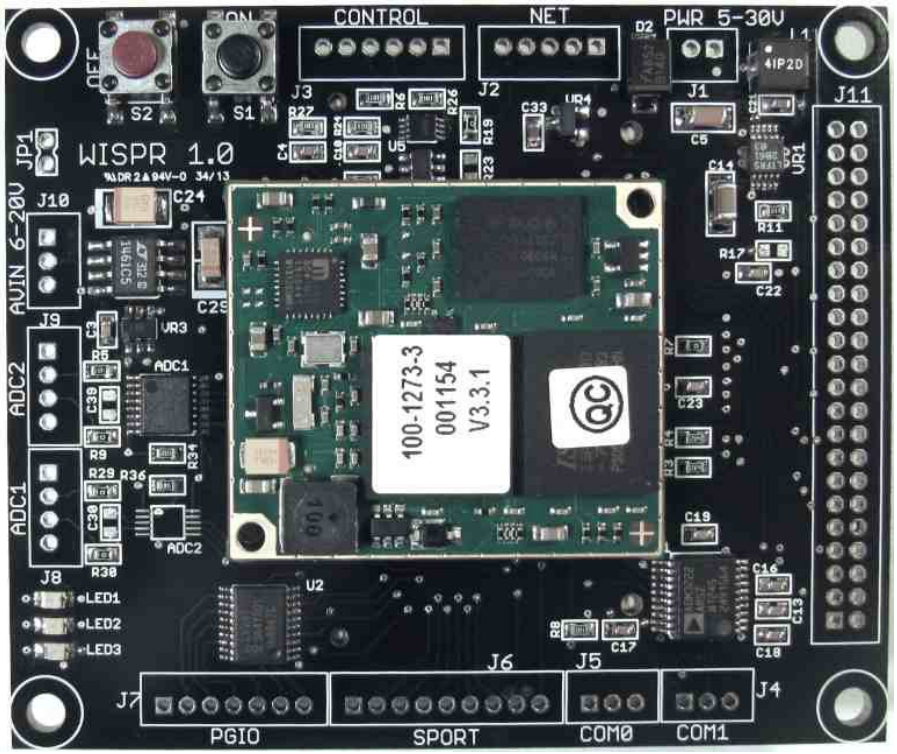
USERS MANUAL

WISPR V1.1



Embedded Ocean Systems
Seattle WA 98112

September, 2014

Table of Contents

List of Figures

List of Tables

1. Introduction

The WISPR system is a highly capable mixed signal motherboard designed for space limited and low power embedded digital signal processing applications. The motherboard provides an Analog Devices BF537E Blackfin core module, two ADC signal channels, Ethernet, two RS-232 serial channels, and several external data bus interfaces. The WISPR system also provides an integrated CF Card for mass data storage and an IDE/PATA bus interface for external drives.

The system supports software development in the uCLinux operating system (http://blackfin.uclinux.org) and VisualDSP++ (www.analog.com). The system uses a customized uCLinux kernel that will boot from flash memory on power-up into a fully functional embedded OS with device drivers for the specific hardware and optimized DSP functions. Full documentation for the open source umClinux environment is available at http://docs.blackfin.uclinux.org/doku.php.

The WISPR system provides signal interfaces for:

- Digital Supply Power
- 10/100 MBit Ethernet
- Power control and I^2C interface
- Two RS-232 UARTS
- SPORT Bus Interface
- Buffered Digital I/O Lines
- Two Differential Input ADC Channels sampled at 125 or 62.5 kSPS
- Analog Supply Power
- IDE (PATA) Bus Interface.

2. Specifications

2.1 Electrical Specification

Symbol	Parameter	Min	Typical	Max	Unit
VIN	Input Supply Voltage	4.2	12-24	30	Volts
I_IN	VIN Current	TBD	TBD	TBD	mA
AVIN	Analog Input Supply	5.5	12-24	20	Volts
AI_IN	AVIN Current	TBD	TBD	TBD	mA

