tri-state the output signal.

If there are multiple serial peripherals utilizing the same serial I/O lines and communication may occur with any peripheral at any time, then the $\overline{CS}$ signal must be used. The ADS1210/11 may be in the Master Mode or the Slave Mode. In the Master Mode, the $\overline{CS}$ signal is used to hold-off serial communication with a "ready" ($\overline{DRDY}$ LOW) ADS1210/11 until the main controller can accommodate the communication. In the Slave Mode, the $\overline{CS}$ signal is used to enable communication with the ADS1210/11.

The $\overline{CS}$ input has another use. If the $\overline{CS}$ state is left LOW after a read of the Data Output Register has been performed, then the next time that $\overline{DRDY}$ goes LOW, the ADS1210/11 Instruction Register will not be entered. Instead, the Instruction Register contents will be re-used, and the new contents of the Data Output Register, or some part thereof, will be transmitted. This will occur as long as $\overline{CS}$ is LOW and not toggled.

This mode of operation is called the Continuous Read Mode and is shown in the read flowcharts of Figures 25 and 26. It is also shown in the Timing Diagrams of Figures 18 and 19 in the Timing section. Note that once $\overline{CS}$ has been taken HIGH, the Continuous Read Mode will be enabled (but not entered) and can never be disabled. The mode is actually entered and exited as described above.

Power-On Conditions for SDIO

Even if the SDIO connection will be used only for input, there is one important item to consider regarding SDIO. This only applies when the ADS1210/11 is in the Master Mode and $\overline{CS}$ will be tied LOW. At power-up, the serial I/O lines of most microcontrollers and digital signal processors will be in a tri-state condition, or they will be configured as inputs. When power is applied to the ADS1210/11, it will begin operating as defined by the default condition of the Command Register (see Table X in the System Configuration section). This condition defines SDIO as the data output pin.

Since the ADS1210/11 is in the Master Mode and $\overline{CS}$ is tied LOW, the serial clock will run whenever $\overline{DRDY}$ is LOW and an instruction will be entered and executed. If the SDIO line is HIGH, as it might be with an active pull-up, then the instruction is a read operation and SDIO will become an output every DRDY LOW period—for 32 serial clock cycles. When the serial port on the main controller is enabled, signal contention could result.

The recommended solution to this problem is to actively pull SDIO LOW. If SDIO is LOW when the ADS1210/11 enters the instruction byte, then the resulting instruction is a write of one byte of data to the Data Output Register, which results in no internal operation.

If the SDIO signal cannot be actively pulled LOW, then another possibility is to time the initialization of the controller's serial port such that it becomes active between adjacent $\overline{DRDY}$ LOW periods. The default configuration for the ADS1210/11 produces a data rate of 814Hz—a conversion period of 1.2ms. This time should be more than adequate for most microcontrollers and DSPs to monitor $\overline{DRDY}$ and initialize the serial port at the appropriate time.

Master Mode

The Master Mode is active when the MODE input is HIGH. All serial clock cycles will be produced by the ADS1210/11 in this mode, and the SCLK pin is configured as an output. The frequency of the serial clock will be one-half of the X_{IN} frequency. Multiple instructions cannot be issued during a single conversion period in this mode—only one instruction per conversion cycle is possible.

The Master Mode will be difficult for some microcontrollers, particularly when the X_{IN} input frequency is greater than a few MHz, as the serial clock may exceed the microcontroller's maximum serial clock frequency. For the majority of digital signal processors, this will be much less of a concern. In addition, if SDIO is being used as an input and an output, then the transition time from input to output may be a concern. This will be true for both microcontrollers and DSPs. See Figure 20 in the Timing section.

Note that if $\overline{CS}$ is tied LOW, there are special considerations regarding SDIO as outlined previously in this section. Also note that if $\overline{CS}$ is being used to control the flow of data from the ADS1210/11 and it remains HIGH for one or more conversion periods, the ADS1210/11 will operate properly. However, the result in the Data Output Register will be lost when it is overwritten by each new result. Just prior to this update, $\overline{DRDY}$ will be forced HIGH and will return LOW after the update.

Slave Mode

Most systems will use the ADS1210/11 in the Slave Mode. This mode allows multiple instructions to be issued per conversion period as well as allowing the main controller to set the serial clock frequency and pace the serial data transfer. The ADS1210/11 is in the Slave Mode when the MODE input is LOW.

There are several important items regarding the serial clock for this mode of operation. The maximum serial clock frequency cannot exceed the ADS1210/11 X_{IN} frequency divided by 5 (see Figure 15 in the Timing section).

When using SDIO as the serial output, the falling edge of the last serial clock cycle of the instruction byte will cause the SDIO pin to begin its transition from input to output. Between three and four X_{IN} cycles after this falling edge, the SDIO pin will become an output. This transition may be too fast for some microcontrollers and digital signal processors.

If a serial communication does not occur during any conversion period, the ADS1210/11 will continue to operate properly. However, the results in the Data Output Register will be lost when they are overwritten by the new result at the start of the next conversion period. Just prior to this update, $\overline{DRDY}$ will be forced HIGH and will return LOW after the update.

Making Use of $\overline{DSYNC}$

The $\overline{DSYNC}$ input pin and the DSYNC write bit in the Command Register reset the current modulator count to zero. This causes the current conversion cycle to proceed as normal, but all modulator outputs from the last data output to the point where DSYNC is asserted are discarded. Note that the previous two data outputs are still present in the ADS1210/11 internal memory. Both will be used to compute the next conversion result, and the most recent one will be used to compute the result two conversions later. DSYNC does not reset the internal data to zero.

There are two main uses of DSYNC. In the first case, DSYNC allows for synchronization of multiple converters. In regards to the $\overline{DSYNC}$ input pin, this case was discussed under "Synchronizing Multiple Converters" in the Timing section. In regards to the DSYNC bit, it will be difficult to set all of the converter's DSYNC bits at the same time unless all of the converters are in the Slave Mode and the same instruction can be sent to all of the converters at the same time.

The second use of DSYNC is to reset the modulator count to zero in order to obtain valid data as quickly as possible. For example, if the input channel is changed on the ADS1211, the current conversion cycle will be a mix of the old channel and the new channels. Thus, four conversions are needed in order to ensure valid data. However, if the channel is changed and then DSYNC is used to reset the modulator count, the modulator data at the end of the current conversion cycle will be entirely from the new channel. After two additional conversion cycles, the output data will be completely valid. Note that the conversion cycle in which DSYNC is used will be slightly longer than normal. Its length will depend on when DSYNC was set.

Reset, Power-On Reset, and Brown-Out

The ADS1210/11 contains an internal power-on reset circuit. If the power supply ramp rate is greater than 50mV/ms, this circuit will be adequate to ensure that the device powers up correctly. (Due to oscillator settling considerations, communication to and from the ADS1210/11 should not occur for at least 25ms after power is stable.)

If this requirement cannot be met or if the circuit has brown-out considerations, the timing diagram of Figure 27 can be used to reset the ADS1210/11. This timing applies only when the ADS1210/11 is in the Slave Mode and accomplishes the reset by controlling the duty cycle of the SCLK input. In general, a reset is required after power-up, after a brown-out has been detected, or when a watchdog timer event has occured.

If the ADS1210/11 is in the Master Mode, a reset of the device is not possible. If the power supply does not meet the minimum ramp rate requirement, or brown-out is of concern, low on-resistance MOSFETs or equivalent should be used to control power to the ADS1210/11. When powered down, the device should be left unpowered for at least 300ms before power is reapplied. An alternate method would be to control the MODE pin and temporarily place the ADS1210/11 in the Slave Mode while a reset is initiated as shown in Figure 27.

Two-Wire Interface

For a two-wire interface, the Master Mode of operation may be preferable. In this mode, serial communication occurs only when data is ready, informing the main controller as to the status of the ADS1210/11. The disadvantages are that the ADS1210/11 must have a dedicated serial port on the main controller, only one instruction can be issued per data ready period, and the serial clock may define the maximum clock frequency of the converter.

In the Slave Mode, the main controller must read and write to the ADS1210/11 "blindly". Writes to the internal registers, such as the Command Register or Offset Calibration Register, might occur during an update of the Data Output Register. This can result in invalid data in the DOR. A two-wire interface can be used if the main controller can read and/or write to the converter, either much slower or much faster that the data rate. For example, if much faster, the main controller can use the $\overline{DRDY}$ bit to determine when data is becoming valid (polling it multiple times during one conversion cycle). Thus, the controller obtains some idea of when to write to the internal register. If much slower, then reads of the DOR might always return valid data (multiple conversions have occurred since the last read of the DOR or since any write of the internal registers).

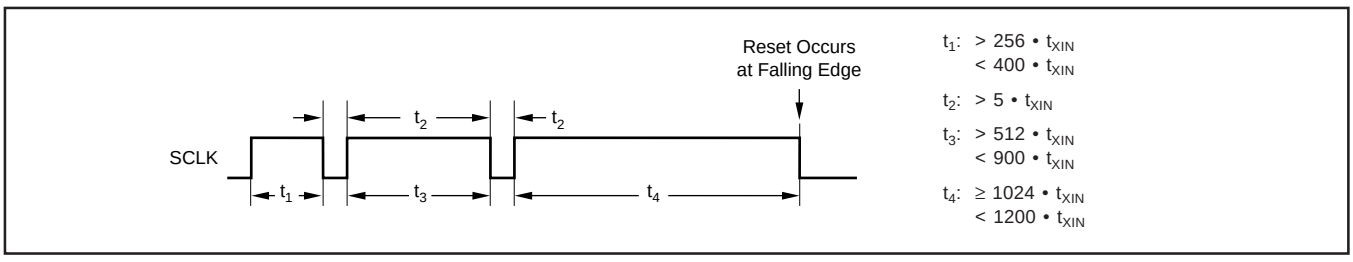


FIGURE 27. Resetting the ADS1210/11 (Slave Mode only).

Three-Wire Interface

Figure 28 shows a three-wire interface with a 8xC32 microprocessor. Note that the Slave Mode is being selected and the SDIO pin is being used for input and output.

Figure 29 shows a different type of three-wire interface with a 8xC51 microprocessor. Here, the Master Mode is used. The interface signals consist of SDOUT, SDIO, and SCLK.

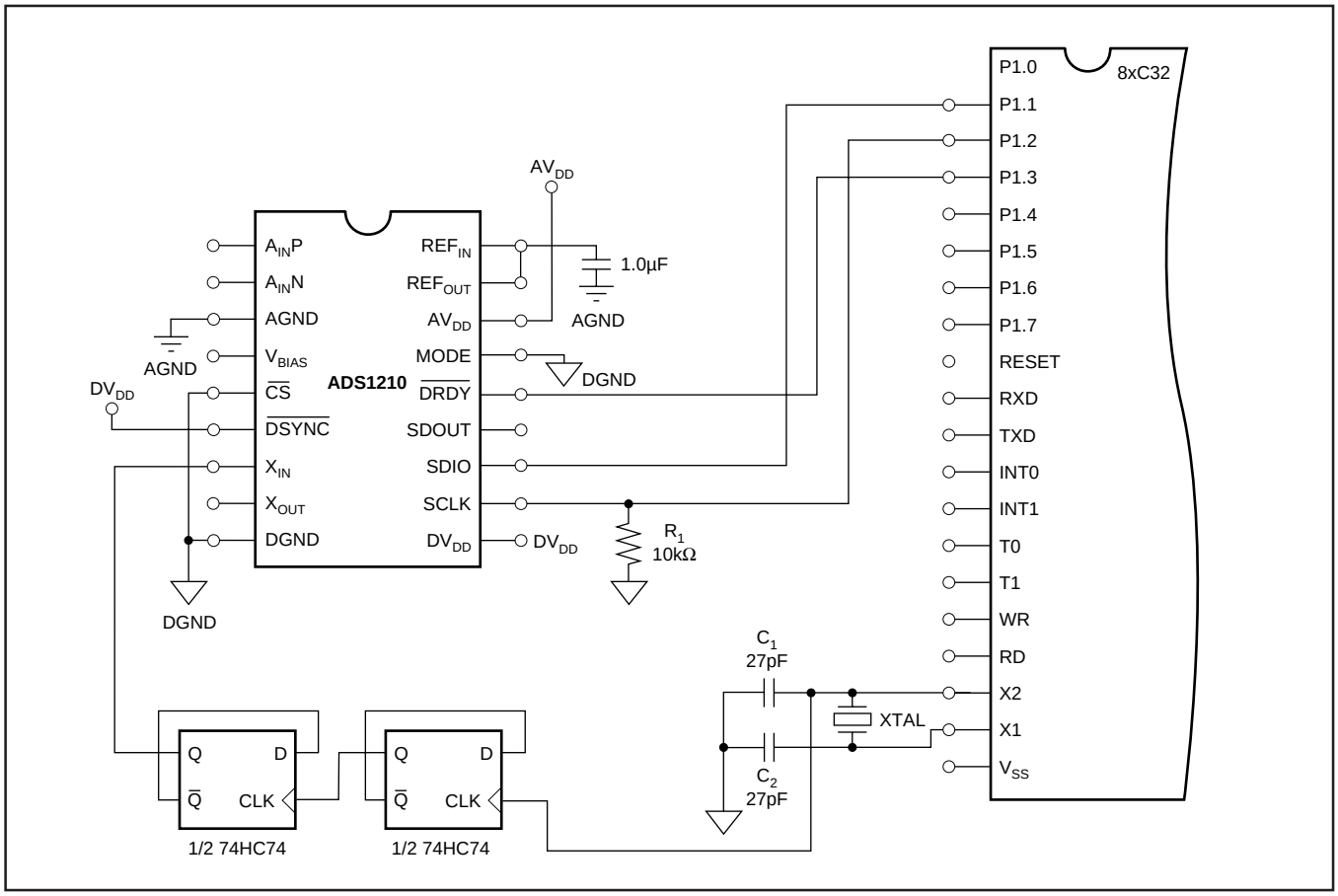


FIGURE 28. Three-Wire Interface with a 8xC32 Microprocessor.

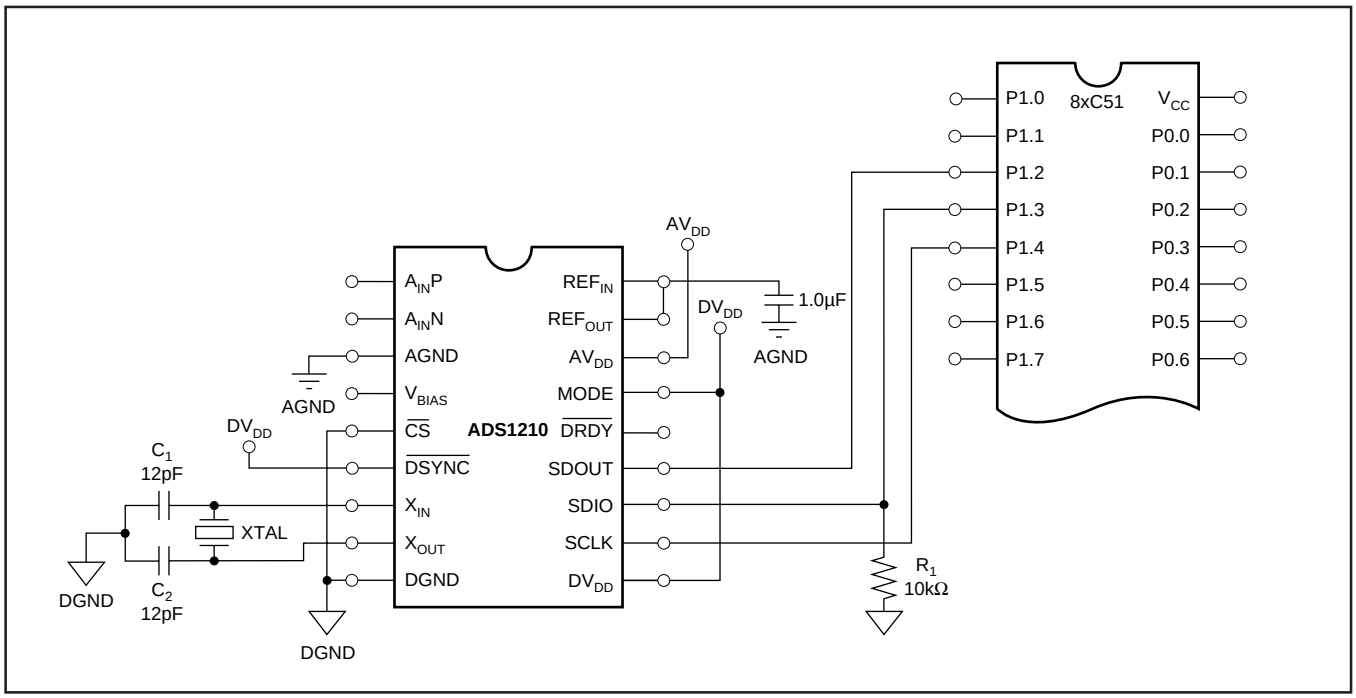


FIGURE 29. Three-Wire Interface with a 8xC51 Microprocessor.

Four-wire Interface

Figure 30 shows a four-wire interface with a 8xC32 microprocessor. Again, the Slave Mode is being used.

Multi-wire Interface

Figures 31 and 32 show multi-wire interfaces with a 8xC51 or 68HC11 microprocessor. In these interfaces, the mode of the ADS1210/11 is actually controlled dynamically. This could be extremely useful when the ADS1210/11 is to be used in a wide variety of ways. For example, it might be desirable to have the ADS1210/11 produce data at a steady rate and to have the converter operating in the Continuous Read Mode. But for system calibration, the Slave Mode might be preferred because multiple instructions can be issued per conversion period.

Note that the MODE input should not be changed in the middle of a serial transfer. This could result in misoperation of the device. A Master/Slave Mode change will not affect the output data.

Note that the X_{IN} input can also be controlled. It is possible with some microcontrollers and digital signal processors to produce a continuous serial clock, which could be connected to the X_{IN} input. The frequency of the clock is often settable over some range. Thus, the power dissipation of the ADS1210/11 could be dynamically varied by changing both the Turbo Mode and X_{IN} input-trading off conversion speed and resolution for power consumption.

I/O Recovery

If serial communication stops during an instruction or data transfer for longer than $4 \cdot t_{DATA}$, the ADS1210/11 will reset its serial interface. This will not affect the internal registers. The main controller must not continue the transfer after this event, but must restart the transfer from the beginning.

