ODAS5-IR User Guide
Oceanographic Data Acquisition System

Version 4.0 (2018-12-20)

Rockland Scientific International Inc.
520 Dupplin Rd
Victoria, BC, CANADA, V8Z 1C1
http://www.rocklandscientific.com/

Contents

List of Figures

List of Tables

Listings

1 Introduction

This document describes the software and hardware that makes up your ODAS5-IR data acquisition system. This software is used with internally recording instruments that feature the small form factor Persistor CF2 computer. Only a brief overview of the hardware is supplied in this manual. This version of ODAS5-IR (v4.0) produces ODAS v6 data files that must be processed using version 3.1 or later of the ODAS Matlab Library. For more information on data processing, please refer to the ODAS Matlab Library v4.3 manual.

2 Getting Started

The ODAS5-IR software is responsible for controlling the internal components of your Rockland instrument and initiating data acquisition. This section presents a guide to:

- Establish communication with your instrument
- Launch data acquisition
- Download and upload data to and from your instrument

2.1 Communication with your instrument

To use your Rockland instrument, you must first establish communication between a host computer and the instrument. This provides access to the instrument software which enables data acquisition, data transfer and several other capabilities. See Section 5 for more information.

The communication is done through a RS-232 serial connection to communique directory with your instrument's software, and a USB connection to facilitate data transfer to and from your instrument.

Your instrument package will include a *Deck Cable* to establish those two channels of communication through a USB connection and a RS-232 serial connector. The latter is responsible for controlling the instrument and starting data acquisition.

Some instruments do NOT have a RS-232 9 pin connector. Deck cables on these instruments will have a single USB connector. An additional board is included to convert a serial signal to a USB signal. The communication process is otherwise identical.

Communication with your instrument is done through the serial connection and requires a terminal emulator. If you are using MacOS, we recommend the CoolTerm software which is both free and simple to use.

If you are using Windows, Rockland provides an installer which has both the RSILink and MotoCross software, the latter being the recommended terminal emulator. Refer to Section 4 for more information about the installer and the system's requirements. To install, launch the executable.

In addition, you must also identify your instrument COM port. To do so [1]:

1. Open your computer's "Device Manager". In the list of devices, your computer may or may not have a Ports (COM & LPT) section.
2. If this section exists, identify the current ports of your computer (See Figure 1 (a)).
3. If you have a RS-232 9-pin connector, connect it to your computer. Your device manager should display a new item in the Ports (COM & LPT) section. This is your Serial COM port number (See Figure 1 (b)).
4. Otherwise, connect your USB cable to your computer. Your device manager should display two new items in the Ports (COM & LPT) section. One of those is your Serial COM port number the other is associated with your USB connection (See Figure 1 (c)).

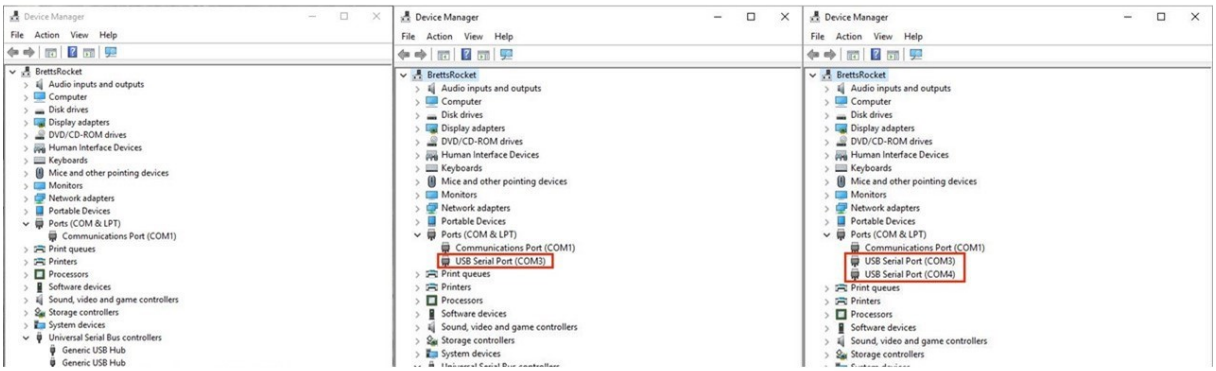

Figure 1: (a) device manager of your computer (you may or may not have a Ports section), (b) device manager after connecting a RS-232 serial connector, (you can easily identify your instrument serial port number), (c) device manager after connecting your USB cable (if you do not have a serial connector). Your serial COM port is one of the two new ports.