Table 1: Electrical Specifications

2.2 Mechanical Specification

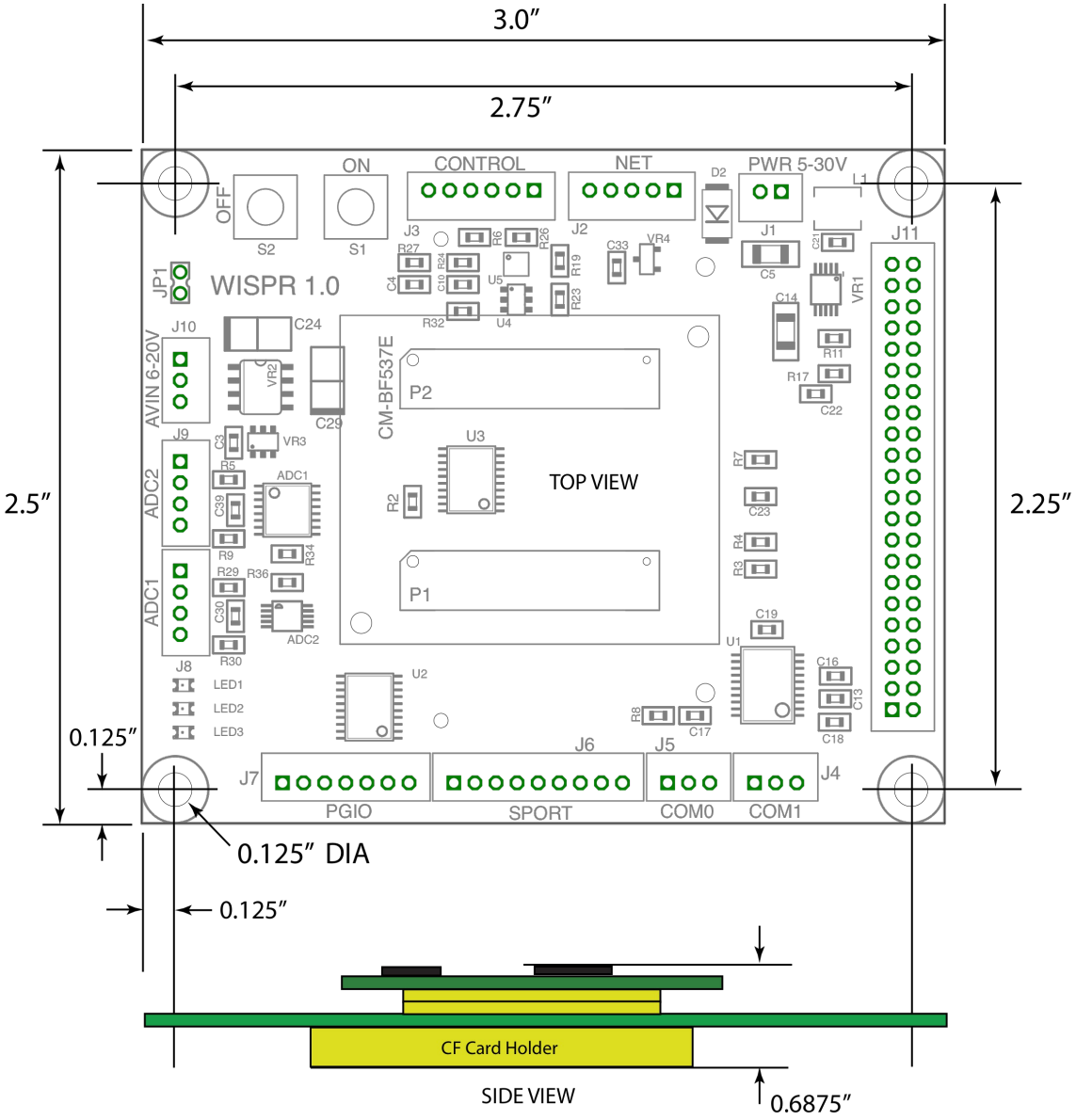


Illustration 1: Top and side view with outline dimensions (inches). Connectors are not shown on side view.

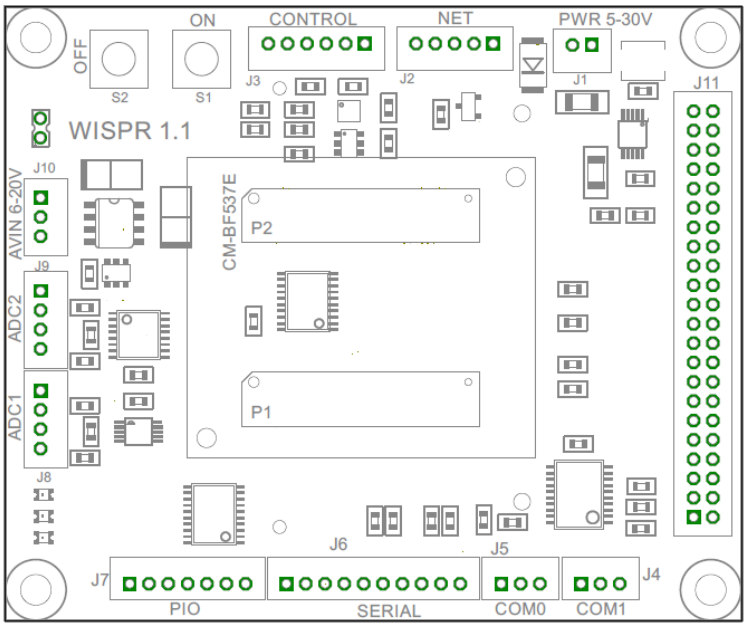


Figure 1: WISPR V1.1 Top assembly drawing

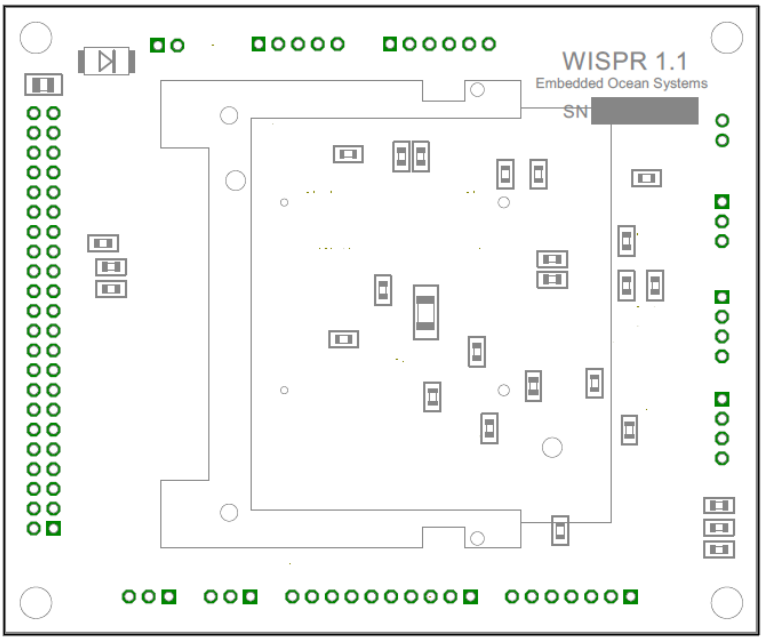


Figure 2: WISPR V1.1 Bottom assembly drawing

Connector pin 1 is indicated with a square solder pad on all connectors

2.3 Connector Specification

Connectors are located on the top of the WISPR motherboard and labeled J1 through J11, as illustrated in Figure 2. All connectors except J11 are 2mm pitch, PH series, crimp style connectors: http://www.jst-mfg.com/product/pdf/eng/ePH.pdf.

The CF card and IDE connector (J11) are located on the bottom side of the board. Each connector is marked with a number and a name, for example J1 and PWR. Pin 1 is indicated with a ***square solder pad*** on all connectors.

2.3.1 Main Power Connector

Connector J1 provides the input supply voltage for the main 3.3V digital supply switching regulator. The input is diode protected against reverse polarity.

For proper control operation, the main power connector should be connected before supplying control signals.

J1	**PWR**	Type	Level	Symbol
1	Ground	GND		GND
2	Input Supply Voltage	PWR	5 to 30 VDC	VIN

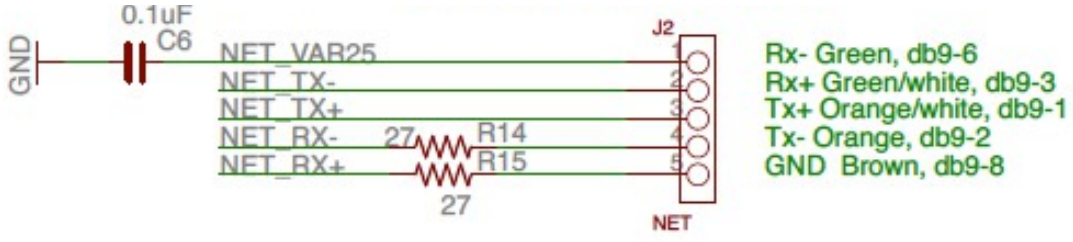
Table 2: Main power connector

2.3.2 Ethernet Connector

Connector J2 provides 10/100 Base-TX Ethernet. These signals are connected directly to the Ethernet transceiver and are not transformer isolated.