This feature is very useful if the main controller can be reset at any point. After reset, simply wait $8 \cdot t_{DATA}$ before starting serial communication.

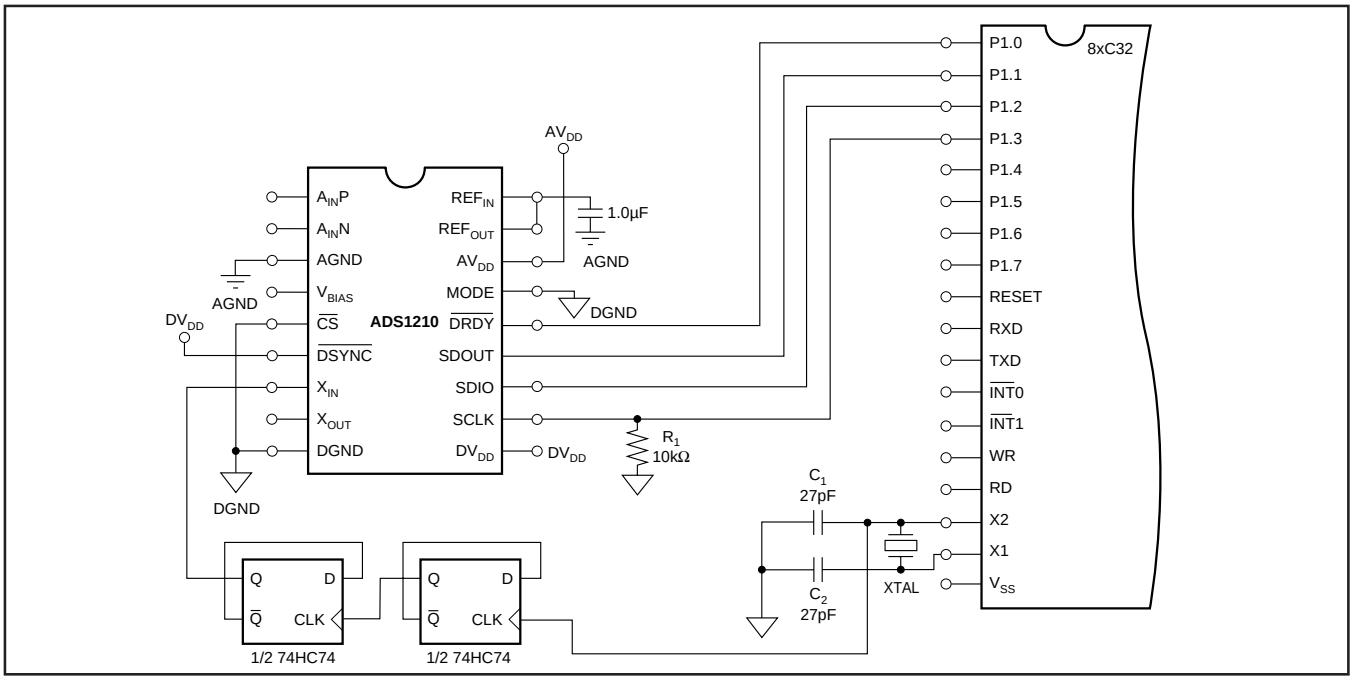


FIGURE 30. Four-Wire Interface with a 8xC32 Microprocessor.

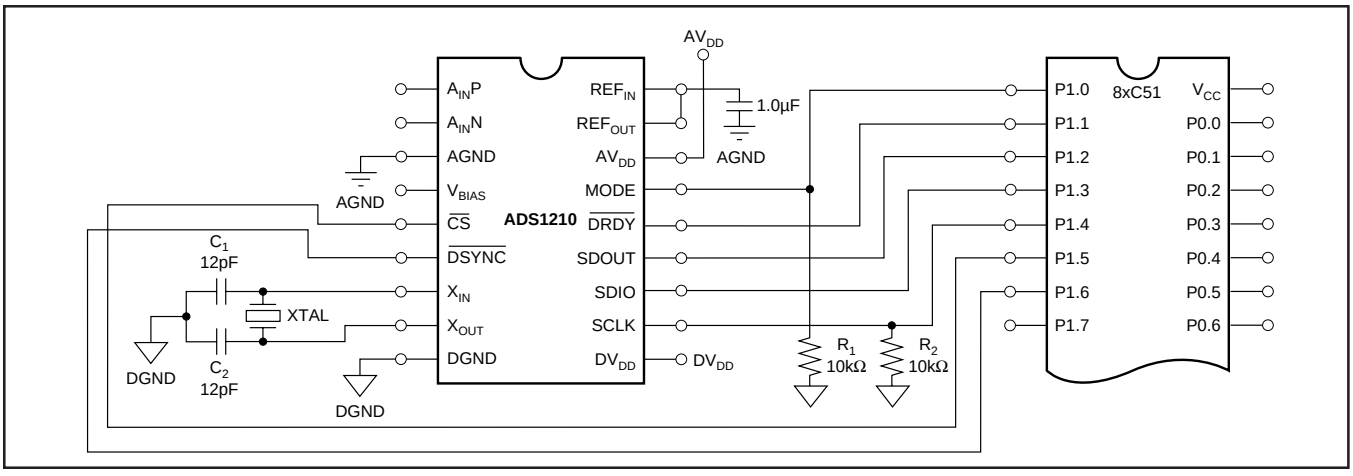


FIGURE 31. Full Interface with a 8xC51 Microprocessor.

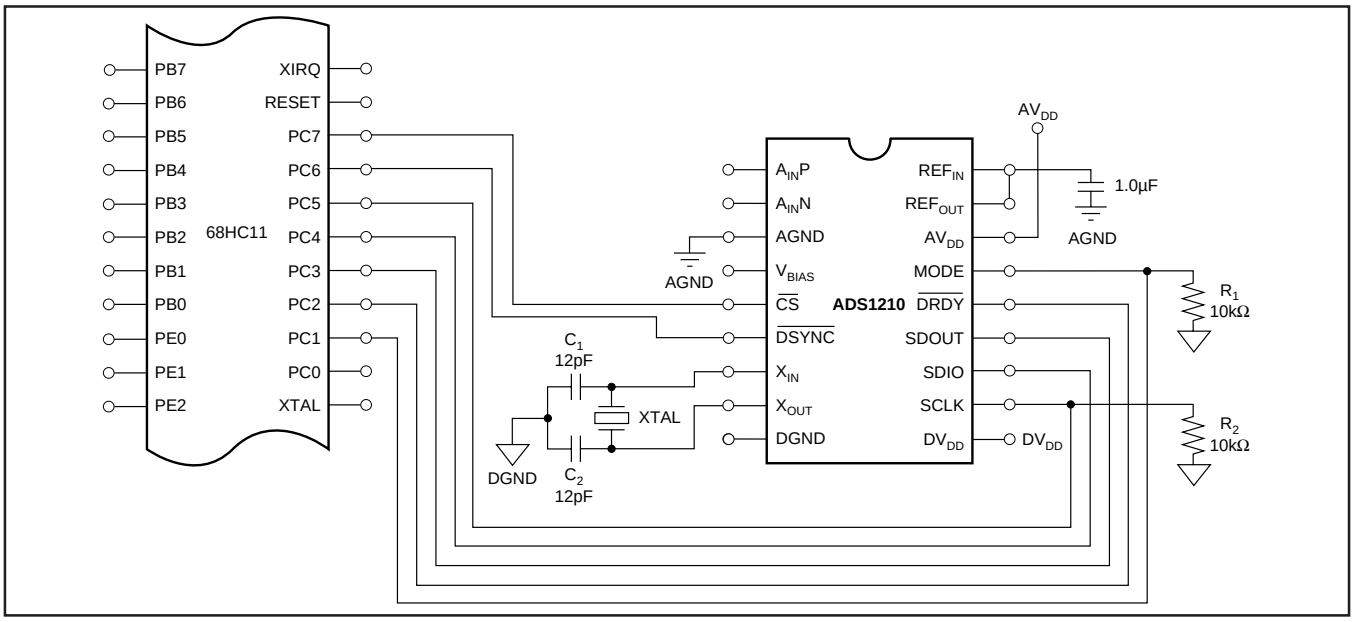


FIGURE 32. Full Interface with a 68HC11 Microprocessor.

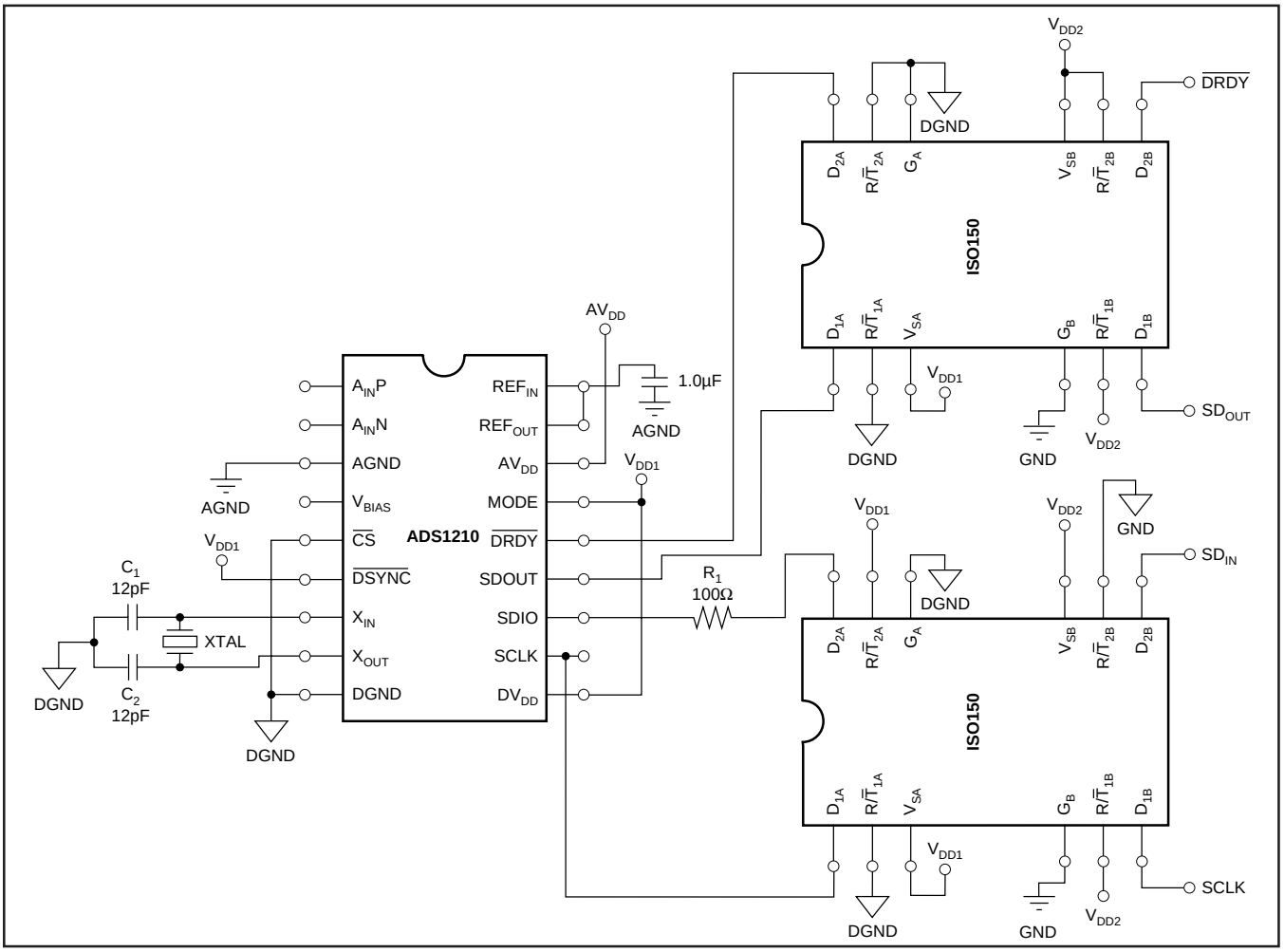


FIGURE 33. Isolated Four-Wire Interface.

Isolation

The serial interface of the ADS1210/11 provides for simple isolation methods. An example of an isolated four-wire interface is shown in Figure 33. The ISO150 is used to transmit the digital signals over the isolation barrier.

In addition, the digital outputs of the ADS1210/11 can, in some cases, drive opto-isolators directly. Figures 34 and 35 show the voltage of the SDOUT pin versus source or sink current under worst case conditions. Worst-case conditions for source current occur when the analog input differential

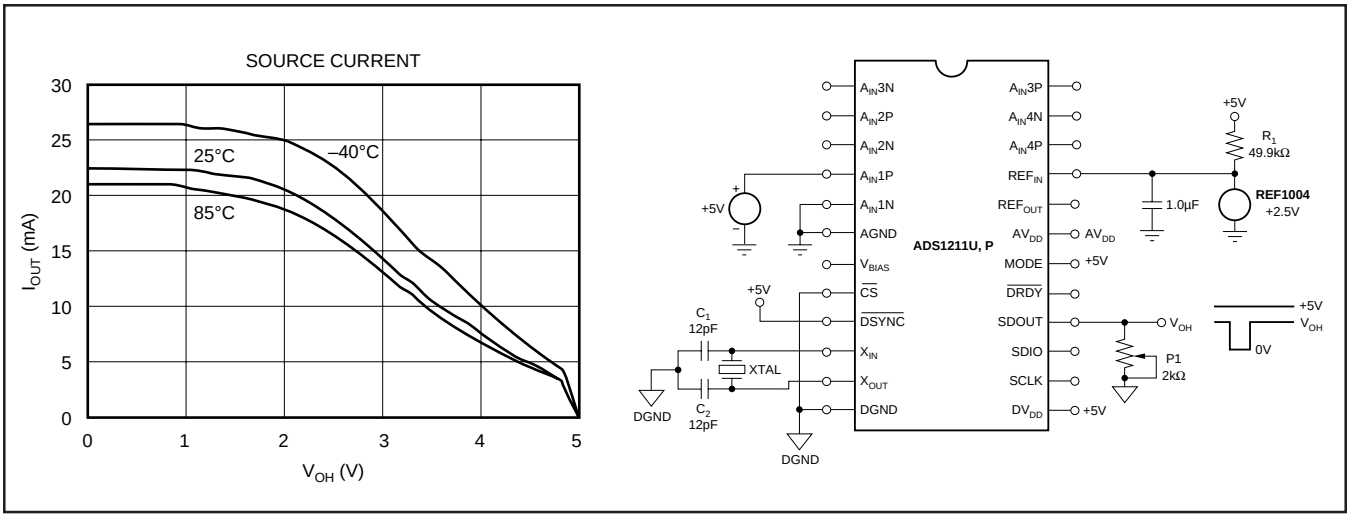


FIGURE 34. Source Current vs V_{OH} for SDOUT Under Worst-Case Conditions.

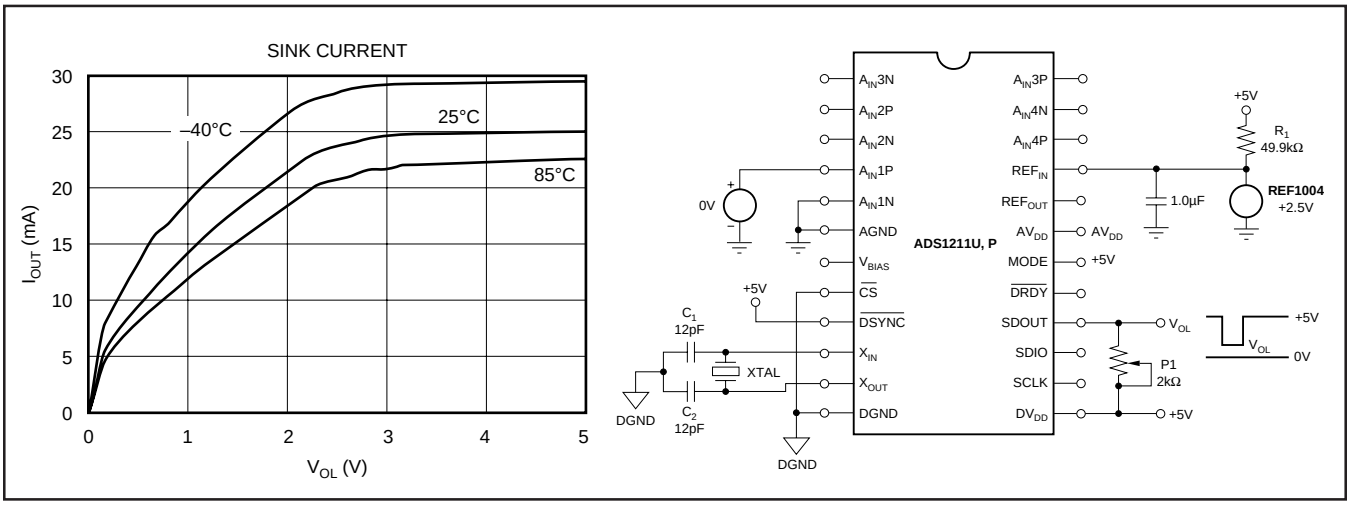


FIGURE 35. Sink Current vs V_{OL} for SDOUT Under Worst-Case Conditions.

voltage is 5V and the output format is Offset Binary ($FFFFFF_H$). For sink current, the worst-case condition occurs when the analog input differential voltage is 0V and the output format is Two's Complement (000000_H).

Note that SDOUT is tri-stated for the majority of the conversion period and the opto-isolator connection must take this into account.

Synchronization of Multiple Converters

The $\overline{DSYNC}$ input is used to synchronize the output data of multiple ADS1210/11s. Synchronization involves configuring each ADS1210/11 to the same Decimation Ratio and Turbo Mode setting, and providing a common signal to the X_{IN} inputs. Then, the $\overline{DSYNC}$ signal is pulsed LOW (see Figure 22 in the Timing section). This results in an internal reset of the modulator count for the current conversion. Thus, all the converters start counting from zero at the same time, producing a $\overline{DRDY}$ LOW signal at approximately the same point (see Figure 36).

Note that an asynchronous $\overline{DSYNC}$ input may cause multiple converters to be different from one another by one X_{IN} clock cycle. This should not be a concern for most applications. However, the Timing section contains information on exactly synchronizing multiple converters to the same X_{IN} clock cycle.