Follow the steps below to establish communication with your instrument:

1. Open the MotoCross terminal emulator.
2. Change the port number ('Port #') to correspond to the port identified above (Figure 2). If you identified two (2) ports above, try each one by procceding to point 3. The "sign on" message indicates that you have selected the right port number.
3. Turn your instrument on to establish the connection. Listing 3 shows the typical "sign on" message that should appear in MotoCross if the process was successful. Otherwise, Section 2.1.1 provides trouble shooting tips to help solve communication issues.

[1] These instruction assume that you are using Windows and MotoCross

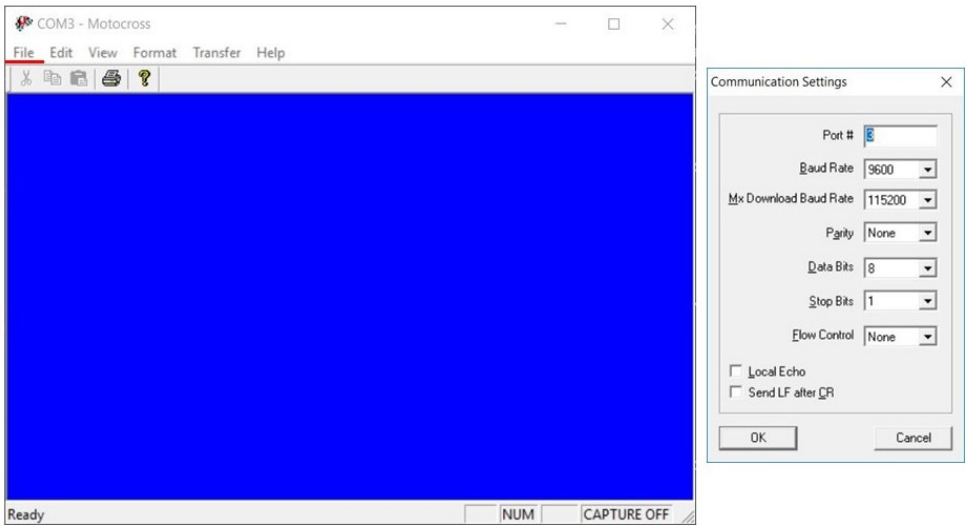

Figure 2: MotoCross' initial display and its Communication Settings window. The window can be accessed through the highlighted file menu item followed by pressing *Communication Settings*.

2.1.1 Troubleshooting Tips

Connecting with your instrument may require some troubleshooting. A few tips to help you connect to your instrument are as follows:

1. When available, connect to a USB 2.0 port [2]. USB 3.0 ports can be more sensitive to the signal because the underwater cables and connectors might not adhere to the newer specifications. If a USB 2.0 port is not available, try connecting a USB 2.0 HUB between the computer and the Rockland instrument.
2. Verify that the COM port number of the RS-232 serial port is between 0 and 9. If the COM port is greater than 9, you must reassign a lower port number to the device. This can be done from the Device Manager.
3. All USB to serial adaptors will work for basic communications but most will fail when using MotoCross's `Load` command. It is recommended to use a FTDI based device - such as MPN: CHIPI-X10 or MPN: UC232R-10. These devices are available from most electronics suppliers such as Digikey. Rockland provides the Tripp-Lite model USA19HS for instruments that have a 9-pin serial connector.
4. Ensure that the Main and CR123 batteries are adequately charged. For more information, consult your Instrument User Manual.
5. If you continue to have communication issues, please contact Rockland Customer Support (support@rocklandscientific.com).

[2] Typically, a USB 2.0 port will be black whereas, a USB 3.0 will be blue

2.2 Acquiring data

After you have connected to your instrument, you are ready to begin data acquisition. This is done with the `odas5ir` command, one of many custom commands within PicoDOS developed for RSI instruments. Listing 1 presents a truncated list of the text provided when using the `help` command. A complete list of commands and their description is provided in Section 5.