J2	**NET**	TYPE	LEVEL	SYMBOL
1	Transformer Voltage Reference	PWR		NET_VAR25
2	TX- (Orange)	OUTPUT		NET_TX-
3	TX+ (Orange/White)	OUTPUT		NET_TX+
4	RX- (Green)	INPUT		NET_RX-
5	RX+ (Green/White)	INPUT		NET_RX+

Table 3: Ethernet Connector



2.3.3 Power Control Connector

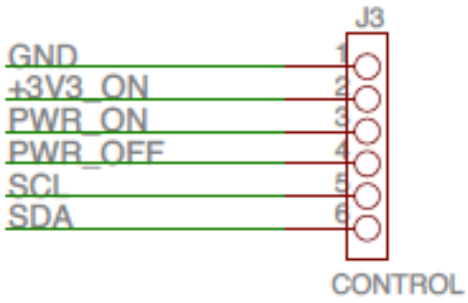
Connector J3 provides TTL level (5V tolerant) control signals to turn main regulated power ON/OFF. An active high pulse on PWR_ON will turn the main 3.3V regulator ON. An active high pulse on PWR_OFF will turn the main 3.3V regulator OFF. An active high pulse duration of greater than 1 msec is recommended.

For proper control operation, the main power connector should be connected before supplying control signals.

In addition to the power control lines, the I^2C bus lines are made available to control external devices.

J3	**CONTROL**	TYPE	LEVEL	SYMBOL
1	Ground	GND		GND
2	3.3V Regulator ON	OUTPUT	0 to 5 V	+3V3_ON
3	Regulator ON Pulse – Active High	INPUT	0 to 5 V	PWR_ON
4	Regulator OFF Pulse – Active High	INPUT	0 to 5 V	PWR_OFF
5	I^2C Bus SCL	I/O	3.3 V	SCL
6	I^2C Bus SDA	I/O	3.3 V	SDA

Table 4: Control Connector



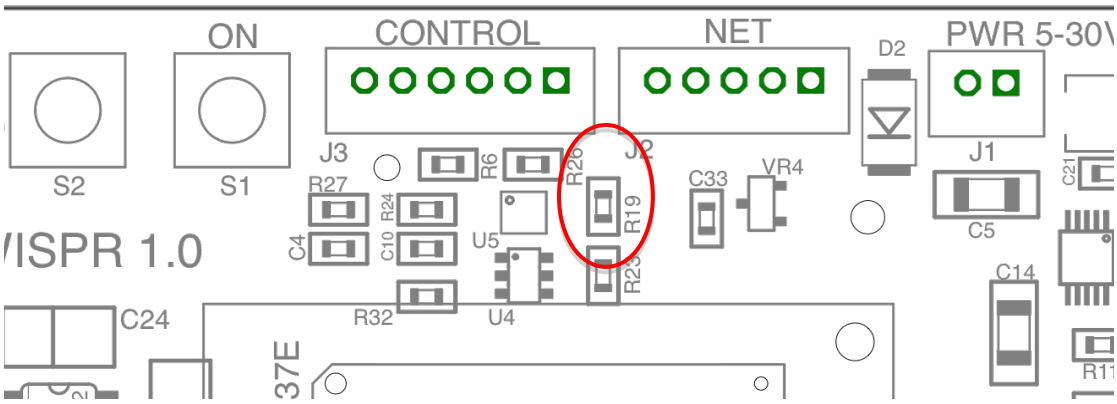


Figure 3: Remove R19 to disable ON/OFF control on J3

Disabling ON/OFF Control

To disable the ON/OFF control signals (PWR_ON and PWR_OFF) remove the resistor R19, as shown in Figure 3. With R19 removed the WISPR system will power ON immediately when voltage is applied to the PWR connector J1. Pin 2 (+3.3V_ON) of J3 can then becomes and input and can be used as an ON/OFF control line. Holding +3V3_ON low will power OFF the WISPR and the system will power ON when held high.

2.3.4 RS-232 Serial Port Connectors

Connector J4 provide RS-232 transceiver signals for UART channel COM1.

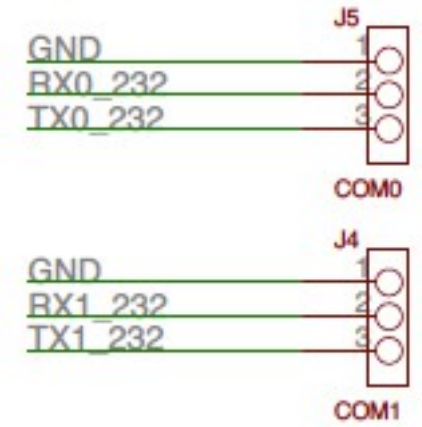
J4	**COM 1**	TYPE	LEVEL	SYMBOL
1	Ground	GND		GND
2	RS-232 COM1 Receive	INPUT	RS232	RX0_232
3	RS-232 COM1 Transmit	OUTPUT	RS232	TX0_232

Table 5: COM 1 Connector

Connector J5 provide RS-232 transceiver signals for UART channel COM0.

J5	**COM 0**	TYPE	LEVEL	SYMBOL
1	Ground	GND		GND
2	RS-232 COM0 Receive	INPUT	RS-232	RX0_232
3	RS-232 COM0 Transmit	OUTPUT	RS-232	TX0_232

Table 6: COM 0 Connector



2.3.5 SERIAL Bus Connector

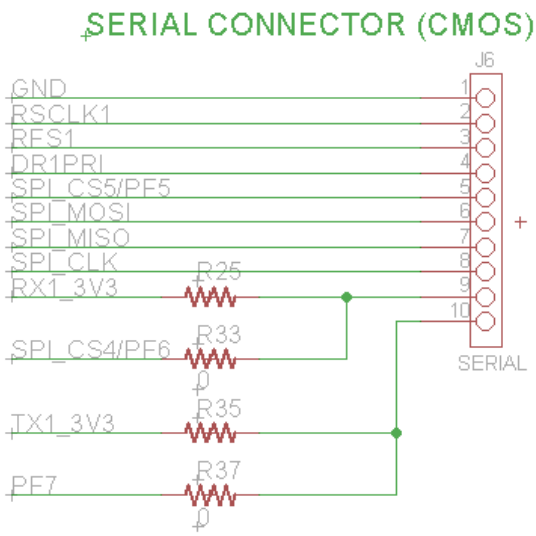
Connector J6 provides CMOS level, Serial Peripheral Port (SPORT) receive port, a Serial Periferal Interface (SPI), and a CMOS UART port. The bus lines are unbuffered and connect directly to the Blackfin processor. The SPORT and SPI bus support a variety of serial data communication protocols to connect standard peripheral devices, such as ADCs or codecs, without external glue logic. A detailed description is found in the ADSP-BF537 Blackfin® Processor Hardware Reference.