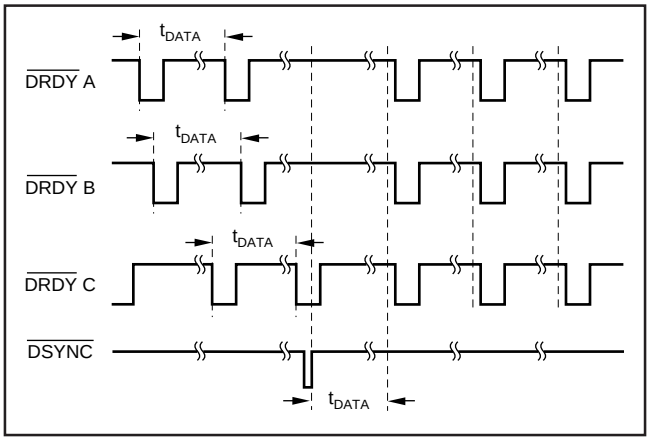


FIGURE 36. Affect of Synchronization on Output Data Timing.

LAYOUT

POWER SUPPLIES

The ADS1210/11 requires the digital supply (DV_{DD}) to be no greater than the analog supply (AV_{DD}) +0.3V. In the majority of systems, this means that the analog supply must come up first, followed by the digital supply. Failure to observe this condition could cause permanent damage to the ADS1210/11.

Inputs to the ADS1210/11, such as SDIO, A_{IN}, or REF_{IN}, should not be present before the analog and digital supplies are on. Violating this condition could cause latch-up. If these signals are present before the supplies are on, series resistors should be used to limit the input current (see the Analog Input and V_{BIAS} sections of this data sheet for more details concerning these inputs).

The best scheme is to power the analog section of the design and AV_{DD} of the ADS1210/11 from one +5V supply and the digital section (and DV_{DD}) from a separate +5V supply. The analog supply should come up first. This will ensure that A_{IN} and REF_{IN} do not exceed AV_{DD} and that the digital inputs are present only after AV_{DD} has been established, and that they do not exceed DV_{DD}.

The analog supply should be well regulated and low noise. For designs requiring very high resolution from the ADS1210/11, power supply rejection will be a concern. See the PSRR vs Frequency curve in the Typical Performance Curves section of this data sheet for more information.

The requirements for the digital supply are not as strict. However, high frequency noise on DV_{DD} can capacitively couple into the analog portion of the ADS1210/11. This noise can originate from switching power supplies, very fast microprocessors or digital signal processors.

For either supply, high frequency noise will generally be rejected by the digital filter except at interger multiplies of f_{MOD}. Just below and above these frequencies, noise will alias back into the passband of the digital filter, affecting the conversion result.

If one supply must be used to power the ADS1210/11, the AV_{DD} supply should be used to power DV_{DD}. This connection can be made via a 10Ω resistor which, along with the decoupling capacitors, will provide some filtering between DV_{DD} and AV_{DD}. In some systems, a direct connection can be made. Experimentation may be the best way to determine the appropriate connection between AV_{DD} and DV_{DD}.

GROUNDING

The analog and digital sections of the design should be carefully and cleanly partitioned. Each section should have its own ground plane with no overlap between them. AGND should be connected to the analog ground plane as well as all other analog grounds. DGND should be connected to the digital ground plane and all digital signals referenced to this plane.

The ADS1210/11 pinout is such that the converter is cleanly separated into an analog and digital portion. This should allow simple layout of the analog and digital sections of the design.

For a single converter system, AGND and DGND of the ADS1210/11 should be connected together, underneath the converter. Do not join the ground planes, but connect the two with a moderate signal trace. For multiple converters, connect the two ground planes at one location as central to all of the converters as possible. In some cases, experimentation may be required to find the best point to connect the two planes together. The printed circuit board can be designed to provide different analog/digital ground connections via short jumpers. The initial prototype can be used to establish which connection works best.

DECOUPLING

Good decoupling practices should be used for the ADS1210/11 and for all components in the design. All decoupling capacitors, but specifically the 0.1µF ceramic capacitors, should be placed as close as possible to the pin being decoupled. A 1µF to 10µF capacitor, in parallel with a 0.1µF ceramic capacitor, should be used to decouple AV_{DD} to AGND. At a minimum, a 0.1µF ceramic capacitor should be used to decouple DV_{DD} to DGND, as well as for the digital supply on each digital component.

SYSTEM CONSIDERATIONS

The recommendations for power supplies and grounding will change depending on the requirements and specific design of the overall system. Achieving 20 bits or more of effective resolution is a great deal more difficult than achieving 12 bits. In general, a system can be broken up into four different stages:

Analog Processing
Analog Portion of the ADS1210/11
Digital Portion of the ADS1210/11
Digital Processing

For the simplest system consisting of minimal analog signal processing (basic filtering and gain), a self-contained microcontroller, and one clock source, high-resolution could be achieved by powering all components by a common power supply. In addition, all components could share a common ground plane. Thus, there would be no distinctions between “analog” and “digital” power and ground. The layout should still include a power plane, a ground plane, and careful decoupling.

In a more extreme case, the design could include: multiple ADS1210/11s; extensive analog signal processing; one or more microcontrollers, digital signal processors, or microprocessors; many different clock sources; and interconnections to various other systems. High resolution will be very difficult to achieve for this design. The approach would be to break the system into as many different parts as possible. For example, each ADS1210/11 may have its own “analog” processing front end, its own analog power and ground (possibly shared with the analog front end), and its own “digital” power and ground. The converter’s “digital” power and ground would be separate from the power and ground for the system’s processors, RAM, ROM, and “glue” logic.

APPLICATIONS

The ADS1210/11 can be used in a broad range of data acquisition tasks. The following application diagrams show the ADS1210 and/or ADS1211 being used for bridge transducer measurements, temperature measurement, and 4-20mA receiver applications.

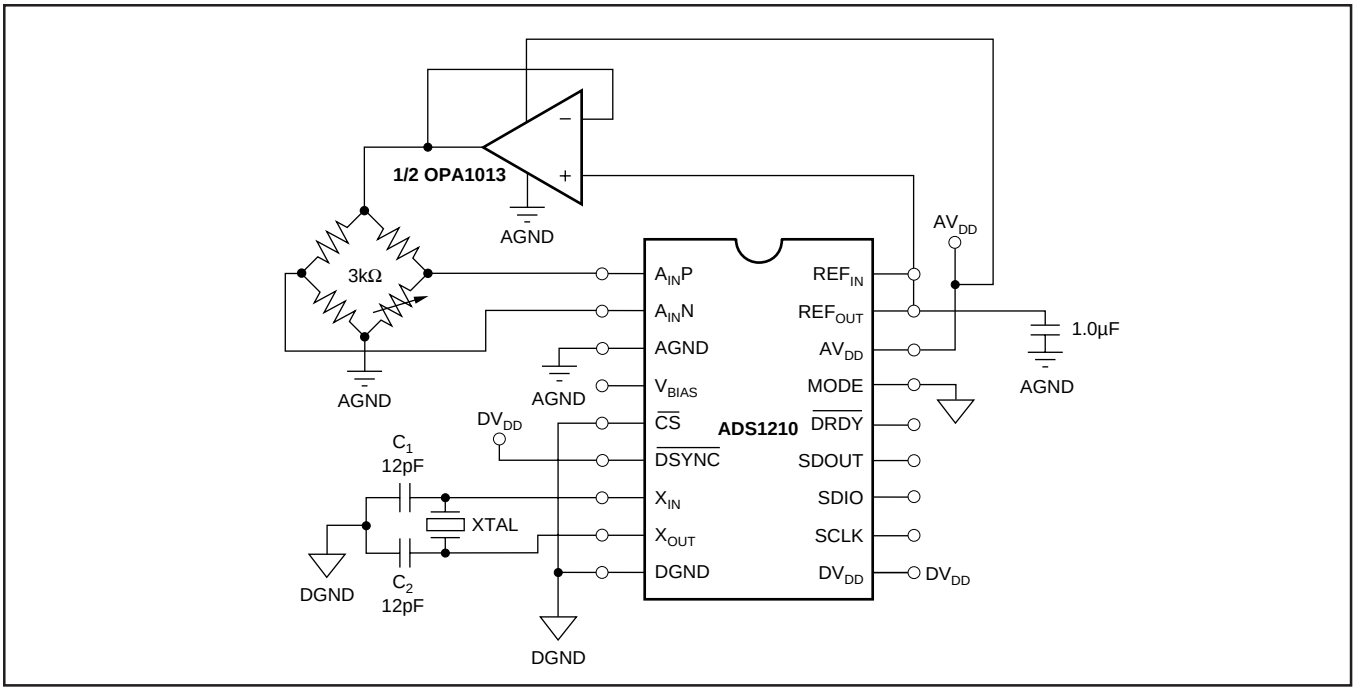


FIGURE 37. Bridge Transducer Interface with Voltage Excitation.

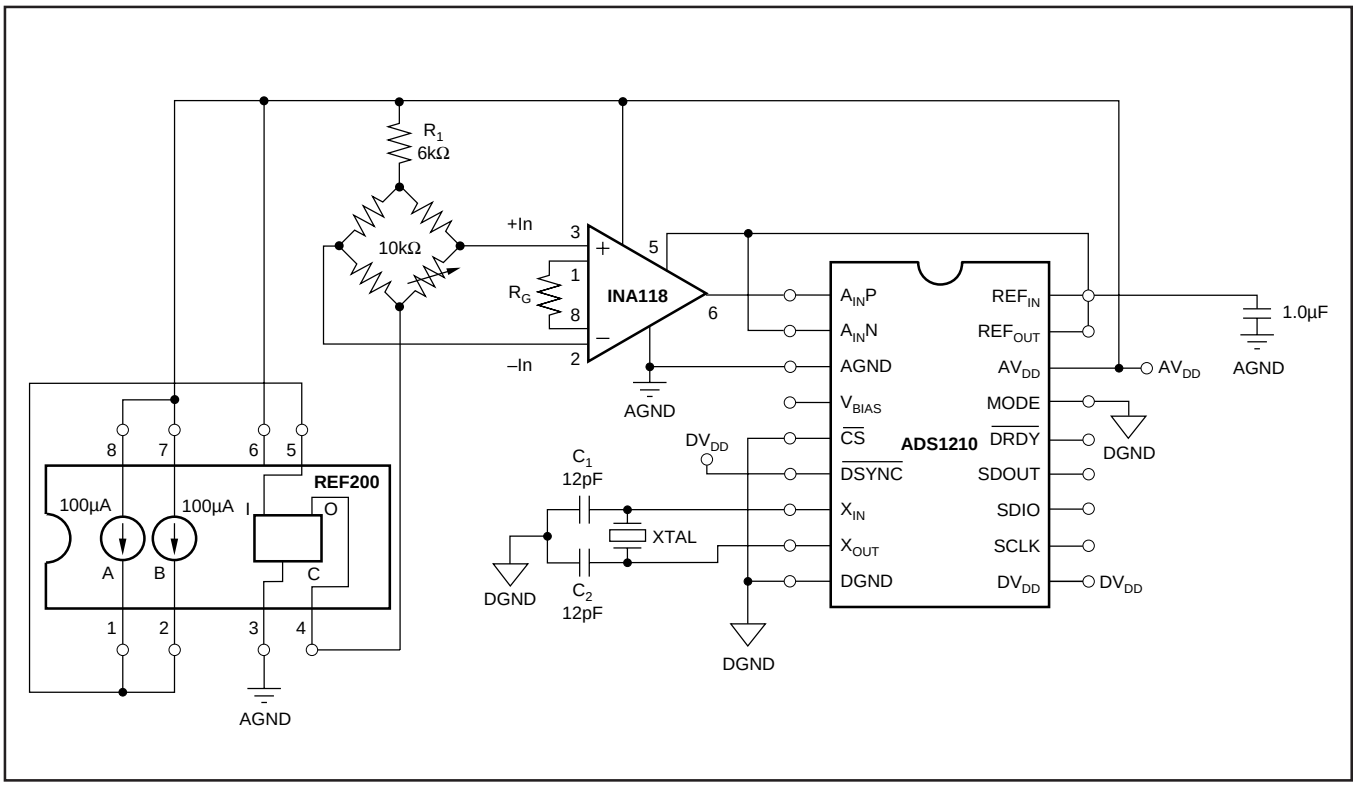


FIGURE 38. Bridge Transducer Interface with Current Excitation.

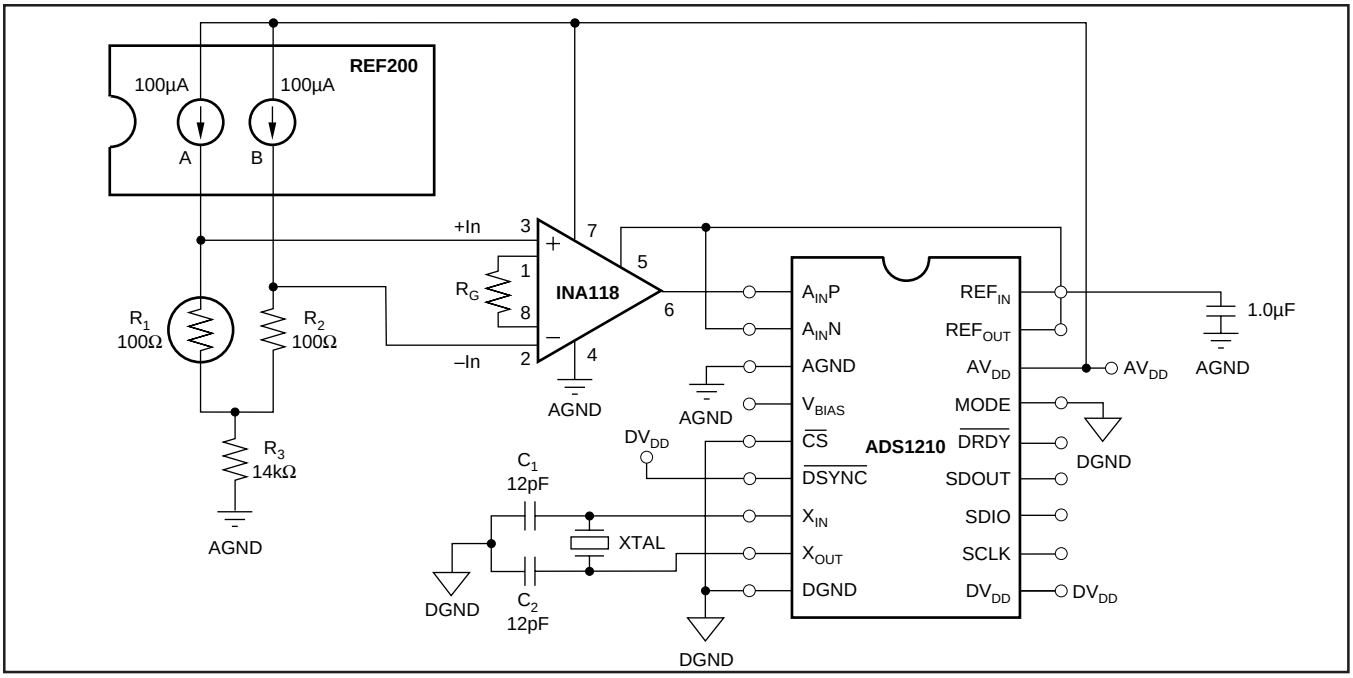


FIGURE 39. PT100 Interface.

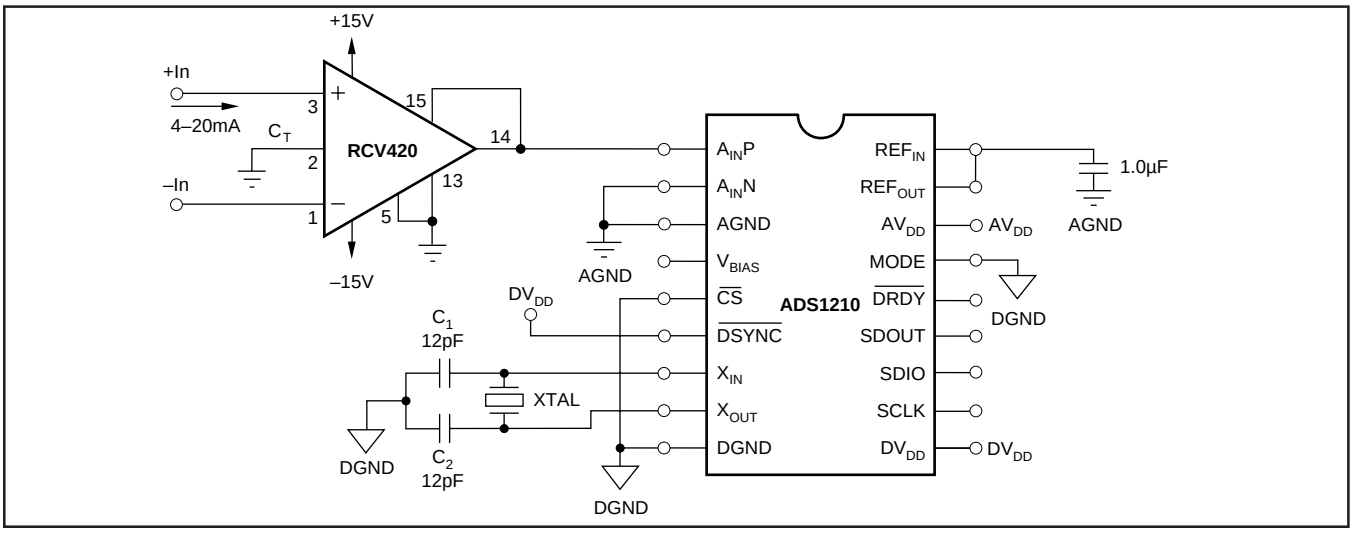


FIGURE 40. Complete 4-20mA Receiver.