Listing 1: PicoDOS command summary after typing `help`

```
C:\>help

~~~~~~~~~~~~~~~~~~~~  ToPico custom commands   ~~~~~~~~~~~~~~~~~~~~
ODAS5IR     Data logging software     ODASAUV     Data logging via AUV
RESTORE     Format CF card for ODAS   PICO        Back to real PicoDOS
SNIPPET     Extract Data segment      SHUTDOWN    Power off instrument
APP         run flash app [args...]   ATTRIB      [+ - RASH] [d:][p][name]
                      ...................
```

> If your help menu does not include the `odas5ir` option you may be in pico mode. Type `app` and carriage return to change mode and try again.

The `odas5ir` command launches the data acquisition application. The objective of this function is to sample the different sensors and generate a data file. A list and description of the various options associated with this function is presented in Section 5.8.

Examples The following examples demonstrate how `odas5ir` can be executed to collect data records and place them into a file.

`odas5ir -n`
: Start data acquisition immediately using settings from the default configuration file *setup.cfg*.

`odas5ir -f ocean.cfg -L 3000 -V`
: Start data acquisition using the configuration file “ocean.cfg”, a system clock speed of 3 MHz, and activate verbose data output.

> The `odas5ir` command requires at least one flag, otherwise it will simply output the help menu.

2.3 Transfer files

The scope of this section is to provide instructions to downloading and uploading files from and to your instrument. There are multiple methods for files transfer (see Section 8). The following focuses on USB data transfer using the RSILink application to download data files and upload a configuration file.

To get started, you must install the RSILink software on your computer (see Section 4). Once completed, follow the instructions below to establish a USB connection with your instrument through RSILink and transfer files:

1. Establish communication with your instrument. (For more details refer to Section 2.1)
2. Type: `usbl` in MotoCross. Listing 2 provides an example of the expected output.

Listing 2: `usbl` command line output

```
C:\>usbl

Program: USBL.c:  Sep 11 2018 12:46:12
Persistor CF2 SN:6774   BIOS:4.6    PicoDOS:4.3