J6	**SPORT**	TYPE	LEVEL	SYMBOL
1	Ground	GND		GND
2	RSCLK1	INPUT	CMOS	RSCLK1
3	RFS1	INPUT	CMOS	RFS1
4	DR1PRI	INPUT	CMOS	DR1PRI
5	SPI Chip Select 5	OUTPUT	CMOS	SPI_CS5/PF5
6	SPI MOSI	OUTPUT	CMOS	SPI_MOSI
7	SPI MISO	INPUT	CMOS	SPI_MISO
8	SPI CLOCK	OUPUT	CMOS	SPI_CLK
9	COM 1 Receive / SPI CS4	I/O	CMOS	RX1_3V3
10	COM 1 Transmit / PF7	O/I	CMOS	TX1_3V3

Table 7: SERIAL Connector

In addition, the CMOS level UART signals for COM1 are provided. These lines provide a direct CMOS serial interface (NOT RS-232 level).

Pins 9 and 10 can also be used as SPI bud Chip Selects (CS). The function of Pins 9 is selected by populating resistor R25 or R33. The function of Pins 10 is selected by populating resistor R35 or R35.

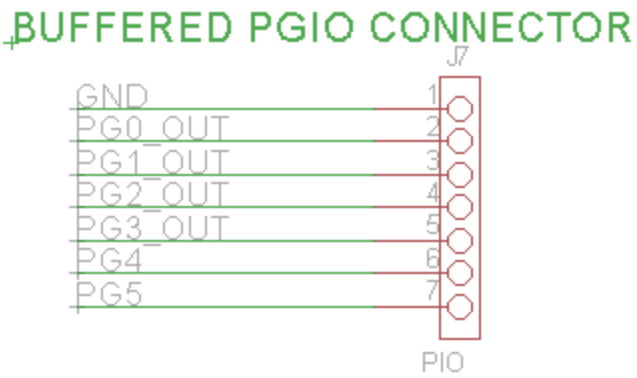


2.3.6 Digital I/O Connector

Connector J7 provides general purpose digital I/O (GPIO) signals. Four GPIO lines . Four channels (PG0 through PG3) are connected to a non-inverting buffer/line driver (74LV241) and are configured as buffered outputs. Two channels (PG4, PG5) are unbuffered inputs/outputs connected directly to the Blackfin processor..

J7	**PGIO**	TYPE	LEVEL	SYMBOL
1	Ground	GND		GND
2	PG0 - Buffered	OUTPUT	0 to 3.6 V	PG0_OUT
3	PG1 - Buffered	OUTPUT	0 to 3.6 V	PG1_OUT
4	PG2 - Buffered	OUTPUT	0 to 3.6 V	PG2_OUT
5	PG3 - Buffered	OUTPUT	0 to 3.6 V	PG3_OUT
6	PG4 - Unbuffered	INPUT/OUTPUT	0 to 3.6 V	PG4
7	PG5 - Unbuffered	INPUT/OUTPUT	0 to 3.6 V	PG5

Table 8: PGIO Connector

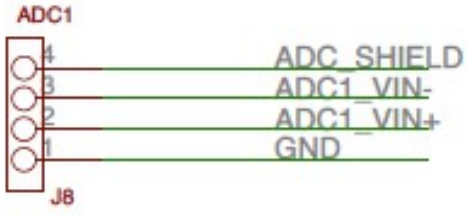


2.3.7 Analog to Digital Converter Signal Input Connector.

Connector J8 provides analog input to the **AD768X SAR** analog to digital converter

J8	**ADC1**	TYPE	LEVEL	SYMBOL
1	Ground	GND		GND
2	-V Input	INPUT	0 to 5 V	ADC1_VIN-
3	+V Input	INPUT	0 to 5 V	ADC1_VIN+
4	ADC Shield	GND		ADC_SHIELD

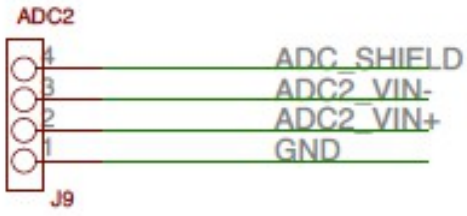
Table 9: ADC 1 Connector



Connector J9 provides analog input to the **AD7766** analog to digital converter. The AD776 is a high performance, 24-bit, oversampled SAR analog-to-digital converter that outputs data samples at 125 kSPS or 62.5kSPS. The differential input voltage is mapped to the 24 bit word as shown in Figure with V_{REF+} = 5V.

J9	ADC2	TYPE	LEVEL	SYMBOL
1	Ground	GND		GND
2	-V Input	INPUT	0 to 5 V	ADC2_VIN-
3	+V Input	INPUT	0 to 5 V	ADC2_VIN+
4	ADC Shield	GND		ADC_SHIELD

Table 10: ADC 2 Connector



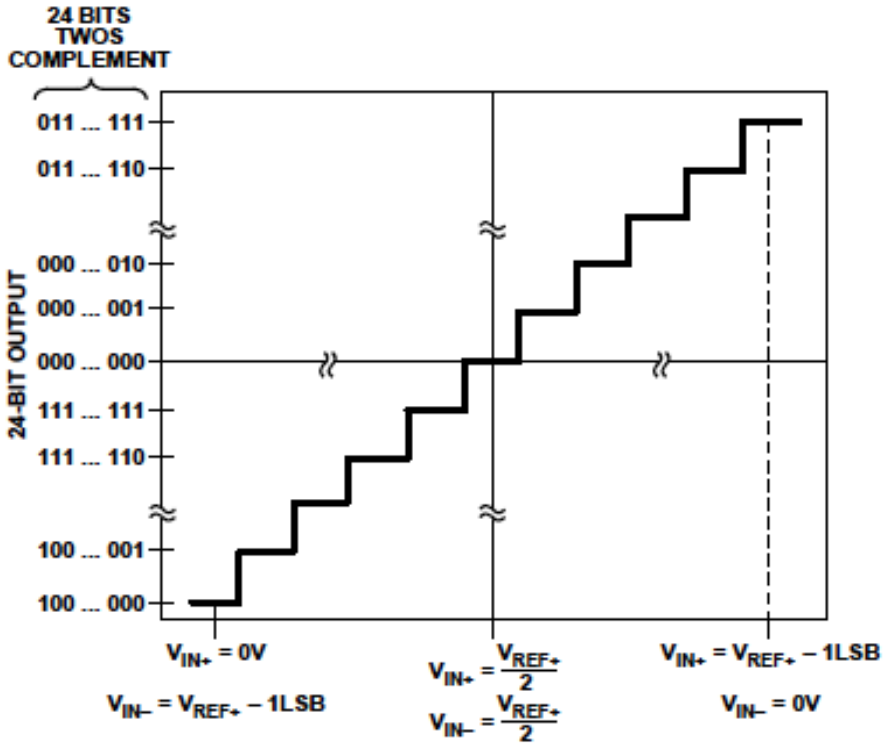


Figure 5: AD7766 Input Voltage Transfer Function

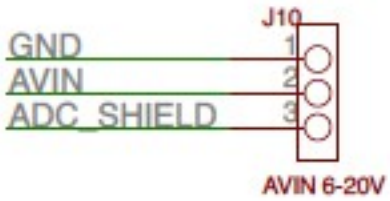
The analog shield is pin provides a separate analog ground plane beneth the ADC converts that is not connecte to the common ground plabe of the board. Closing the contacts on JP1 connects the analog shild plane to the GND plane.