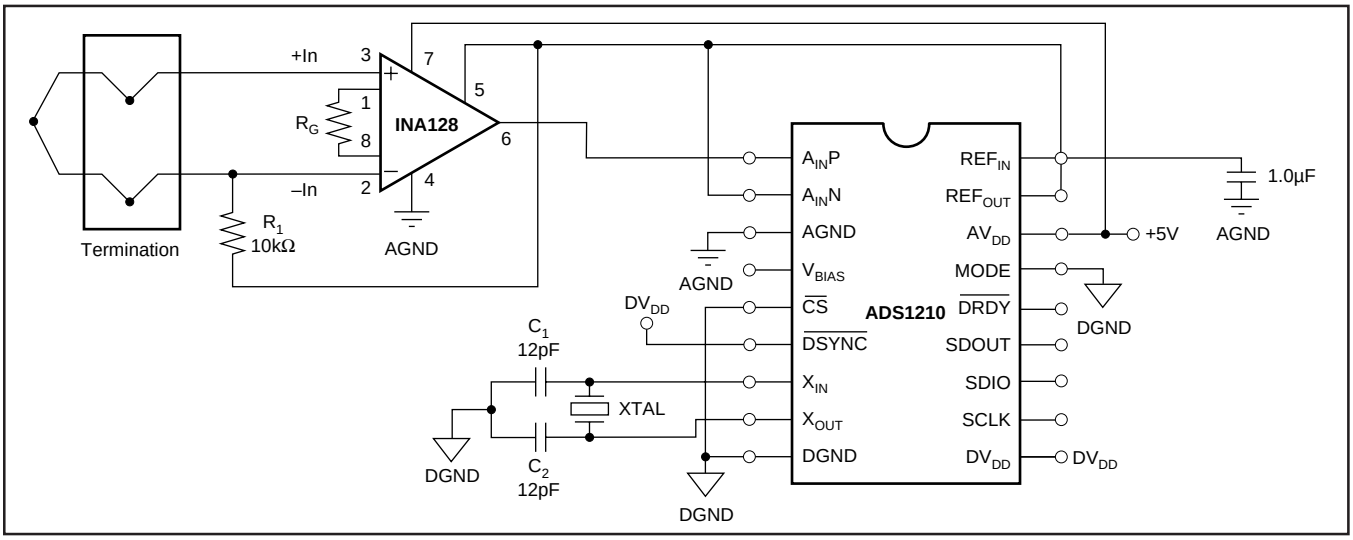


FIGURE 41. Single Supply, High Accuracy Thermocouple.

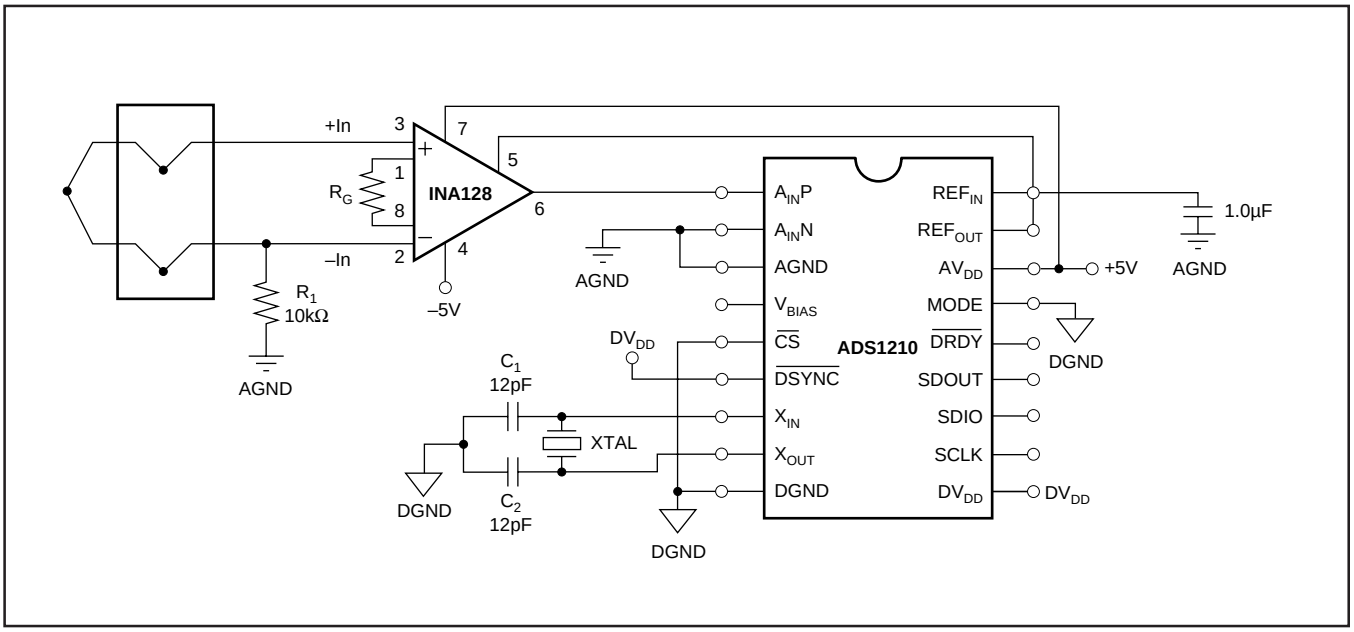


FIGURE 42. Dual Supply, High Accuracy Thermocouple.

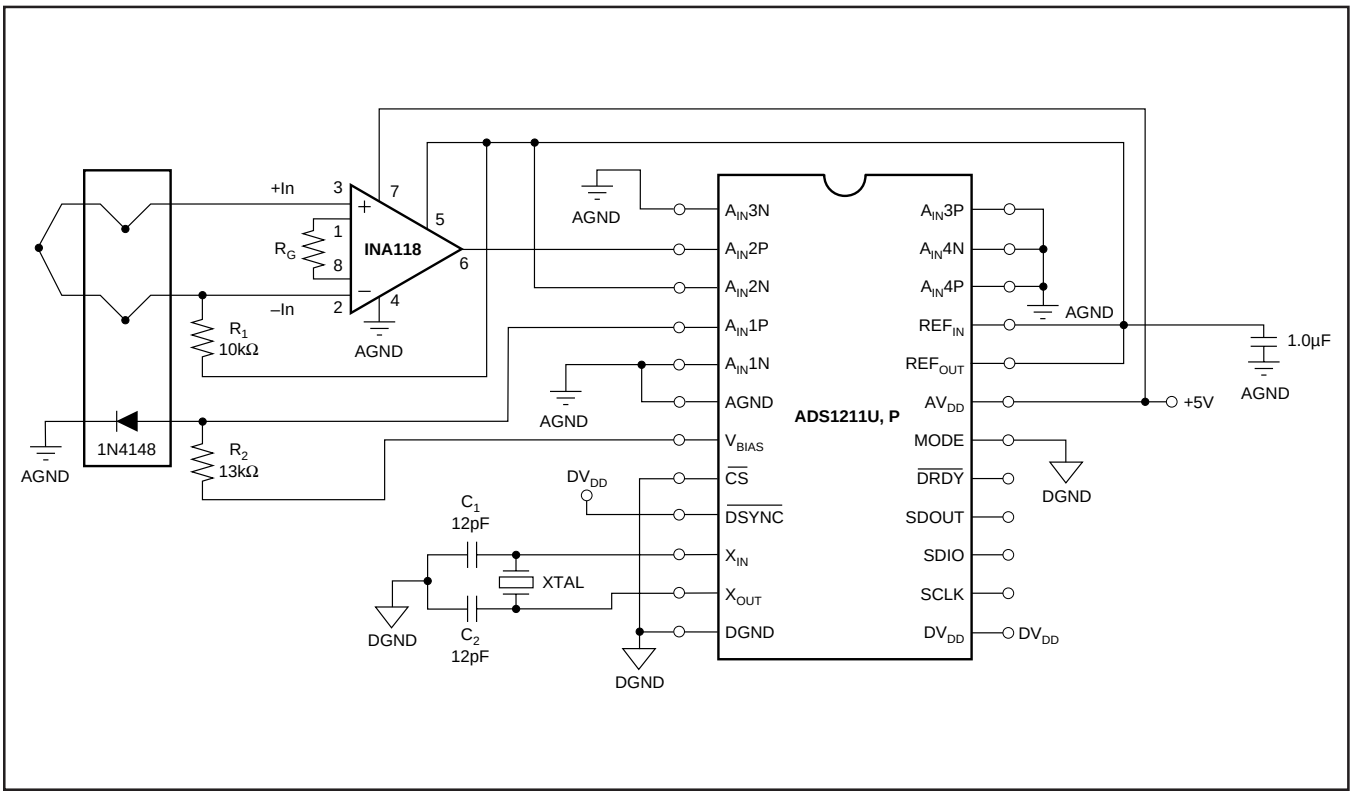


FIGURE 43. Single Supply, High Accuracy Thermocouple Interface with Cold Junction Compensation.

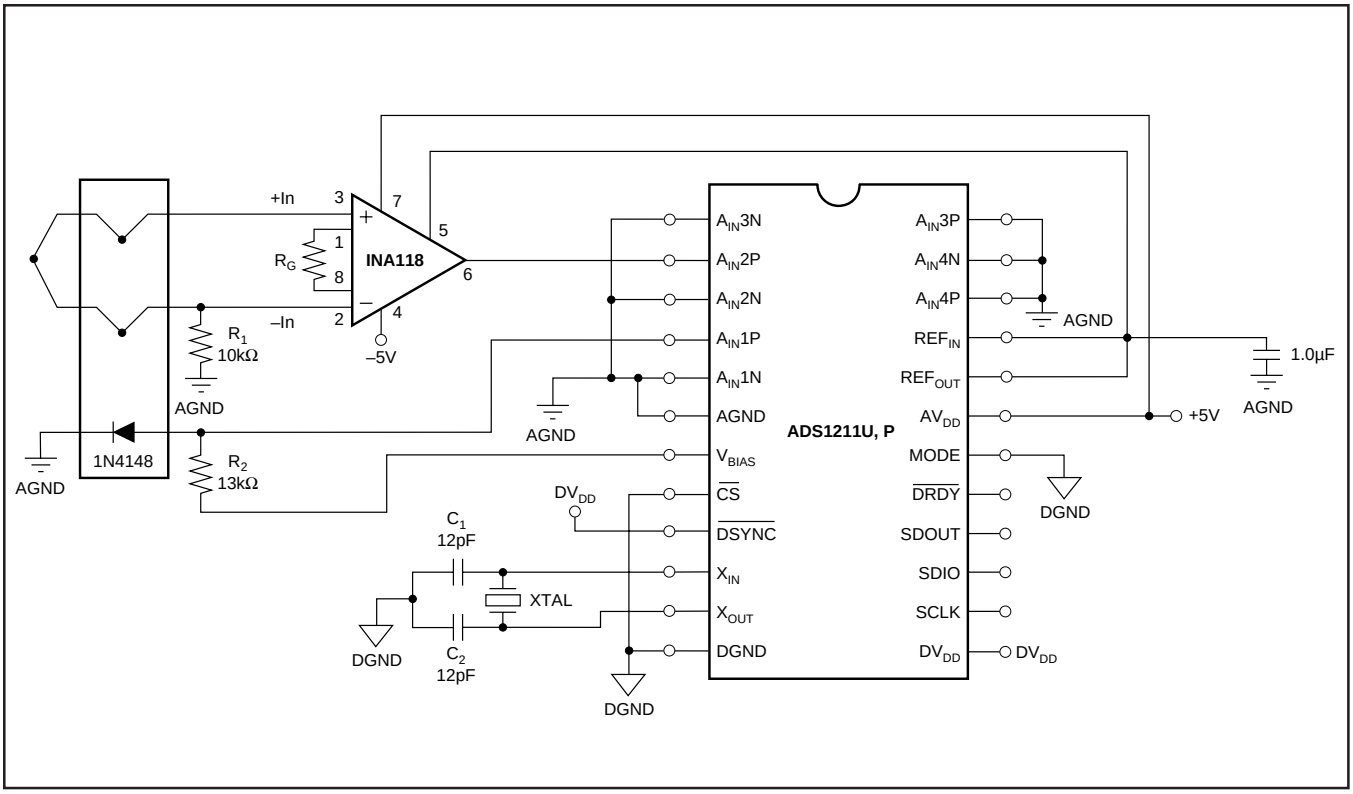


FIGURE 44. Dual Supply, High Accuracy Thermocouple Interface with Cold Junction Compensation.

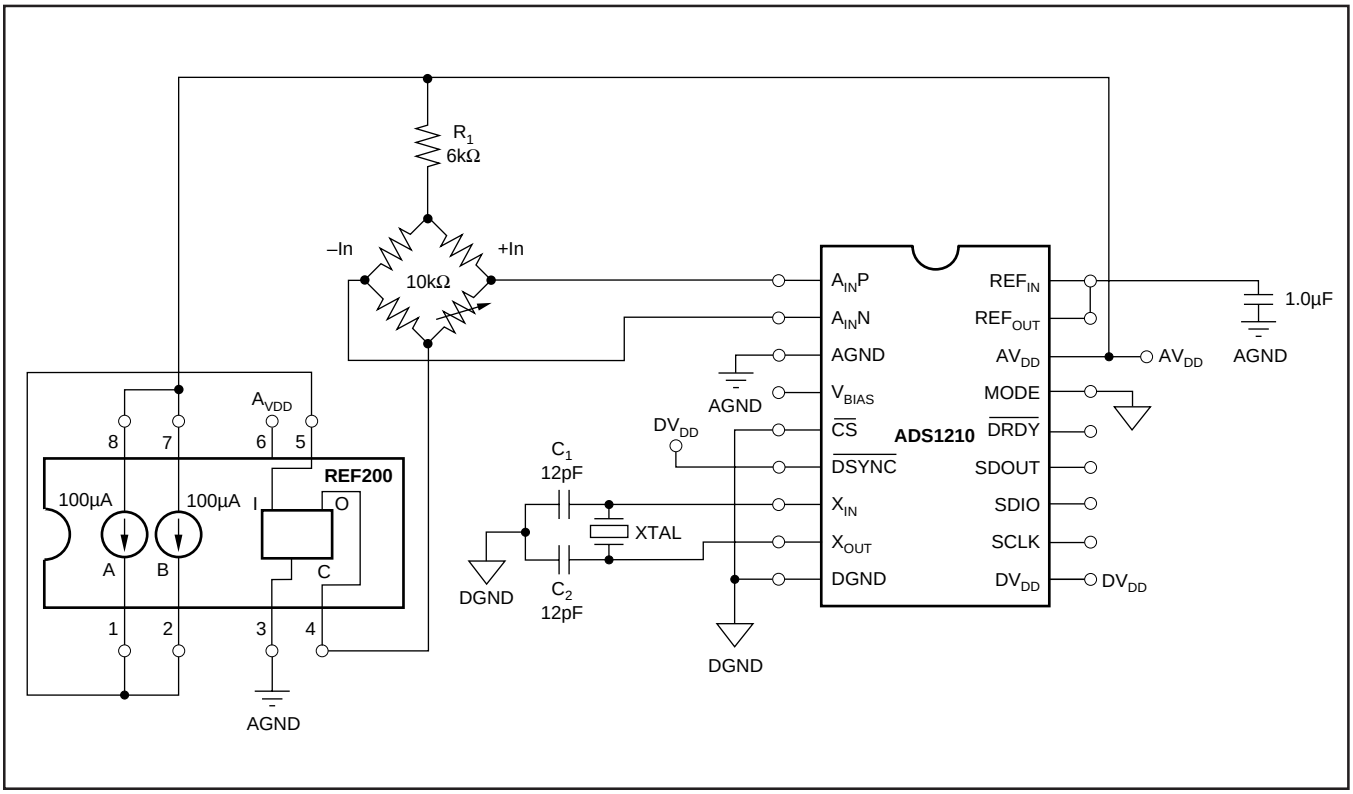


FIGURE 45. Low Cost Bridge Transducer Interface with Current Excitation.

TOPIC INDEX

FIGURE INDEX

TABLE INDEX

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

SR PARGPS PRODUCT FLYER

The Symmetric Research PARGPS timing accessory provides precision time stamping for data acquired with the SR PARxCH A/D boards. Based on the Garmin GPS 18x LVC receiver, the system provides 10 usec accuracy for the PAR1CH and PAR4CH systems, and 0.80 usec absolute accuracy for the PAR8CH.

The PARGPS marks time stamps and location in real time as PARxCH analog data is acquired. Industry standard RS232 NMEA strings are used for coarse timing and location, while a precision PPS (pulse per second) signal is used for accurate time stamping. The NMEA messages are connected to a PC COM port by a standard DB9 serial port cable, while the PPS signal is fed directly to the digital IO connector of the PARxCH for processing by the A/D system.

Full software support is included at no cost for Windows 2K/XP and Linux. Acquisition applications such as PARxCH Scope and Simp automatically store PARGPS ongoing time and location information along with the 24 bit analog data. Source code and circuit diagrams are included for system customization.

HARDWARE FEATURES

- Ideal for applications requiring accurate time stamping of PARxCH analog data
- Uses Garmin GPS 18x LVC receiver with standard NMEA string serial messages
- Accurate PPS pulse per second signal with toggling red LED
- Time stamp accuracy of 800 ns (PAR8CH) or 10 us (PAR4CH,PAR1CH)

- PARGPS board sits outside the PC and stacks with the PARxCH
- NMEA string serial messages connect to PC RS232 COM ports
- PPS signal connects to PARxCH digital input for precision time stamping

- 9 vdc / 130 ma power consumption, same physical dimension as PAR4CH
- Cables, vinyl steel enclosure, and Garmin antenna included with ”in the box” systems

SOFTWARE FEATURES

- Ready to go software for immediate data acquisition with time stamping
- Utilities to display GPS time and location while acquisition is in progress
- Win 2K/XP and Linux compatible
- Full documentation including source code and circuit diagrams

PRICES

SR Product	Description	Price
PARGPS	PARxCH GPS timing and location accessory	$285
PAR1CH	1 channel 24 bit A/D system for PC parallel port	$220
PAR4CH	4 channel 24 bit A/D system for PC parallel port	$575
PAR8CH	8 channel 24 bit A/D system for PC parallel port	$980
TRANS-WALL-09V-DC	Wall transformer for powering PARGPS separately	$15

Symmetric Research **Email: info@symres.com** **Web: www.symres.com**

Initial product release 2001 January 2012 cash prices listed

PARGPS User's Guide

Manual Release 12/05/07

Web: www.symres.com

LIMITED WARRANTY

WHAT IS COVERED

Symmetric Research warrants its PARGPS 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 PARGPS 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 PARGPS product for use with customer provided power supplies. Incorrectly connecting power may permanently damage the system.

Furthermore, PARGPS systems that have been customer modified 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 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 PARGPS product.

TABLE OF CONTENTS

INTRODUCTION

The Symmetric Research PARGPS timing module provides accurate time stamping for data acquired with the SR PARxCH A/D 24 bit A/D boards. Typical timing accuracy is 10 microseconds.