2.3.8 Analog Power Connector

Connector J10 provides the supply voltage for the ADC voltage reference and supply voltage regulators.

J10	**AVIN**	TYPE	LEVEL	SYMBOL
1	Ground	GND		GND
2	Analog Supply Voltage	PWR	6 to 20 V	AVIN
3	ADC Shield	GND		GND

Table 11: ADC Power Connector

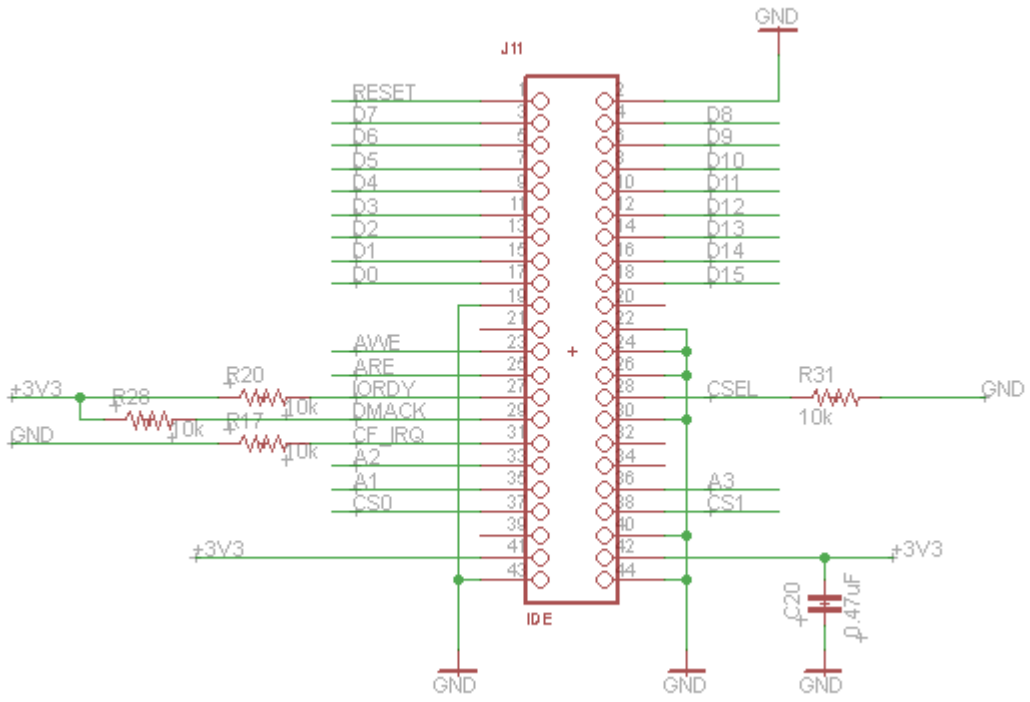


2.3.9 IDE Bus Connector

Connector J11 provides IDE bus (PATA) signals to connect an external mass storage device to the processor. Pin 1 of the IDE connector is indicated with a ***square solder pad***.

Note that the IDE bus voltage and logic levels are 3.3V.

The IDE/ATA connector conforms to the Information Technology - AT Attachment with Packet Interface - 5 (ATA/ATAPI-5) Working Draft, known as T13 1321D, which can be found at http://www.seagate.com/support/disc/manuals/ata/d1153r17.pdf.


GND
J11
RESET
D7
D6
D5
D4
D3
D2
D1
D0
D8
D9
D10
D11
D12
D13
D14
D15
AWE
ARE
IORDY
DMACK
CF_IRQ
A2
A1
CS0
A3
CS1
CSEL
+3V3
R20
R28
R17
10k
R31
C20
0.47uF
IDE

Signal assignments for 50-pin IDE connector

Signal name	Connector contact	Connector contact	Signal name
RESET-	1	2	Ground
DD7	3	4	DD8
DD6	5	6	DD9
DD5	7	8	DD10
DD4	9	10	DD11
DD3	11	12	DD12
DD2	13	14	DD13
DD1	15	16	DD14
DD0	17	18	DD15
Ground	19	20	(keypin)
DMARQ	21	22	Ground
DIOW-:STOP	23	24	Ground
DIOR-:HDMARDY-:HSTROBE	25	26	Ground
IORDY:DDMARDY-:DSTROBE	27	28	CSEL
DMACK-	29	30	Ground
INTRQ	31	32	Obsolete [1)]
DA1	33	34	PDIAG
DA0	35	36	DA2
CS0-	37	38	CS1-
DASP-	39	40	Ground
+5 V (logic)	41	42	+5 V (motor)
Ground(return)	43	44	Reserved - no connection

Table 12: IDE Connector

3. Application Development

The WISPR system will boot on power-up into a fully functional embedded Linux OS (http://blackfin.uclinux.org) with custom device drivers and optimized DSP functions. A full set of example programs are provided to get you started quickly. You can use the example programs as they are for basic data logging or modify them for your own custom applications.

To develop your own applications you must set up a development host. Essentially, the development host is a Linux or Windows PC with a serial port, terminal emulator software, an Ethernet card, and the ***Blackfin toolchain*** software. The tool chain contains the cross-compiler that allows you to compile software on your development host that will run on the WISPR system. There is no compiler on the WISPR system itself.

Instructions for setting up a development host are given below or can be found (along with a lot more) at: http://blackfin.uclinux.org/doku.php?id=setting_up_your_development_host.

The WISPR is loaded with Linux version 2.6.34.7-ADI-2010R1. So install the toolchain that corresponds to this version of the kernel.

The instructions for installing the toolchain on Linux and Windows is at
http://blackfin.uclinux.org/doku.php?id=toolchain
and
http://blackfin.uclinux.org/doku.php?id=toolchain:installing

Note that the instructions for installing the development host are only for cross-compiling application code that will run on the WISPR board in Linux. These instructions are not for compiling the Linux kernel or bare-metal applications.

3.1.1 Windows

The Windows toolchain package installer comes in a single executable (blackfin-toolchain-win32-2010R1.exe). This is found at
http://sourceforge.net/projects/adi-toolchain/files/2010R1/2010R1-RC4 .
Download and install this to setup your Windows development host. The installer takes care of changing your Windows PATH environment variables so they should show up automatically in new applications. You should not need to change your environment yourself.

Also, please keep in mind that this is a command line toolchain that using Makefiles to setup and compile projects. Complete examples of how to build a project are given. But, if you want a GUI, then Eclipse is available with instructions at:
http://docs.blackfin.uclinux.org/doku.php?id=toolchain:eclipse

After you have installed the toolchain, you can cross-compile your application using the **make** command. An example Makefile is given in the example code to get you started.

3.1.2 Linux

Instructions for installing the toolchain in Linux are found at
http://blackfin.uclinux.org/doku.php?id=toolchain:installing.

The toolchain is installed with the following RPMs:

- blackfin-toolchain-2010R1-RC4.i386.rpm
- blackfin-toolchain-uclibc-default-2010R1-RC4.i386.rpm
- blackfin-toolchain-elf-gcc-4.3-2010R1-RC4.i386.rpm

They are installed using the linux command:

- rpm -Uhv blackfin-toolchain-2010R1-RC4.i386.rpm
- rpm -Uhv blackfin-toolchain-elf-gcc-4.3-2010R1-RC4.i386.rpm
- rpm -Uhv blackfin-toolchain-uclibc-default-2010R1-RC4.i386.rpm

The RPM files can be found at
http://sourceforge.net/projects/adi-toolchain/files/2010R1/2010R1-RC4/i386/

This is a command line toolchain that using Makefiles to setup and compile projects. Complete examples of how to build a project are given. Eclipse is also available with instructions at: http://docs.blackfin.uclinux.org/doku.php?id=toolchain:eclipse

3.2 Login Console

To get started using the WISPR you will need a terminal program for serial RS232 communication. The uCLinux OS running on the WISPR uses the COM0 UART for standard input and output. All system messages including boot messages are sent to the system console on COM0. The boot process can be monitored (and interrupted) using a terminal program such as PuTTY, Kermit, or Hyperterm with serial port settings:

```
speed 115200
carrier-watch off
prefixing all
parity none
stop-bits 1
flow-control none
```

More information about terminal programs can be found at
http://docs.blackfin.uclinux.org/doku.php?id=terminal_programs

PuTTY is a convenient communication program for Windows (http://www.putty.org). To open a console, start the **putty.exe** program and

configure the serial connection as shown in Figure 4. You can also login to the WISPR system using ssh or telnet via the Ethernet interface. In this case, use PuTTY to configure an ssh connections as shown in Figure 5.

The WISPR system has a single user with:

```
username:  root
password:  uClinux
```

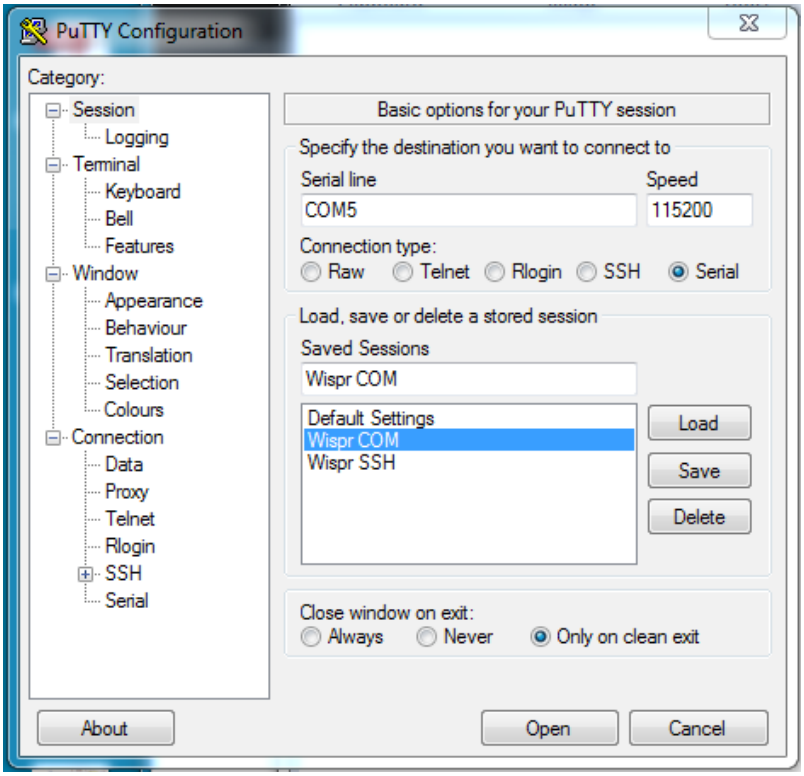


Figure 4: PuTTY Serial Console

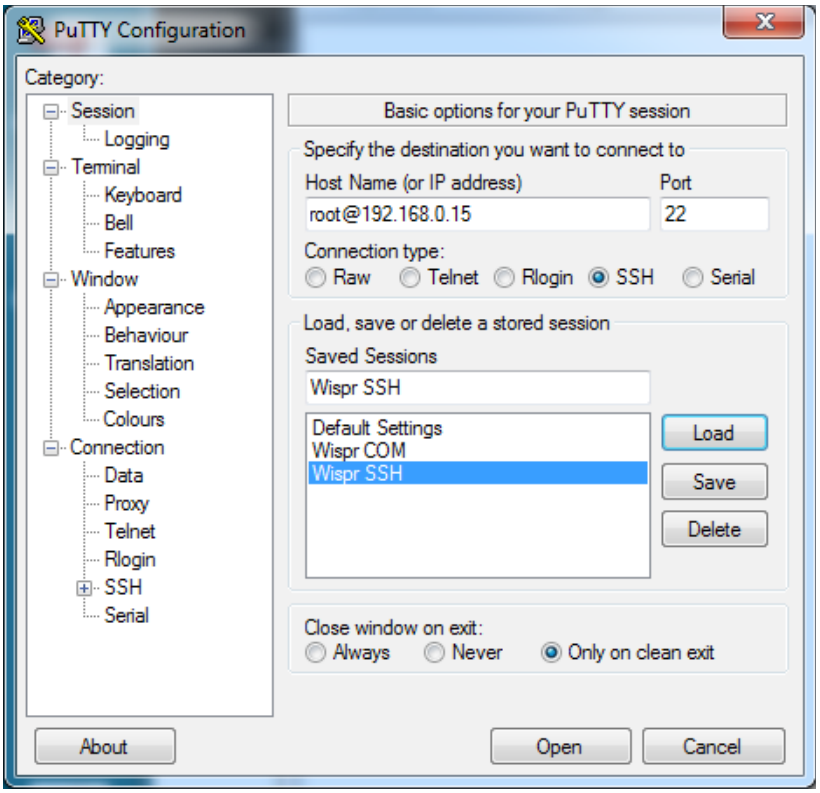


Figure 5: PuTTY ssh connection

3.2.1 Transferring files using scp

Once you have cross-compiled your application, you will need to load it onto the WISPR system. You can always read and write directly to the CF card by removing it from the system. But file transfer is more conveniently done. If you can't physically access the CF card, you can transfer data files to and from the CF card using ***scp***. For example is you want to upload data file from the WISPR to your host use the command:

```
# scp root@192.168.0.20:/mnt/*.dat .
```

If you are using a Windows development host, then you need to install and use an SSH client. There are several Windows SSH clients available for free, but we recommend http://www.putty.org. From the PuTTY page download the latest version of the **pscp.exe** executable. This is a standalone command-line SCP client program. For example, if you want to copy your new application executable that you just cross-compiled to the CF card on the WISPR, you would use the command:

```
# pscp –scp run_main root@192.168.0.20:/mnt
```

The WISPR system user name and password is:

```
username:  root
password:  uClinux
```

3.3 User Software

On power up, the WISPR system will boot the Linux kernel and start the user application and stored on the CF card. All user applications are stored on the CF card. The user application is started by executing a start script, which is also stored on the CF card. At power up the system will execute the following sequence:

1. Start the boot loader (uBoot).
2. Load the uCLinux kernel from flash and start the OS.
3. Mount the CF card.
4. Execute the script named **`start`** located on the CF card.