Based on the Trimble ACE-III GPS receiver or the Garmin GPS 18 LVC receiver, the SR PARGPS board provides both time and location information on an ongoing real time basis as data is acquired. It uses industry standard RS232 National Marine Electronics Association (NMEA) messages for location and coarse timing. Precision timing is provided by the highly accurate pulse-per-second (PPS) signal. With its direct connection to the PARxCH digital inputs, the PARGPS board provides excellent time stamping with the least dependence on PC interrupt latency.

The PARGPS board sits outside the PC in a vinyl covered steel enclosure that stacks nicely with the PAR4CH. Alternatively, for PC104 applications, an adapter plate is available to connect the PARGPS and PAR4CH to a PC104 stack. A standard DB9 serial port cable to COM1 or COM2 carries the NMEA messages from the PARGPS to the PC, while the PPS signal is fed directly to the digital I/O connector of the PARxCH for processing by the A/D system.

Included with the system is full software support for Win2000/XP and Linux. The high level PARxCH acquisition applications such as scope and simp have been upgraded for GPS support, along with a new DAT file format for storing ongoing time and location information along with the acquired data. Additional utilities are included for various operations with the time information. As with all SR products, the included circuit diagrams, low level function libraries and source code allow users to customize the system to their specifications. The Win2000/XP and Linux software features true kernel mode drivers for good performance and reliable operation.

Other items included with the system are a cable for sharing power with the PARxCH, a cable for connecting the PPS signal to the PARxCH, a cable for connecting to the PC serial port, a 6 meter antenna, and circuit diagrams. This is everything needed to be up and running right away.

We hope you enjoy using the SR PARGPS

INSTALLATION

You must have an SR PARxCH A/D acquisition board to use the PARGPS system. The PARxCH and PARGPS work together. Installation of the PARGPS is straightforward. First, install the PARxCH as described in the PARxCH User's Guide. Then you'll need to daisy chain power with the PARxCH, connect to the PARxCH digital I/O header, connect to the PC serial port, connect the antenna, and install the software. The individual steps in more detail are:

CONNECTING POWER

The PARGPS requires power independently of the PC. Connect the wall transformer supplied with the PARxCH to one of the 2.1mm connectors on the back of the PARGPS unit. Then use the 12" black cable supplied with the PARGPS to daisy chain power from the extra 2.1mm connector on the PARGPS back to the PARxCH.

Once plugged in, you can verify the wall transformer is on by checking the green LED near the power connector on the back of the PARGPS. If it is on, your wall transformer is plugged in and powered on.

The PARGPS can be powered with a wide range of wall transformers. Any AC or DC wall transformer with a voltage between 9 and 24 volts is acceptable as long as it has a 2.1mm, center-plus plug. If you have a choice, chose a voltage nearer to 9 volts rather than 24. Running at lower power supply voltages will reduce the heat dissipation of the internal regulators. If you are running the PARGPS outside of its enclosure in a PC104 application, you can connect power via the 4 position Molex connectors J402 or J403 instead. Consult the circuit diagrams and board legend for proper connections.

The PARGPS is protected with diodes so it is unlikely incorrectly connected power supplies will damage it. However, the PARGPS 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. Please read the limited warranty at the beginning of this manual.

CONNECTING THE PPS SIGNAL TO THE PARXCH

Use the included short ribbon cable with 15 pin female D-shell connectors for connecting the PPS signal from the PARGPS board to the digital I/O connector on the right front of the

PARxCH A/D. This connection is easiest to make when the PARGPS unit is stacked above or below the PARxCH.

CONNECTING TO THE PC SERIAL PORT

Use the included 6 ft molded 9 pin D-shell cable for connecting to the PC serial port. Connect the male end of this cable to the 9 pin female D-shell connector on the back of the PARGPS, and the female end to a serial port connector on your PC. It doesn't matter whether you connect to COM1 or COM2, but make a note of which one you select.

CONNECTING THE ANTENNA

The antenna connection is on the far left side of the back panel. For the PARGPS Rev C board that works with the Trimble ACE-III receiver, it is a small SMB coax connector. For the PARGPS Rev D board that works with the Garmin GPS 18 LVC receiver/antenna, it is a 9 pin male D-shell connector. Connect the supplied antenna to this connector. The other end of the cable contains the active part of the GPS antenna. This active end of the **antenna must have access to the open sky** around the equator in order for the system to work. The antenna will not work if buried under layers of concrete or in basements. More sophisticated antennas for difficult environments are available. Contact SR for information about companies that can supply such antennas.

After following the 4 steps above, the red LED on the front of the PARGPS unit should be toggling on and off at a 1 second rate. If not, check power and the green LED on the back of the PARGPS. Note that the red LED **does not** indicate GPS satellites have been acquired. The Rev C system emits a PPS signal even with no satellite lock. The Rev D system will not start toggling the red LED until it gets enough satellites to begin emitting a PPS signal, but it will continue emitting PPS even if it loses satellite lock later. Only by running the diagnostic software can you determine the PPS quality and number of satellites the system is currently seeing.

INSTALLING THE SOFTWARE

A CD-ROM is included with the PARGPS. Software for each supported operating system can be found in the respective directory. Each OS subdirectory includes a compressed PKZIP or Linux gzipped tar format file, and an installation batch or script file. To install the PARGPS software, change to the OS subdirectory appropriate for your computer and run the install batch script. This will automatically unzip and create the default \sr\parxch and \sr\pargps or Linux /usr/local/sr/parxch and /usr/local/sr/pargps directory structure on your hard disk.

We highly recommend that 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. See the readme.txt files on the CD for more information.

For Win2000/XP and Linux installations, besides copying the software to your hard disk, you must also install a kernel mode device driver for both the PARxCH and the PARGPS. Administrator or root permissions are required for this step, but once the driver is installed, ordinary users can use it freely.

To install the PARxCH and PARGPS kernel mode device driver, change to the respective driver subdirectory on your hard disk and run the indriver program. When run with no arguments indriver shows a usage message that describes the required arguments. For the PARxCH this is xch model, port address, and port irq. For the PARGPS it is the GPS model and driver name. You can remove the device driver using its name and the respective rmdriver program. When choosing the PARxCH arguments, you must pick ones which correspond to the parallel port you are using for the A/D board.

For example, to install the PARxCH and PARGPS drivers typical usage would be:

```
> cd \sr\parxch\driver
> indriver PAR4CH 0x378 7
> cd \sr\pargps\driver
> indriver GARMIN SrParGps0
```

To remove them use:

```
> cd \sr\parxch\driver
> rmdriver SrParXch0
> cd \sr\pargps\driver
> rmdriver SrParGps0
```

If you forget the exact assigned device driver name, you can find it with the showdriver utility or on a list of installed drivers provided by the OS. For Win2000/XP, open the Device Manager and look for drivers of the SR Precision Instrumentation class. For Linux, run /sbin/lsmod. For additional information on installing the device driver, please refer to the readme.txt files in the both driver subdirectories.

If you have the PARxCH and PARGPS powered on and correctly connected, you can verify correct operation by running the diag programs. First run the PARxCH diag program in the parxch diag directory to test the A/D board and its connections. Then run the PARGPS diag program located in the pargps diags directory to test the PARGPS and its connections. Both diag programs are designed to be run from a command prompt. Follow the on screen messages for more information.

For programmers, the core part of the software supplied with the system is a collection of low level functions that are required to communicate with the PARGPS. For Windows systems, these functions can be linked in statically or used dynamically from the Dynamic Link Library pargps.dll. Programs using pargps.dll must be able to find it at run time or Windows will give an error message. The best way to inform the system of the location of pargps.dll is to add its location to your execution path. The following command can be executed from the command line or added to your autoexec.bat file:

```
> set path=%path%;\sr\pargps\lib
```

For Linux systems, the core low level functions can also be linked in statically or used dynamically from the shared library pargps.so. Programs using the shared library should set the LD_LIBRARYPATH environment variable so the library can be found at runtime. One way to do this is by executing the following command line or adding it to your .profile file:

```
LD_LIBRARYPATH=$ LD_LIBRARYPATH:/usr/local/sr/pargps/lib ; export  \
$LD_LIBRARYPATH
```

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.

INSTALLATION CHECKLIST:

* Install the PARxCH as described in the PARxCH User's Manual and run the PARxCH diagnostic program to be sure it is working correctly.
* Plug the wall transformer from the PARxCH into the PARGPS and daisy chain power over to the PARxCH. Check that the green LED on the PARGPS back panel is on.
* Connect the supplied 9 pin D-shell cable from the 9 pin female D-shell on the PARGPS to your PC serial port.
* Connect the supplied 15 pin D-shell ribbon cable between the PARxCH digital I/O connector and the PARGPS.
* Connect the supplied antenna.
* Run the PARGPS diagnostic program in the pargps diags directory. This will show whether the PARxCH and PARGPS drivers are installed and talking with each other. It also verifies that the PPS signal is being received and passed to the PARxCH and that the NMEA messages are coming in over the serial port. Note that it may take up to 10 minutes after 3 satellites are found for valid location information to appear in the NMEA strings.
* Try out the simp program to acquire some data and see the system work

DIAGNOSTIC AND UTILITY PROGRAMS

The SR PARGPS comes with diagnostic and utility programs you can run immediately after installing the system. These programs will help you become more familiar with the system. The source code is included for those wanting to modify these programs for custom applications.

For more details, refer to the readme.txt files in the diags, currtime and utils directories and comments in the source files.

diag

There are two simple diagnostic programs, located in the diags directory, that let you verify the PARGPS and PARxCH are properly connected and functioning correctly. They test that both the PARGPS and PARxCH drivers are properly installed and check the PPS signal and NMEA serial messages.

Diagser takes one argument specifying the serial port and displays the NMEA messages.

Diag takes up to six command line arguments. These specify the names of the PARGPS and PARxCH drivers, which serial port to use for the NMEA messages, which parallel port mode to use, and which PARxCH and PARGPS device to use. The arguments can be given in any order and arguments matching the provided default can be omitted as long as at least one argument is given. Run diag with no arguments for a usage message and a list of valid argument values.

currtime

This is a command line program that displays the current GPS time. An active GPS program such as PARGPS diag or one of the PARxCH acquisition programs with GPS enabled must be running for currtime to work correctly. These active GPS programs store GPS time information in the device driver for currtime to retrieve and display. A GUI version of this program, called GpsTime, is available in the vbasic directory.

sethdrtm

This is a utility program to update the start time in the header portion of a .DAT file based on the GPS data included in the body of the .DAT file. The PARxCH scope and simple

acquisition programs can be used to generate .DAT files. The sethdrtm program is located in the utils directory.

scope,simp,dat2asc

These programs come with the PARxCH software, but are mentioned here since you will mostly likely want to use them when working with the PARGPS. Scope and simp are acquisition programs. When running them with the PARGPS, be sure to enable GPS and select the .DAT output file format. This can be done from the scope menus or the simp .ini file. Dat2asc is a utility program that converts the binary .DAT file to a simple text format.

LIBRARY FUNCTIONS

The library functions are at the core of the software supplied with the SR PARGPS. They allow users to control board operation from high level languages without having to know the low level details of how the system operates. These functions can be statically linked to C programs for any OS. In addition, they are available as a Dynamic Link Library (DLL) under Windows and as a shared library (.so) under Linux. When used as a DLL, these functions can be called from other high level programming languages such as Microsoft Visual Basic. This chapter covers usage from programming environments like Visual C**. Remember that the PARGPS must be used in conjunction with the PARxCH and GPS data can not be acquired without using the PARxCH functions.**

The outline of how to use the PARxCH and PARGPS library functions is fairly simple. First call both the PARxCH and PARGPS Open functions to open the drivers and perform various initialization steps. Then call the PARxCH AttachGps function so the two drivers can properly interact with each other. Once the boards have been initialized, call both Start functions so the PARGPS begins checking for PPS signals and NMEA serial data and the PARxCH begins acquiring data. Then use ReadDataWithGpsMark, ReadPpsData and ReadSerialData to move the analog, PPS and serial data into PC memory arrays which can be displayed or saved to the hard disk. The analog data comes from the PARxCH hardware FIFO and while the PPS and serial data come from the PARGPS software FIFO. When you are done, call both Stop and Close functions to stop acquiring data and close the drivers.

The PARGPS library also includes several helper functions for working with the NMEA message strings and various representations of time.

When using the PARGPS library functions, be sure to include the header file pargps.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 pargps.h useful for making your programs more readable. When using dynamic linking, make sure the pargps.dll or pargps.so library is on your Windows execution path or Linux LD_LIBRARY path so it can be found at run time.

The following is a discussion of each PARGPS library function. Refer to the PAR4CH program simp4ch.c for an example of how to use the library functions.

OPEN AND CLOSE DEVICE

```
C usage:
       #include "pargps.h"

       DEVHANDLE ParGpsOpen(
                           char* ParGpsName,
                           int SerialPortNumber,
                           int InterruptMode,
                           int *Error
                           );

       int ParGpsClose( DEVHANDLE handle );
```

The Open function opens the PARGPS device driver and provides a device handle that is passed to the remaining library functions. It also initializes the PARGPS board. Open should be the first library function called.

It returns a valid device handle on success and the value BAD_DEVHANDLE otherwise. If the optional Error argument is provided, it is filled with an error code giving more detail about why the open failed. See pargps.h for a list of the possible error codes. Pass NULL to ignore this parameter.

The ParGpsName parameter identifies the device driver. The SerialPortNumber parameter controls which serial port the PARGPS library uses to receive NMEA messages. The InterruptMode should typically be set to INTERRUPT_ALTERNATING. Use the defined constants in pargps.h to specify these parameters.

This function uses the default GPS model which is set at driver installation time. If you wish to select a different GPS model, use the ParGpsFullOpen function instead.

The Close function cleanly shutdowns the device driver. It should be the last library function called. In some cases, it may be called implicitly at program termination. However, you should not rely on this.

PARxCH ATTACH AND RELEASE GPS

```
C usage:
       #include "pargps.h"
       #include "parxch.h"

       int ParXchAttachGps(
                           DEVHANDLE ParXchHandle,
                           DEVHANDLE ParGpsHandle,
                           int *Error
                           );

       int ParXchReleaseGps( DEVHANDLE ParXchHandle );
```

The AttachGps function in the PARxCH library takes a device handle to both the PARxCH and the PARGPS and establishes a connection between the two drivers so they can interact. This interaction is required since the PARGPS PPS signal is presented to the PC via the interrupt associated with the PARxCH parallel port and both drivers need to know when that occurs.

It returns 1 on success and 0 otherwise. If the optional Error argument is provided, it is filled with an error code giving more detail about why the attach failed. See parxch.h for a list of the possible error codes. Pass NULL to ignore this parameter.

The ReleaseGps function removes the connection between the two drivers and should be called before the drivers are closed.

START AND STOP EXECUTION

C usage:

```
#include "pargps.h"

void ParGpsStart( DEVHANDLE handle );

void ParGpsStop( DEVHANDLE handle );
```

After successfully opening the device driver and initializing the PARGPS board, the next step is to start checking for the PPS signal and the serial NMEA messages by calling Start.

Once the PARxCH and PARGPS are started you can use the ReadPpsData and ReadSerialData functions to transfer the GPS data from the PARGPS software FIFO over to the PC. Of course, you will likely also want to use the ParXchReadDataWithGpsMark to transfer the PARxCH analog data from the PARxCH hardware FIFO over to the PC.

After you are done acquiring, call the Stop function and then the Close function to close the driver.

READ DATA

C usage:

```
#include "pargps.h"

unsigned int ParGpsReadPpsData(
       DEVHANDLE handle,
       void *PpsValues,
       unsigned int PpsNvalues
       int *Error,
       );

unsigned int ParGpsReadSerialData(
       DEVHANDLE handle,
       void *SerialValues,
       unsigned int SerialNvalues
       int *Error,
       );
```

ReadPpsData and ReadSerialData each copy their respective types of data from the PARGPS software FIFO to the PC where it is available for further processing. Although a void* type is used in the function declaration, the parameter should be an array of PPSDATA or SERIALDATA structures, respectively. See pargps.h for a definition of these structures.

To use these functions, you need an array to hold the PPS or serial data values and the dimension of that array. Each of these functions reads as much data as possible into its array up to Nvalues and returns the number of points actually read. This may be zero if no values of that type are ready yet.

The optional Error parameter is filled with 0 if all is well and with an error code if something has gone wrong. See pargps.h for a list of the possible error codes. You can ignore this parameter by passing NULL.

A typical code fragment for using ReadPpsData and ReadSerialData would be:

```
/* Declare arrays to contain acquired values. */
PpsValues = (PPSDATA *)malloc( Npps * sizeof( PPSDATA ) );
SerValues = (SERIALDATA *)malloc( Nser * sizeof( SERIALDATA ));

/* Open driver and start execution ... */
```

```
/* Acquire and process data. */
while ( 1 ) {
       Nremain = Nvalues;
       NextPt  = 0;

       /* Get Nvalues worth of data. */
       while ( Nremain > 0 ) {
              Newpts = ParXchReadDataWithGpsMark(
                                    ParXchHandle,
                                    &Values[NextPt],
                                    Nremain,
                                    &ErrPts );
              NewPps = ParGpsReadPpsData(ParGpsHandle,
                                    &PpsValues[NextPtPps],
                                    NremainPps,
                                    &ErrPps );
              NewSer = ParGpsReadSerialData(
                                    ParGpsHandle,
                                    &SerValues[NextPtSer],
                                    NremainSer,
                                    &ErrSerial );
              /* Update Nremains for New points read ... */

              /* Check for errors. */
              if ( ErrPps != ErrSerial != PARGPS_ERROR_NONE )
                     Error("Error %s\n",PARGPS_ERROR_MSG[Err]);
              }

       /* Do something with data. */
       SaveValuesToDisk( Values, PpsValues, SerialValues );
```

See simp.c in the PARxCH simple directory for complete code listings showing how to use these functions.

COUNTER FREQUENCY AND LIBRARY REV

```
C usage:
       #include "pargps.h"

       int ParGpsGetCounterFrequency( DEVHANDLE Handle, long *Freq );

       void ParGpsGetRev( int *Rev );
```

GetCounterFrequency fills the Freq argument with information about how often the 64 bit counter is incremented. This counter, maintained by the OS, is used to mark the PPS and data sample times. So its frequency must be known in order to properly convert counter values into times.