3.3.1 Start Script

The start script is located on the CF card where all the user applications are invoked. All user application programs and supporting functions must be located on the CF card. The start script and application programs are stored on the CF card (instead of the kernel ROM file system), so they can be modified without having to re-flash the system ROM.

If no start script is found on the CF card, the system will boot normally with a Linux console prompt at the console terminal.

A typical start script is show below. This start script will prompt the user to quit the application program. If nothing is entered, the system will automatically start the **`run_main`** application. If the user enters a "q" character and hits return, the system will return to a Linux terminal prompt.

The start script looks like this:

```
#!/bin/sh
# WISPR control script.

# Set IP address manually
/bin/ifconfig eth0 192.168.0.20

# Startup message
echo " "
echo "--- Embedded Ocean Systems WISPR V1.0 ---"

# Startup prompt, enter 'q' for linux prompt
cp /mnt/prompt_console /bin
chmod 777 /bin/prompt_console
count=10
```

```
quit='q'
echo "Starting recording and detection."
echo "Enter 'q' to stop."
while [ "$count" -gt 0 ]
do
    val=`prompt_console -t 100`
    if [ "$val" = "$quit" ]; then
       exit 1
    fi
    count=`expr "$count" - 1`
done

# Copy user application from the CF card
cp /mnt/run_main /bin/run_main
chmod 777 /bin/run_main

# Start user appliction
/bin/run_main –v1 -l /mnt/wispr.log
```

3.3.2 Run_Main Application

The **run_main** example is a data processing/recording application written in C and cross-compiled for the WISPR system using the Blackfin GNU Toolchain. Access and control of all peripherals are enabled through the Linux kernel device file system. Example programs are provided in the board support package to facilitate user application development.

The run_main example illustrates the basic API functionality for developing user application including:

- AD7766 analog to digital converter operation
- Data logging to CF card
- Serial port communication
- LED and GPIO control
- Watch Dog Timer (WDT) operation

Analog to digital conversion

The AD7766 is a 24 bit oversampling SAR converter. API functions are used to control and read data from the AD7766. The API functions make calls to the kernel device diver interface for the Serial Peripheral Port (SPORT) and General Purpose Timers.

The AD7766 API is defined in `ad7766.h` and `ad7766.c`.

Data Scaling

Data words read from the ADC are 32 bit 2-compliment integers with valid data in the last 24 bits of the word. To maximize data storage the 32 data words are typically scaled to 24 or 16 bits before being saved. Scaling involves bit shifting and discarding certain parts of the data word. If a 2 byte (16 bit) data word is specified, then the 32 bit data word is bit shifted by a specified number of bit to the right (>>). The number of bits to shift the word is specified by the user. If shift = 0, then the 16 least significant bits of the 32 bit ADC words are save. If shift = 8, then the 16 most significant bits of the 32 bit ADC words are save. The value of shift can be between 0 and 8. If a 3 bytes word is specified, then the full 24 bits of the ADC data word are save. If a 4 byte word is specified, then the full 32 bit word is save. In this case the word is shifted 8 bits to the right (<< 8) to put the sign bit in the MSB.

3.4 Ethernet configuration

The WIPR system provides a 10/100 Base-T Ethernet interface for control, data transfer, or remote login. The signals at the Ethernet connector (J2) are driven by a physical transceiver (Micrel KSZ8041NLI), but are NOT transformer isolated. The Ethernet transformer voltage reference is made available at the pin 1 of the Ethernet connector (J2) to allow transformer isolation, if desired.

If you are connecting the WISPR Ethernet directly to the Ethernet port of your host computer, then make sure you are using a crossover cable. Otherwise use an Ethernet hub of switch. If you are using a private network, manually set the IP addresses of your host and the WISPR system. You can use whatever fixed IP address you like on the DSP, just make sure it has the same domain as the host. For example, set the host manually to 192.168.0.1 and set the WISPR manually to 192.168.0.10, as illustrated in the start script.

The address can be set manually from the serial port console use the ifconfig command to set the IP address:

```
> ifconfig eth0 192.168.0.20
```

If you need to assign IP address dynamically, then start the DHCPC deamon on the WISPR system. This can be done at the console or in the start script as:

```
> dhcpcd &
```

Test the network by trying to ping the WISPR from the host:

```
# ping 192.168.0.20
```

If this works, then use ssh to log into the system from the host:

```
# ssh root@192.168.0.20
```

The WISPR system user name and password is:

```
username:  root
password:  uClinux
```

3.5 Storage Devices

3.5.1 CompactFlash Card

A Compact Flash (CF) Storage Card is provided for user configurable memory and mass storage. The CF card operates in True IDE Mode, which is electrically compatible with an IDE disk drive. The CF card's on-card intelligent controller manages interface protocols, data storage and retrieval as well as Error Correcting Code (ECC), defect handling and diagnostics, power management and clock control. Once the CF card has been configured by the system, it appears as a standard ATA (IDE) disk drive.

When purchasing a CF card for the WISPR system, make sure that the card support "true IDE mode."