GetRev fills the Rev argument with an integer representing the current revision number of the PARGPS library. For example, a value of 201 indicates version 2.01.

APPENDIX A: ELECTRICAL SPECS

POWER SUPPLY REQUIREMENTS:

The PARGPS system requires a minimum off board power supply of at least 9vdc or vac at 130ma (Rev C) or 85 ma (Rev D) of current for reliable operation. The maximum off board power supply is approximately 24 volts.

A 9vdc 500ma unregulated wall transformer is typically used with the system. When loaded at 130ma, this will supply more than 12vdc of unregulated power. More than enough to adequately power the PARGPS.

Many users will be supplied with a 2.1mm daisy chain cable so a single wall transformer can be shared between the PARGPS and another SR product. The two 2.1mm connectors on the back panel of the PARGPS enclosure are in parallel. See the circuit diagrams for more details.

AC power is acceptable for off board power and will be rectified by the on board circuitry. 9vac wall or chassis mount transformers are perfectly fine with no degradation in PARGPS performance.

On the board itself, off board power is rectified and regulated to +5vdc as required by the on board circuitry. Running at higher off board voltages will increase the heat dissipation in the on board regulator. Use off board power supplies of 12 volts or less to minimize power dissipation.

SERIAL CABLE REQUIREMENTS:

The NMEA messages from the GPS receiver are communicated to the PC over an RS232 serial port. The SR PARGPS system has been designed to be used with serial cables that have all 9 wires connected straight through. The serial cable supplied with the system meets this requirement.

Serial cables that do not include all 9 wires but only the RX/TX pair or are "null modem" will not work. Do not cross RX/TX and make sure RTS and ground are connected. See the circuit diagram for more details.

In addition to NMEA messages, the GPS receiver generates a PPS signal, which is available on a DB15 connector on the PARGPS front panel. This PPS signal is designed to be connected from the PARGPS front panel to the digital IO connector on one of the SR A/D

boards with the supplied DB15 ribbon cable. The length of the supplied ribbon cable works best if the PARGPS is stacked above or below the SR A/D board.

ACCURACY OF PPS SIGNAL:

The PPS signal ultimately generates an interrupt to the PC. We have found typical PC interrupt latency to be about 10 microseconds. However, this latency may be longer in some cases if the system is undergoing heavy interrupt activity. Heavy use of interrupts by other drivers and programs may degrade PPS performance.

Monitor the regularity of the 64 bit PPS count interval spacing as recorded in a DAT file to determine the impact of interrupts on your system. If the number of counts from one PPS mark to another deviates wildly, then your system is suffering exceptional interrupt activity and you should investigate.

INDEX

SR GPS BOARD BLOCK DIAGRAM OVERVIEW

TRIMBLE ACEx GPS MODULE

SCHEMATIC PAGE 2
SMB ANTENNA CONNECTOR ON REAR PANEL

(100)

NEMA MESSAGE / PC RS232 INTERFACE

SCHEMATIC PAGES 3-5
DB9 SERIAL CABLE TO PC, WITH ACEx TXB
NEMA MESSAGE STRINGS EMITTED EVERY SECOND
COARSE HOUR:MIN:SEC TIMING INFO
LAT / LON COORDINATE INFO

(200)

PPS PULSE PER SECOND / PARxCH INTERFACE

SCHEMATIC PAGES 6-7
DB15 CABLE TO PARxCH DIGITAL IO
ONE SYNC PULSE PER SECOND
HIGH ACCURACY EDGE FOR FINE TIME INFO
RED LED ON FRONT PANEL SHOWING PPS

(300)

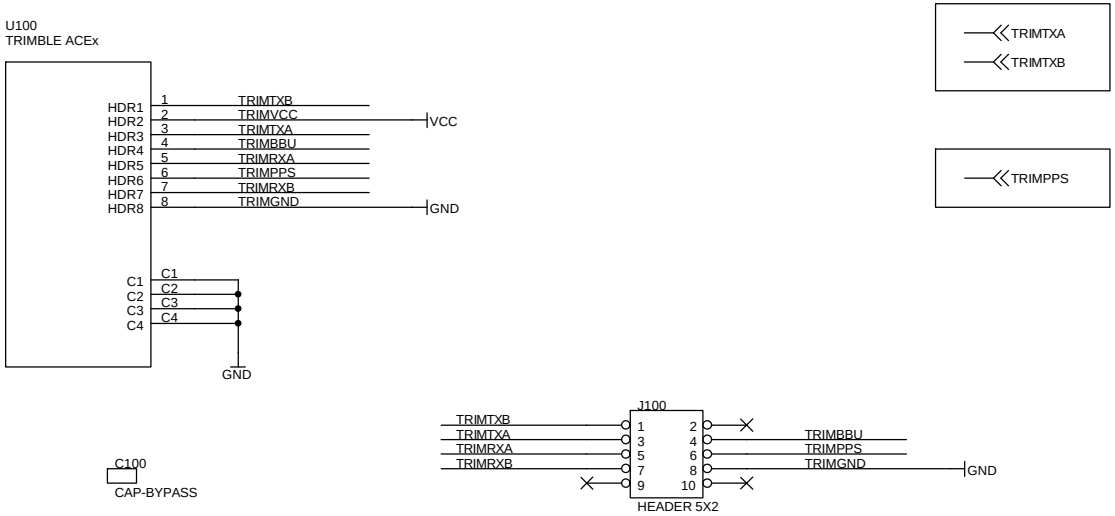
POWER SUPPLY

SCHEMATIC PAGES 8-9
9 TO 24 VAC/VDC ACCEPTABLE INPUTS
2.1MM WALL TRANSFORMER AND 4 PIN MOLEX CONNECTORS
ON BOARD REGULATED TO 5v

(400)

Title: SR GPS OVERVIEW		
Size: A	Document Number: GPS.DSN, (c) SYMMETRIC RESEARCH, 2002	Rev: C
Date: Wednesday, June 26, 2002	Sheet 1 of 9	

TRIMBLE ACEx GPS MODULE



NOTES:

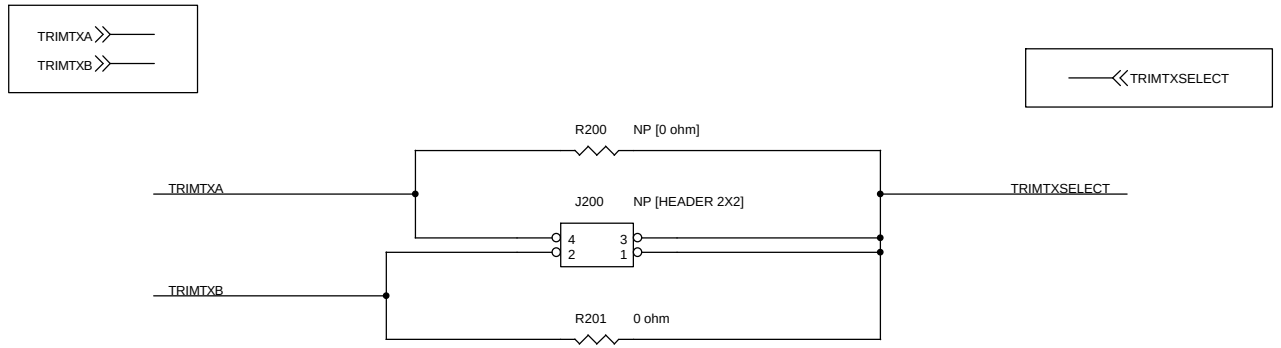
TRIMBLE ACEx MOUNTED UPSIDE DOWN WITH 8 PIN 2mm HEADER PLUGGED INTO FEMALE SOCKET ON SR GPS BOARD

C1-C4 ARE CORNER MOUNTING HOLES ON ACE3

J100 IS A 0.1" MALE HEADER FOR FACTORY PROGRAMMING ONLY

Title: TRIMBLE GPS		
Size: A	Document Number: GPS.DSN, (c) SYMMETRIC RESEARCH, 2002	Rev: C
Date: Friday, July 05, 2002	Sheet 2 of 9	

SELECTION OF TRIMBLE TXA/B MAPPING ONTO PC RS232 SERIAL PORT



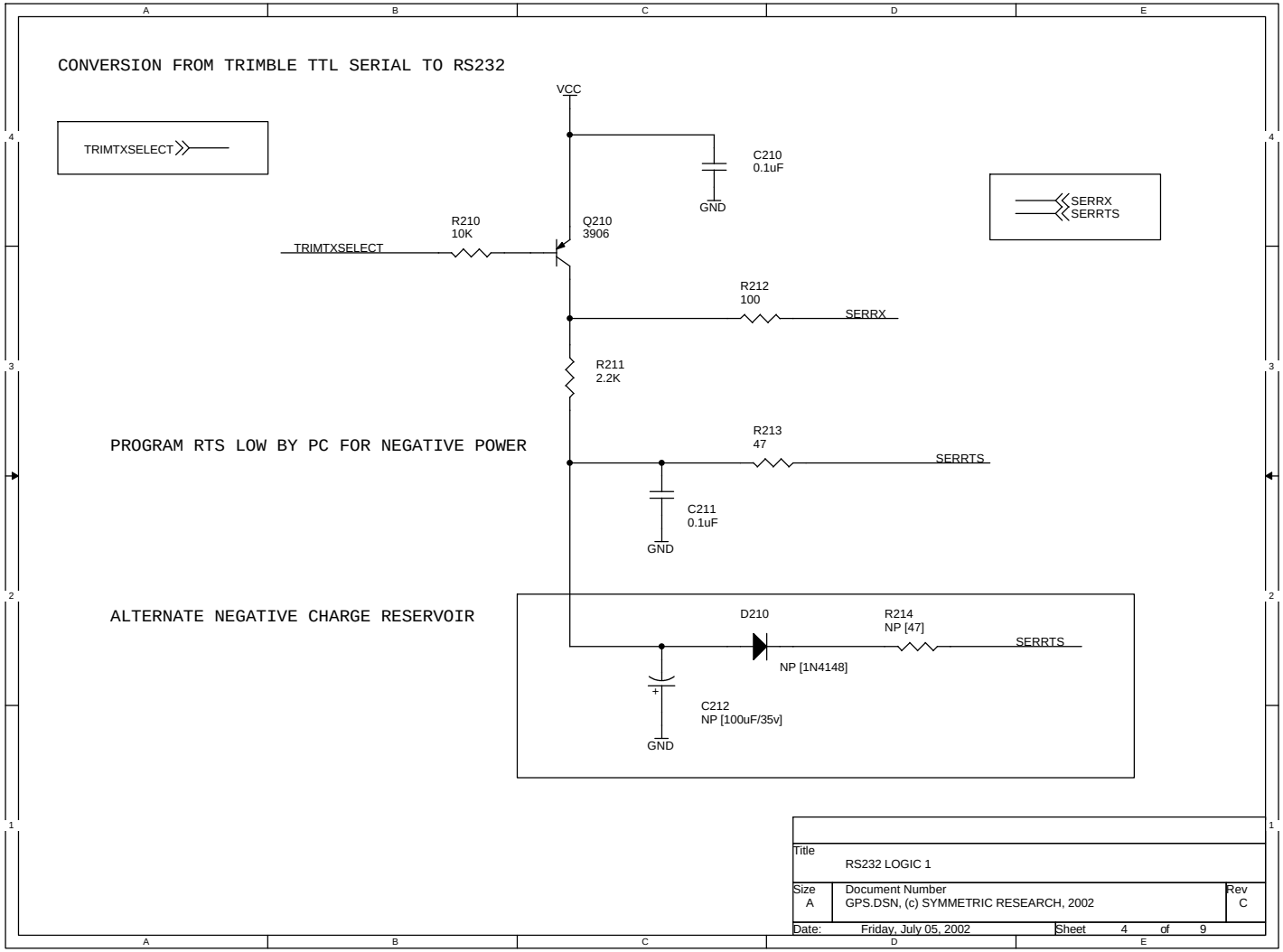
NOTES:

TRIMBLE GPS MODULE HAS TWO SERIAL OUTPUT STREAMS, TXA, TXB

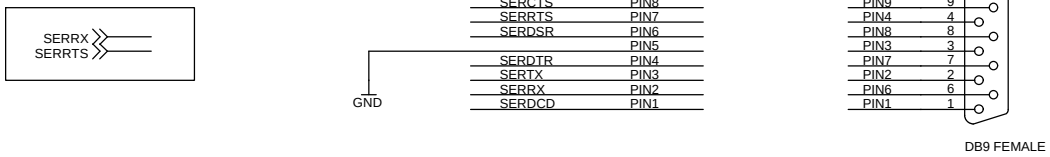
NORMALLY TXB HARDWIRED TO PC RS232 WITH 0 ohm RESISTOR, FOR NMEA OUTPUT

DO NOT INADVERTANTLY CONNECT TXA AND TXB TOGETHER BY PLACING JUMPERS ON BOTH POSITIONS OF J200

Title: RS232 LOGIC 0		
Size: A	Document Number: GPS.DSN, (c) SYMMETRIC RESEARCH, 2002	Rev: C
Date: Friday, July 05, 2002	Sheet 3 of 9	


CONVERSION FROM TRIMBLE TTL SERIAL TO RS232
VCC
TRIMTXSELECT
C210
0.1uF
GND
SERRX
SERRTS
R210
10K
Q210
3906
TRIMTXSELECT
R212
100
SERRX
R211
2.2K
PROGRAM RTS LOW BY PC FOR NEGATIVE POWER
R213
47
SERRTS
C211
0.1uF
GND
ALTERNATE NEGATIVE CHARGE RESERVOIR
D210
R214
NP [47]
SERRTS
NP [1N4148]
C212
NP [100uF/35v]
GND
Title
RS232 LOGIC 1
Size
A
Document Number
GPS.DSN, (c) SYMMETRIC RESEARCH, 2002
Rev
C
Date:
Friday, July 05, 2002
Sheet
4
of
9

DB9 CONNECTOR TO PC RS232 SERIAL PORT



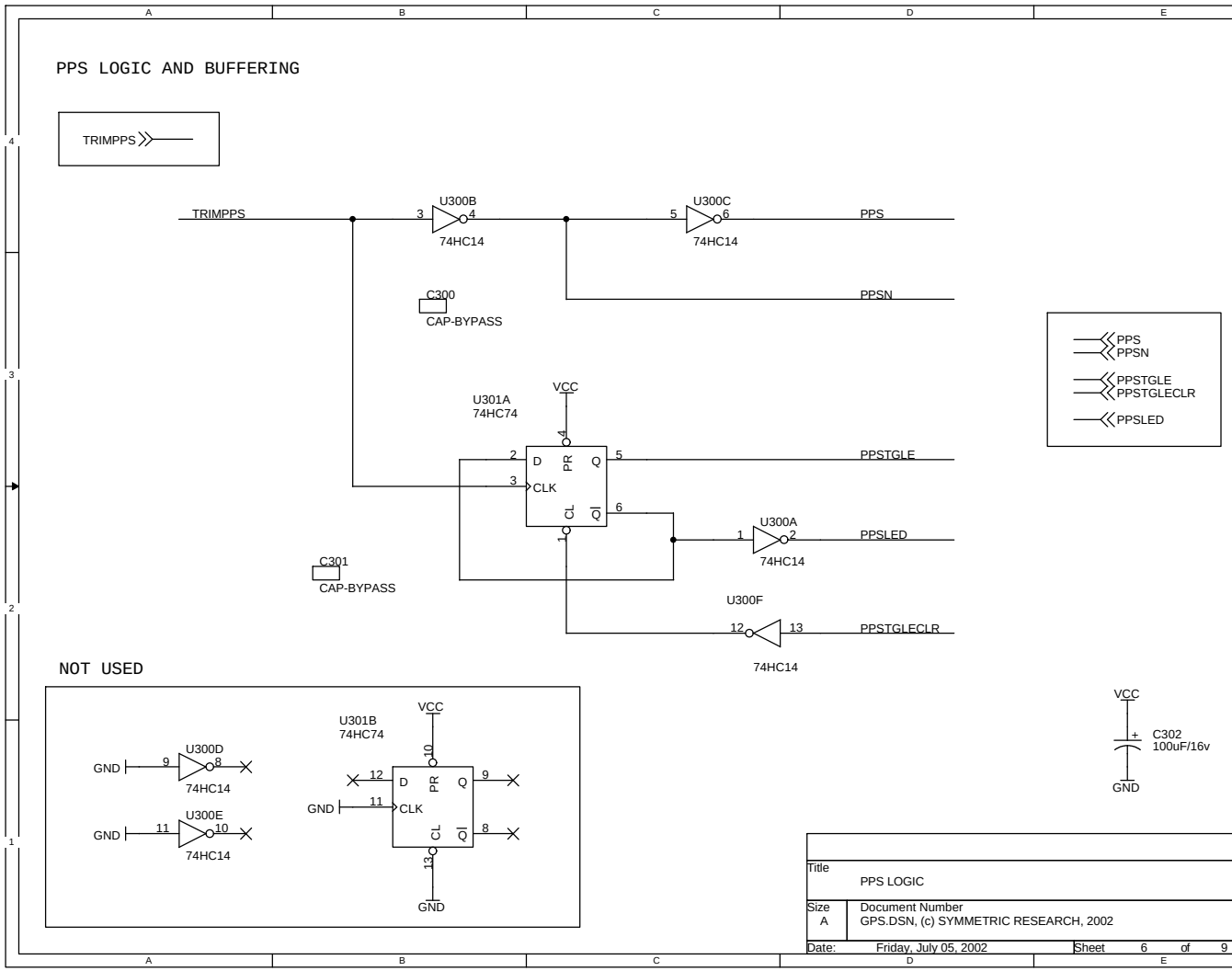
NOTES:

INTERFACE ONLY USES SERRX, SERRTS, AND GND. ALL OTHERS ARE NO CONNECT.

SERRTS USED ONLY TO DERIVE NEGATIVE POWER / VOLTAGE FOR RS232 SIGNAL LEVELS.

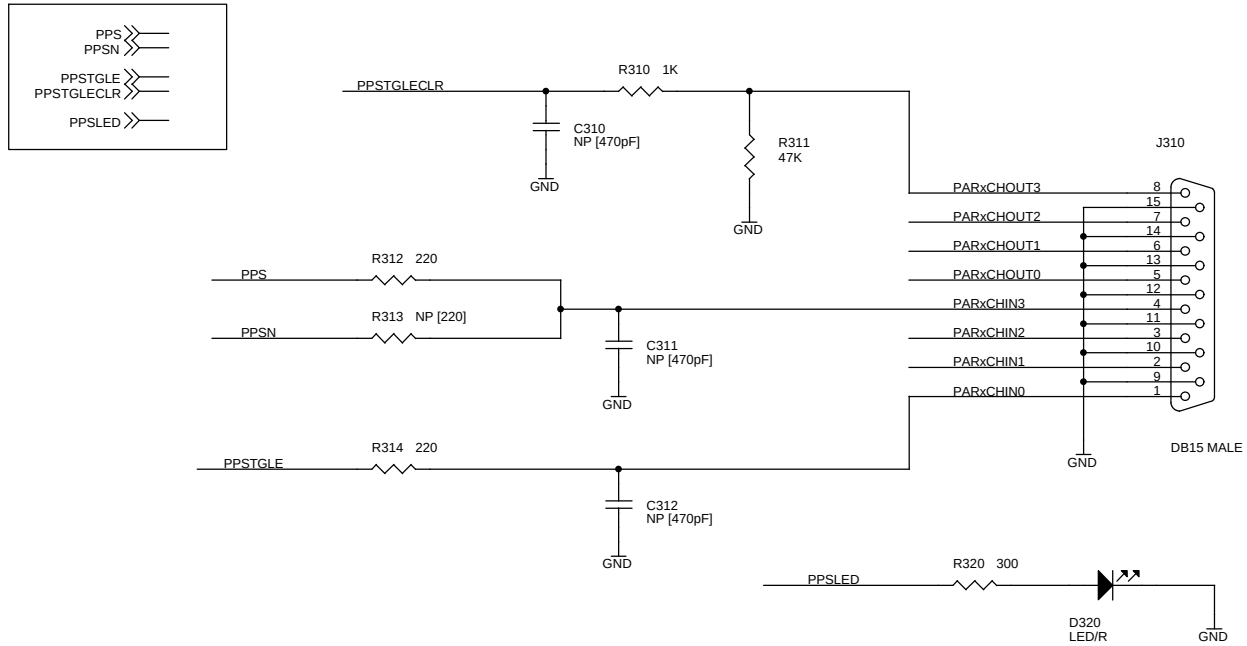
Title: RS232 DB9		
Size: A	Document Number: GPS.DSN, (c) SYMMETRIC RESEARCH, 2002	Rev: C
Date: Friday, July 05, 2002	Sheet 5 of 9	

PPS LOGIC AND BUFFERING



Title	PPS LOGIC	
Size A	Document Number GPS.DSN, (c) SYMMETRIC RESEARCH, 2002	Rev C
Date: Friday, July 05, 2002	Sheet 6 of 9	

PPS PARxCH DIGITAL IO CONNECTOR



NOTES:

PPS = 10 microsecond WIDE PULSE, LEADING RISING EDGE -> MARK OF UTC SECOND
PPSN = NEGATED (inverted) PPS

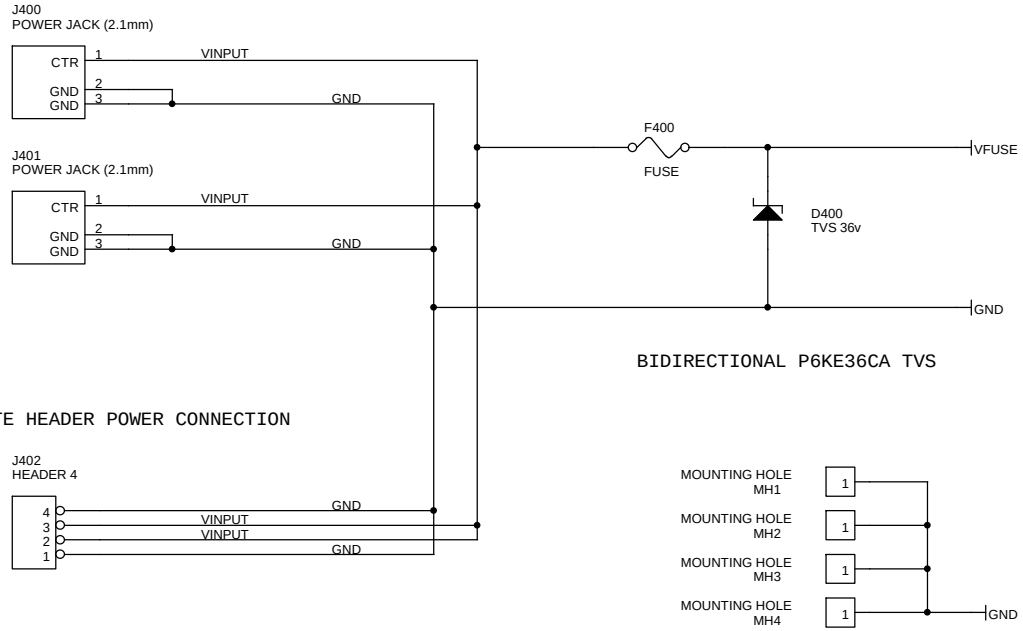
PPSTGLE = TOGGLING PSS, HIGH FOR 1 second, LOW FOR 1 second, ...
PPSTGLECLR = CLEAR PPSTGLE FLIP FLOP

PPSLED = BUFFERED PPSTGLE TO DRIVE RED LED

Title: PPS DB15		
Size: A	Document Number: GPS.DSN, (c) SYMMETRIC RESEARCH, 2002	Rev: C
Date: Friday, July 05, 2002	Sheet 7 of 9	

POWER SUPPLY INPUT CONNECTORS

2.1MM MALE WALL TRANS JACK



BIDIRECTIONAL P6KE36CA TVS

ALTERNATE HEADER POWER CONNECTION

NOTES:

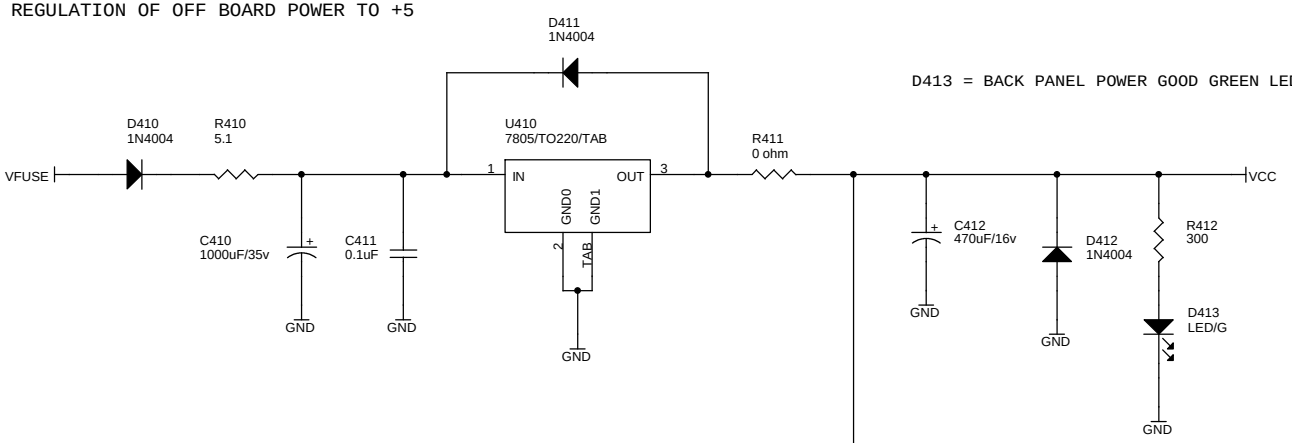
ACCEPTABLE OFF BOARD POWER SUPPLIES = 9 to 24 VAC or VDC

9 VDC @ 500ma PREFERRED

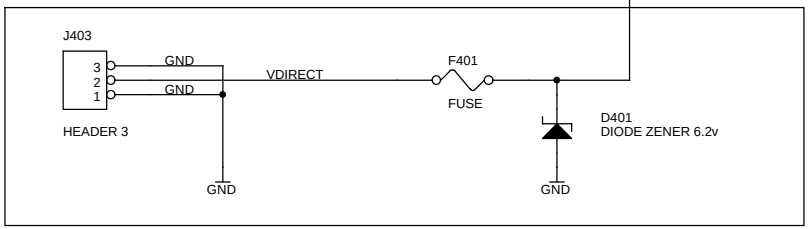
2.1mm POWER CONNECTORS PARALLELED FOR CHAINING TO OTHER SR DEVICES

Title: POWER SUPPLY 0		
Size: A	Document Number: GPS.DSN, (c) SYMMETRIC RESEARCH, 2002	Rev: C
Date: Friday, July 05, 2002	Sheet 8 of 9	

REGULATION OF OFF BOARD POWER TO +5



DIRECT +5v POWER CONNECTION:



*** CAUTION ***

DIRECT +5v POLARITY REVERSALS OR OVERVOLTAGES MAY DAMAGE BOARD

REMOVE R411 WHEN USING DIRECT +5 TO REDUCE CAPACITIVE LOADING

Title: POWER SUPPLY 1		
Size: A	Document Number: GPS.DSN, (c) SYMMETRIC RESEARCH, 2002	Rev: C
Date: Friday, July 05, 2002	Sheet 9 of 9	

SR GPS BOARD BLOCK DIAGRAM OVERVIEW

GARMIN GPS18 PUCK CONNECTOR

SCHEMATIC PAGE 2
DB9 CONNECTOR TO GARMIN GPS18 PUCK

(100)

NEMA MESSAGE / PC RS232 INTERFACE

SCHEMATIC PAGES 3-4
DB9 RS232 SERIAL CABLE TO PC
NEMA MESSAGE STRINGS EMITTED EVERY SECOND
COARSE HOUR:MIN:SEC TIMING INFO
LAT / LON COORDINATE INFO

(200)

PPS PULSE PER SECOND / PARxCH INTERFACE

SCHEMATIC PAGES 5-6
DB15 CABLE TO PARxCH DIGITAL IO
ONE PULSE PER SECOND TIMING SIGNAL
HIGH ACCURACY EDGE FOR FINE TIME INFO
RED LED ON FRONT PANEL SHOWING PPS

(300)

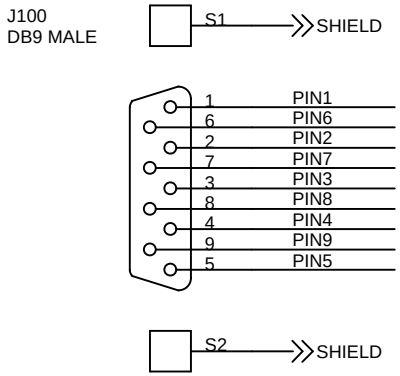
POWER SUPPLY

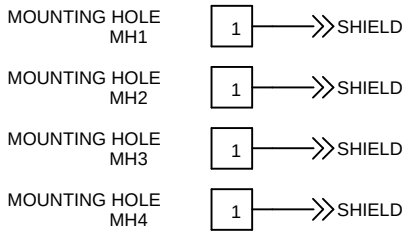
SCHEMATIC PAGES 7-8
9 TO 24 VAC/VDC ACCEPTABLE INPUTS
2.1MM WALL TRANSFORMER AND 4 PIN MOLEX CONNECTORS
ON BOARD REGULATED TO 5v

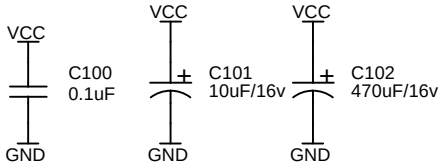
(400)

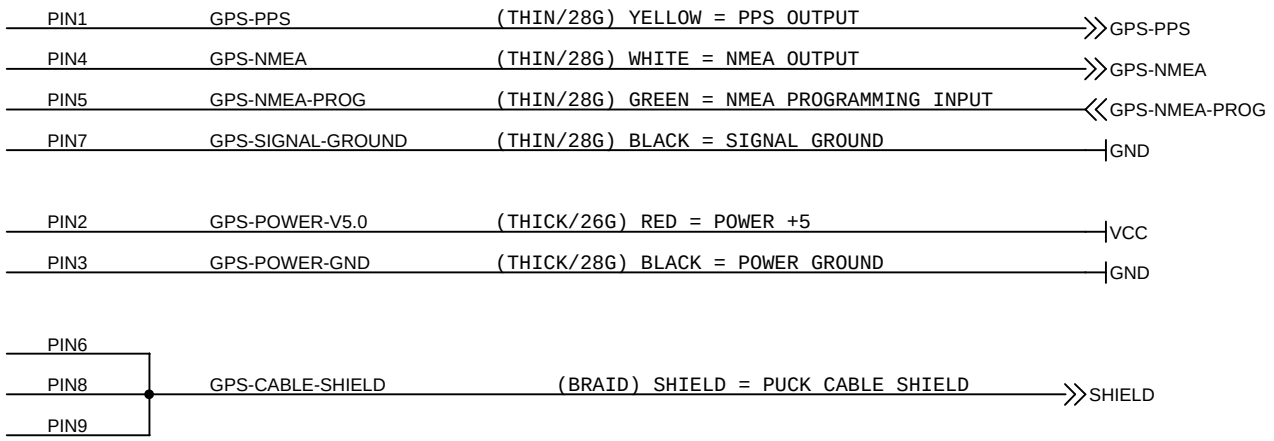
Title: 01 OVERVIEW		
Size: A	Document Number: PARGPS.DSN, (c) SYMMETRIC RESEARCH, 2007	Rev: D
Date: Monday, January 21, 2008	Sheet 1 of 8	

DB9 MALE CONNECTOR TO GARMIN GPS 18 LVC



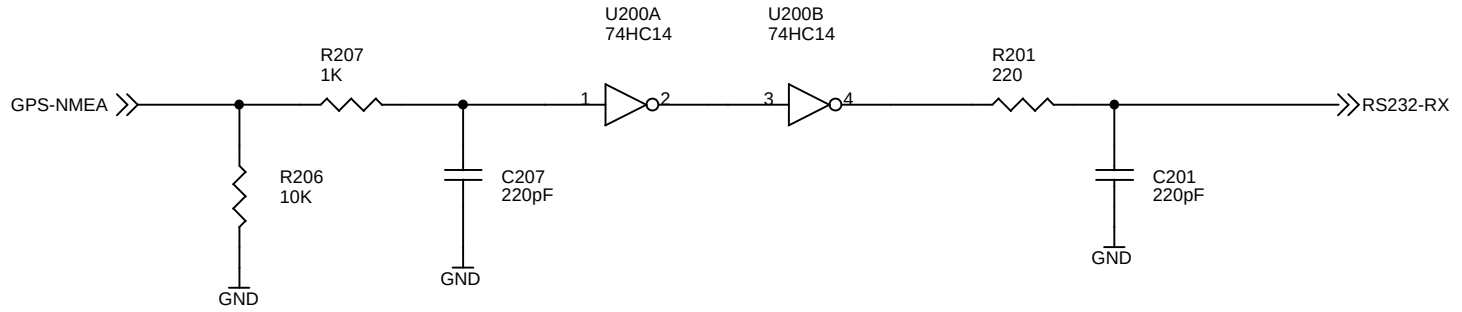






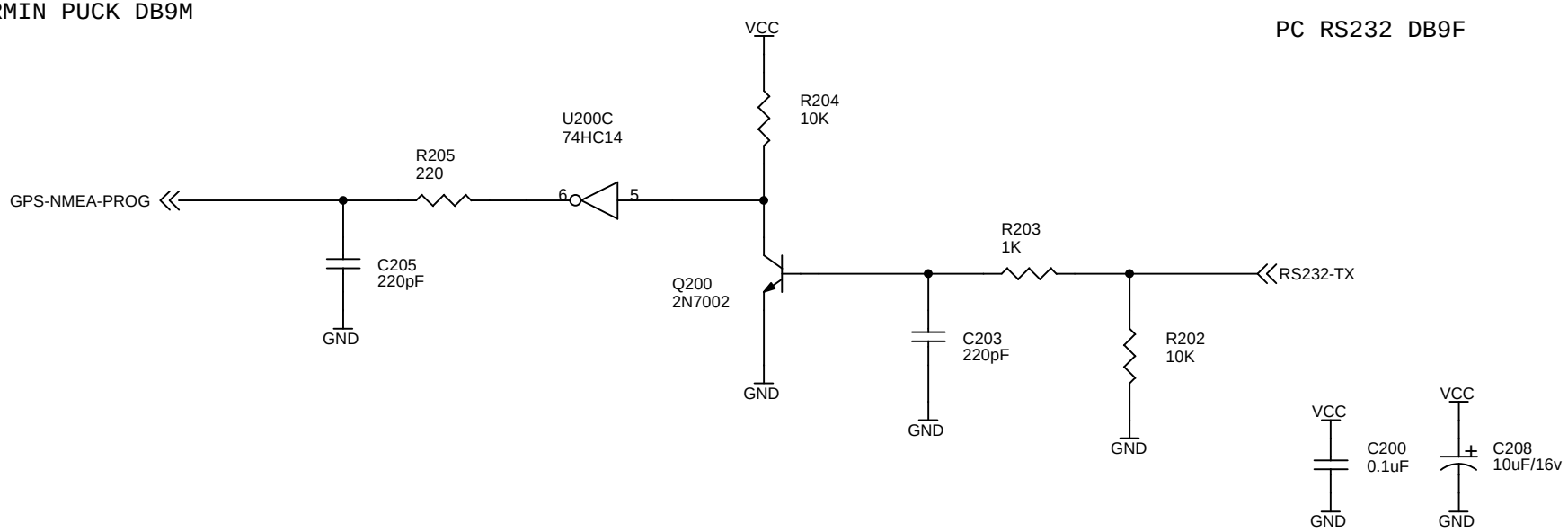
Title	02 GPS PUCK CONNECTOR	
Size A	Document Number PARGPS.DSN, (c) SYMMETRIC RESEARCH, 2007	Rev D
Date: Monday, January 21, 2008	Sheet 2 of 8	