The system will boot without a CF card installed, although the boot message will display an error messages as the system expect to use a CF card for the use application configuration. With a proper card installed and formatted with known file system, the boot messages will display the card information. For example, the boot message will display the drive information such as:

```
ata1.00: CFA: SanDisk SDCFXP-128G, HDX 6.04, max UDMA7
ata1.00: 250003456 sectors, multi 0: LBA48
ata1.00: configured for PIO
ata1: EH complete
scsi 0:0:0:0: Direct-Access ATA SanDisk SDCFXP-1 HDX PQ: 0 ANSI: 5
sd 0:0:0:0: Attached scsi generic sg0 type 0
sd 0:0:0:0: [sda] 250003456 512-byte logical blocks: (128 GB/119 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't
support DPO or FUA
sda: sda1
sd 0:0:0:0: [sda] Attached SCSI removable disk
```

To access the drive from the Linux file system, the CF card device must be for mounted. The CF card can be formatted with FAT32 or a Linux file system. Mounting the CF card is done automatically at boot time, but can be mount or unmounted manually at the console using the commands:

```
> mount /dev/sda1 /mnt
> umount /mnt
```

The contents of the CF card can then be accessed from the `/mnt` directory. Use the `df` command to see the file system information.

```
> df
```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/sda1	124971040	304096	124666944	0%	/mnt

3.5.2 IDE Drive

The IDE bus interface allows an extra mass storage device to be attached to the system such as a 2.5" SSD or HDD (Figure 6). Note that the bus voltage and signal levels on the IDE bus are 3.3 Volts, as opposed to 5 Volts, therefore the drive you use will need to operate at these levels or be power separately. With a CF card and an IDE SSD attached to the system the boot message will display:

```
ata1.00: CFA: SanDisk SDCFXP-128G, HDX 6.04, max UDMA7
ata1.00: 250003456 sectors, multi 0: LBA48
ata1.01: CFA: TS32GSSD25-M, , max UDMA/66
ata1.01: 61930575 sectors, multi 0: LBA
ata1.00: configured for PIO
ata1.01: configured for PIO
ata1: EH complete
scsi 0:0:0:0: Direct-Access  ATA SanDisk SDCFXP-1 HDX PQ: 0 ANSI: 5
sd 0:0:0:0: Attached scsi generic sg0 type 0
sd 0:0:0:0: [sda] 250003456 512-byte logical blocks: (128 GB/119 GiB)
scsi 0:0:1:0: Direct-Access ATA TS32GSSD25-M n/a PQ: 0 ANSI: 5
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:1:0: Attached scsi generic sg1 type 0
sd 0:0:1:0: [sdb] 61930575 512-byte logical blocks: (31.7 GB/29.5 GiB)
sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't
support DPO or FUA
sd 0:0:1:0: [sdb] Write Protect is off
sd 0:0:1:0: [sdb] Write cache: disabled, read cache: enabled, doesn't
support DPO or FUA
sda: sda1
sdb: sdb1
sd 0:0:0:0: [sda] Attached SCSI removable disk
sd 0:0:1:0: [sdb] Attached SCSI disk
```

The SSD can then be mounted on /ssd by executing the command:

```
> mount /dev/sdb1 /ssd
```

Using the `df` command shows the file system information as follows:

```
> df
```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/sdb1	30950112	16	30950096	0%	/ssd
/dev/sda1	124971040	304096	124666944	0%	/mnt

3.5.3 IDE to SATA Bus Bridge (Experimental)

Using an IDE to SATA bus bridge may be an option to attach certain type of newer large capacity SSD devices. In this case the boot message will show:

```
ata1.01: ATA-9: Samsung SSD 840 EVO 120GB, EXT0AB0Q, max UDMA/133
ata1.01: 234441648 sectors, multi 16: LBA48 NCQ (depth 0/32)
ata1.01: configured for PIO
scsi 0:0:1:0: Direct-Access     ATA      Samsung SSD 840  EXT0 PQ: 0 ANSI: 5
sd 0:0:1:0: [sda] 234441648 512-byte logical blocks: (120 GB/111 GiB)
sd 0:0:1:0: Attached scsi generic sg0 type 0
sd 0:0:1:0: [sda] Write Protect is off
```

```
sd 0:0:1:0: [sda] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
 sda: sda1
sd 0:0:1:0: [sda] Attached SCSI disk

> df
Filesystem          1K-blocks      Used Available Use% Mounted on
/dev/sda1           115377640     61464 109455264   0% /mnt
```

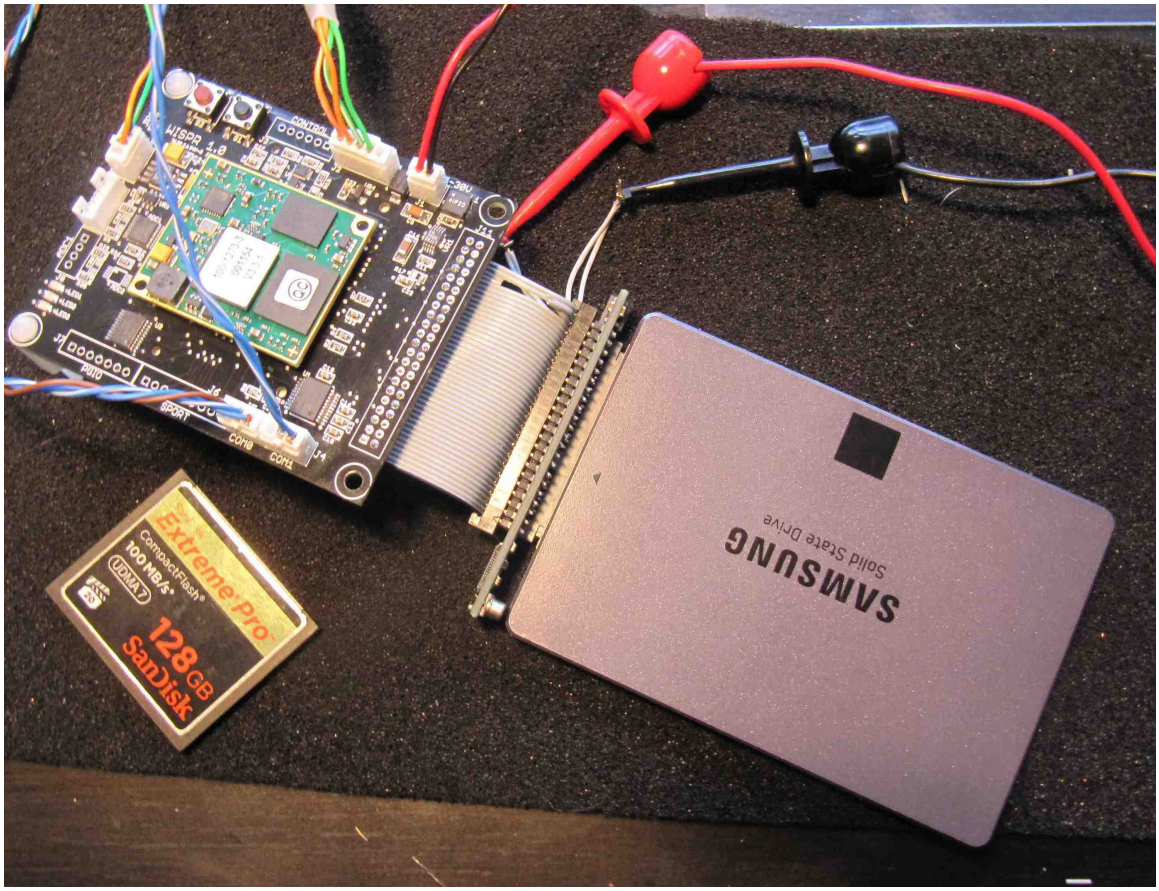


Figure 6: IDE to SATA bus bridge