NMEA RS232 BUFFERING

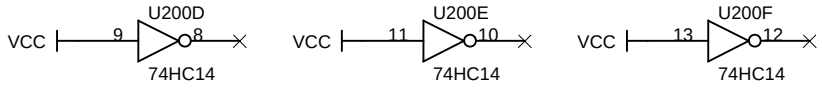


GARMIN PUCK DB9M

PC RS232 DB9F

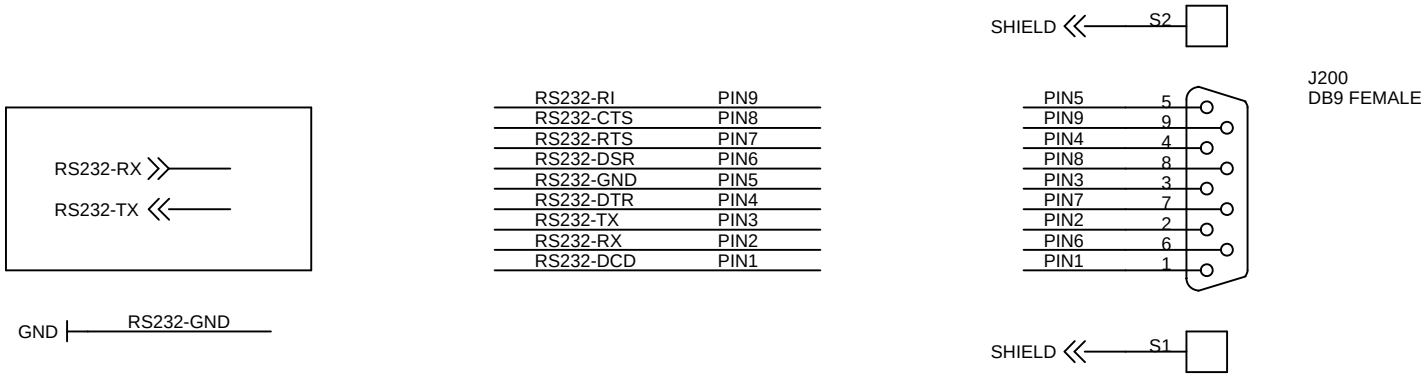


NOT USED



Title	03 NMEA RS232 BUFFERS	
Size A	Document Number PARGPS.DSN, (c) SYMMETRIC RESEARCH, 2007	Rev D
Date: Monday, January 21, 2008	Sheet 3 of 8	

DB9 CONNECTOR TO PC RS232 SERIAL PORT



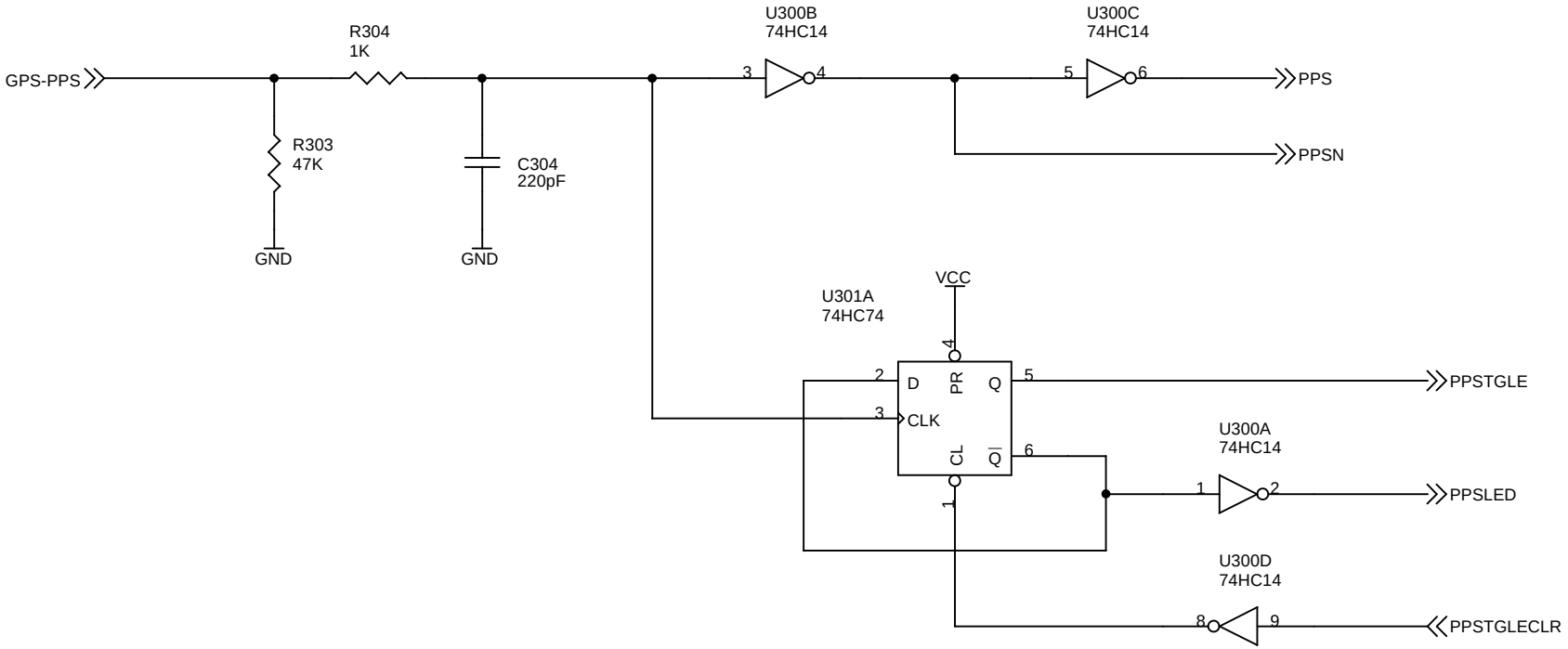
NOTES:

INTERFACE USES ONLY RS232-RX, RS232-TX, AND GND. ALL OTHERS ARE NO CONNECT.

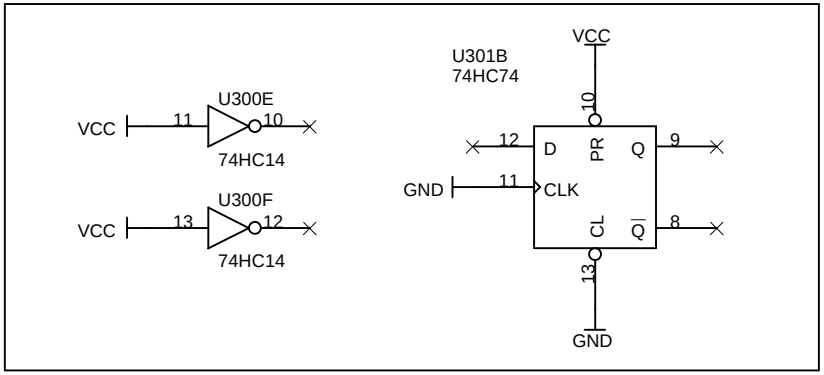
RS232-RX = DB9F PIN 2 = INPUT TO PC
RS232-TX = DB9F PIN 3 = OUTPUT FROM PC

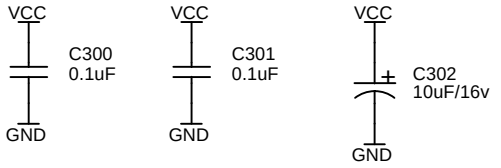
Title: 04 RS232 NMEA CONNECTOR		
Size: A	Document Number: PARGPS.DSN, (c) SYMMETRIC RESEARCH, 2007	Rev: D
Date: Monday, January 21, 2008	Sheet 4 of 8	

PPS LOGIC AND BUFFERING



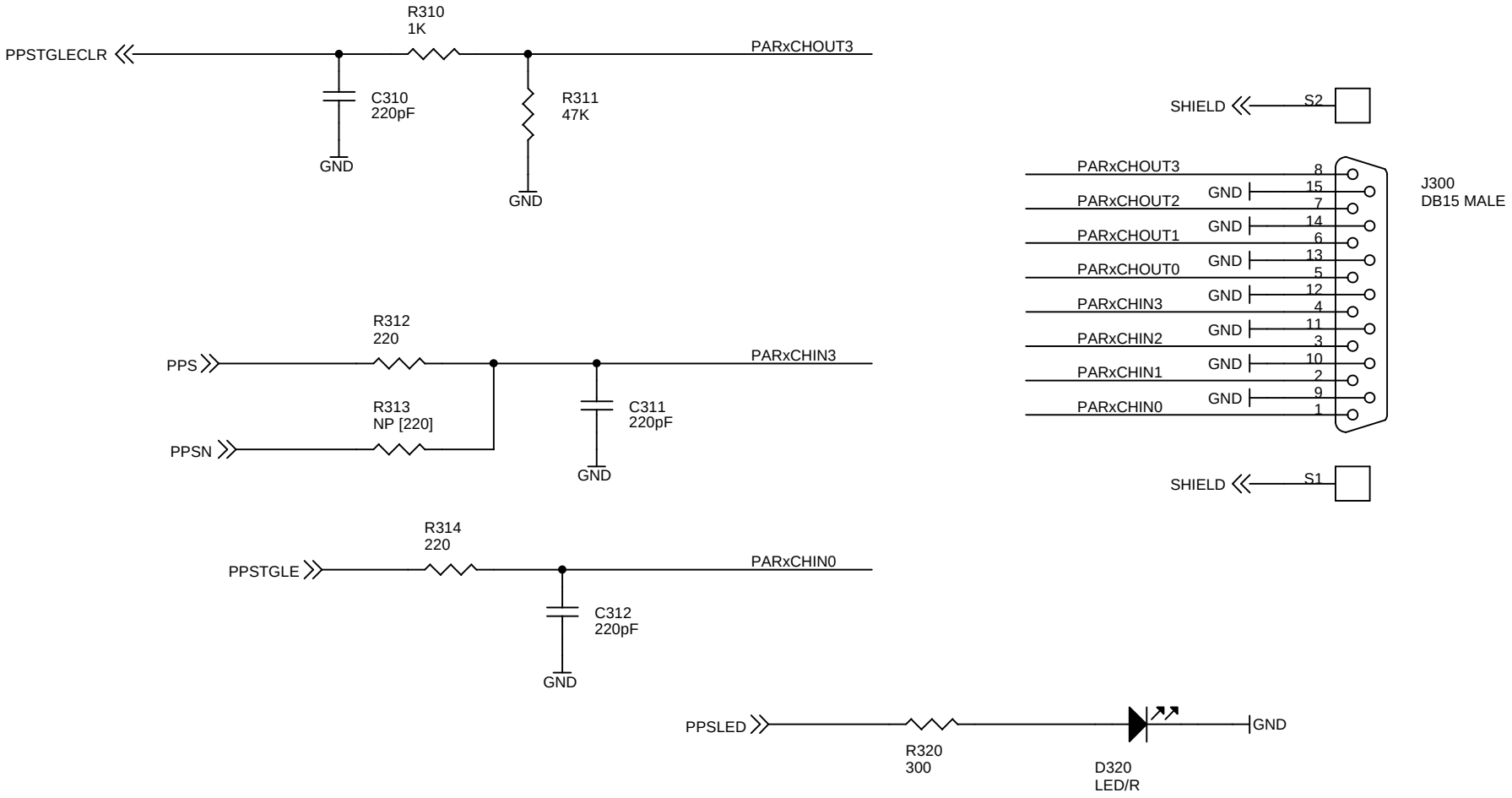
NOT USED





Title: 05 PPS PARXCH BUFFERS		
Size: A	Document Number: PARGPS.DSN, (c) SYMMETRIC RESEARCH, 2007	Rev: D
Date: Monday, January 21, 2008	Sheet 5 of 8	

PPS PARxCH DIGITAL IO CONNECTOR



NOTES:

PPS = NORMALLY LOW, RISING EDGE -> MARK OF UTC SECOND
PPSN = NEGATED (inverted) PPS

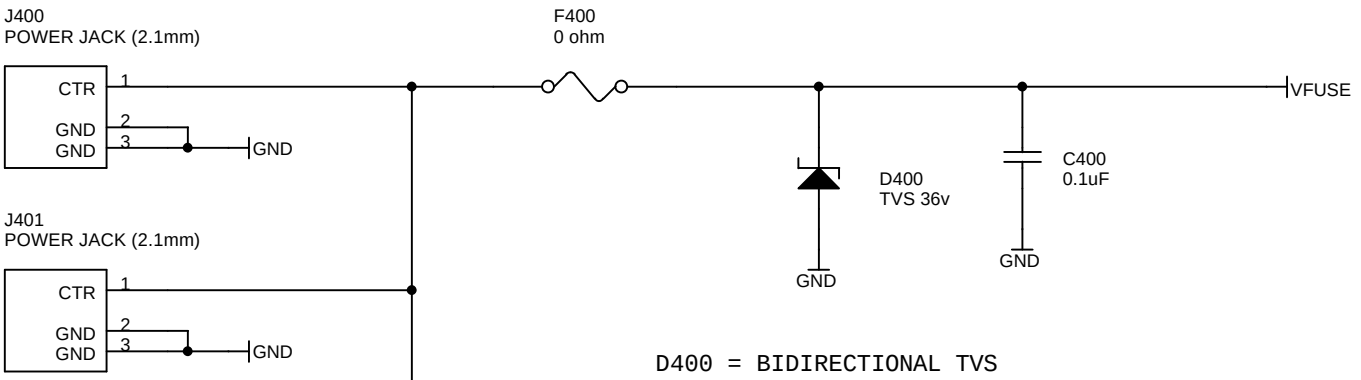
PPSTGLE = TOGGLING PSS, HIGH FOR 1 second, LOW FOR 1 second, ...
PPSTGLECLR = CLEAR PPSTGLE FLIP FLOP

PPSLED = BUFFERED PPSTGLE TO DRIVE RED LED

Title: 06 PPS PARXCH CONNECTOR		
Size: A	Document Number: PARGPS.DSN, (c) SYMMETRIC RESEARCH, 2007	Rev: D
Date: Monday, January 21, 2008	Sheet 6 of 8	

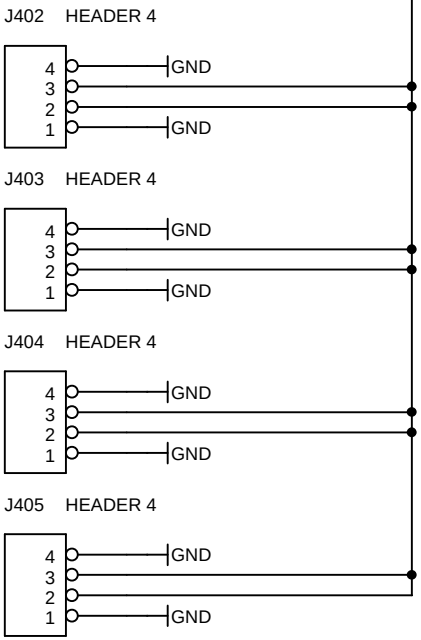
POWER SUPPLY CONNECTORS

2.1MM MALE WALL TRANS JACK



D400 = BIDIRECTIONAL TVS

ALTERNATE MOLEX POWER HEADERS



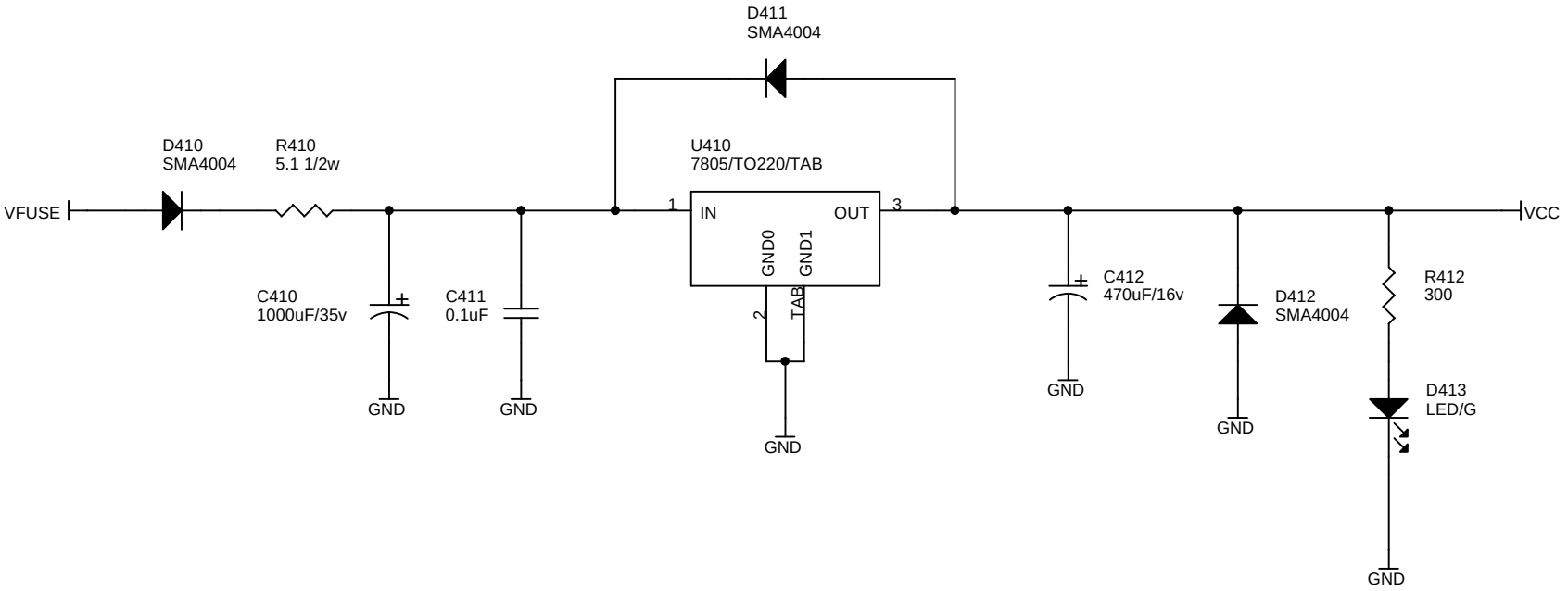
ACCEPTABLE OFF BOARD POWER SUPPLIES = 9 to 24 VAC or VDC

9 VDC @ 500ma PREFERRED

POWER CONNECTORS PARALLELED FOR CHAINING TO OTHER SR DEVICES

Title: 07 POWER SUPPLY 0		
Size: A	Document Number: PARGPS.DSN, (c) SYMMETRIC RESEARCH, 2007	Rev: D
Date: Monday, January 21, 2008	Sheet 7 of 8	

+5 VOLT POWER SUPPLY REGULATION



D413 = BACK PANEL POWER ON LED

Title: 08 POWER SUPPLY 1		
Size: A	Document Number: PARGPS.DSN, (c) SYMMETRIC RESEARCH, 2007	Rev: D
Date: Monday, January 21, 2008	Sheet 8 of 8	