USBL Version 4.0      2018/09/11

USB Link Active.
CIOPuts diverted!
Press 'q' to quit
```

3. Open the RSILink application (see the RSILink User Guide for more information).
4. Select the COM port associated with the USB connection from the dropdown menu [3]. Refresh the list of ports if necessary.
5. Note that the USBL application can take up to 60 seconds to launch. Once launched, press the *Get Root Directory* button which will display the different files in your instrument's root directory.
6. To download files:
   (a) Navigate your CF card directories to select the files you wish to transfer to your computer
   (b) Press the *Download* button
   (c) Select the directory where the files are to be downloaded onto your computer.

[3] This will be a different port from the one used in MotoCross because we are looking for the USB port, not the serial one

You can navigate the directories by double-clicking the desired directory. Double-click *"(..)"* to go up one level and *Get Root Directory* to navigate back to the top directory.

To upload data to the instrument:

1. Select the *Upload* button which promts a dialog window
2. Browse your computer to select the desired file. Note that you may have to change the extension filter if you cannot find the file.

More information regarding the upload and download functionalities can be found in the RSILink User Guide.

3 ODAS Overview

ODAS5-IR is an internally recording data acquisition system for use in applications where the system is deployed in remote environments and when direct communication with the instrument is not feasible. ODAS5-IR consists of software and hardware (Please refer to Fig. 3 for an overview of a typical instrument configuration). The ODAS5-IR application controls all aspects of the data acquisition. ODAS5-IR hardware is comprised of the following components:

- CF2 computer system (small form factor, low-power, 6833x based - made by Persistor).
- ODAS-to-CF2 interface board (called CF2-IF hereafter) that holds the Persistor CF2 computer as a daughter board - P040
- Power Supply board - P050
- One or more digital data producing board(s), such as Analog to Digital Converter (ADC), compass, etc.

Digital data producing boards communicate with the CF2-IF via the digital bi-directional Serial Instrument Bus (SIB).

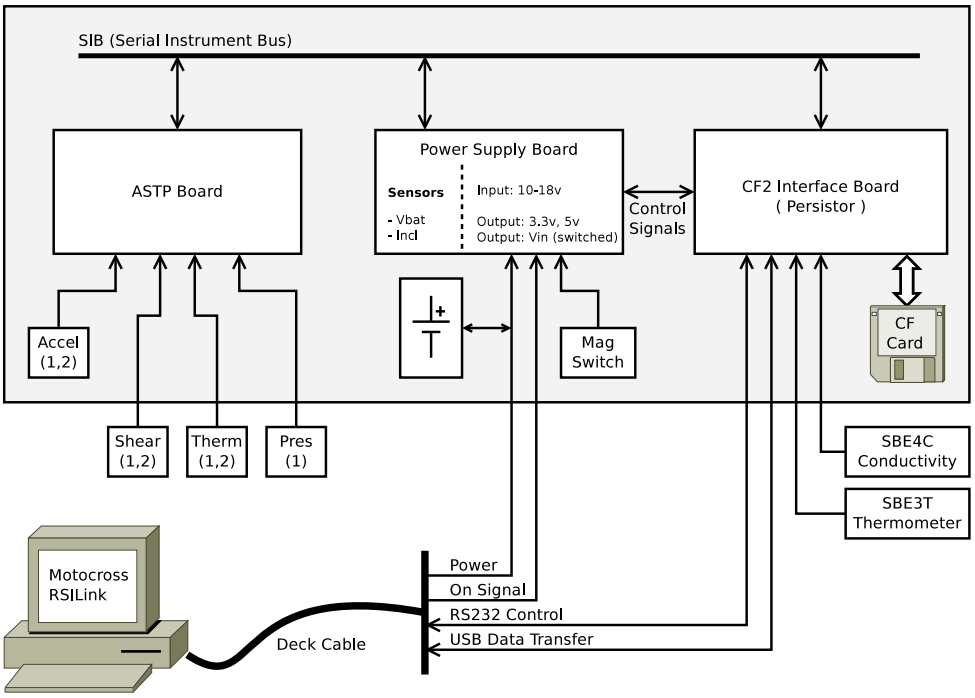

Figure 3: Block diagram of a typical instrument with a Persistor computer for internal data recording.

All sensor systems have a unique address in the range of 0-255. A data acquisition cycle starts when the CF2-IF issues a data request by sending an address to the Serial Instrument Bus. One (and only one) data producing system (such as an ADC) will

match the address and respond with a 16-bit datum word. The CF2-IF buffers the received data and periodically transfers data to the Persistor computer, which in turn transfers the data to the CF card. All data producing systems must have a unique address (so that there is no response conflict) and there must be a response to every address word transmitted by the CF2-IF. ODAS5-IR software controls all aspect of data acquisition, such as:

- sensors (or channels) to be sampled
- order in which to sample the channels
- rate of sampling

The user determines these and other settings with an ASCII configuration file. See Appendix A for details about the configuration file.

3.1 Persistor CF2

The CF2-IF board within the ODAS5-IR system holds a Persistor CF2 computer system manufactured by Persistor Instruments, Inc. At the heart of the CF2 lies a Motorola 68332 Single board computer. The CF2 has a maximum system clock speed of 16 MHz. Its operating voltage is 3.3V and the CF2 features a variety of power saving methods, such as code adjustable system memory and clock speed. The CF2 accepts a range of Compact Flash memory card sizes and it has an onboard 1 MB Flash and 1 MB SRAM.

The Compact Flash card acts as the disk drive and provides storage for programs and data. The CF2 uses a small DOS-like Operating System called PicoDOS (see Section 5.1) and uses a standard RS-232 serial communications line (COM port) for its console I/O. The CF2 system is delivered with the software that your instrument needs for data acquisition (`ODAS5-IR`) and for data download via USB (`USBL`).

4 Software Installation

This section provides additional details about the software used for instrument communication presented in Section 2.1. The windows executable package installs the following software:

MotoCross
: Terminal software designed to interact with the Persistor CF2 micro-computer. Defaults to the correct terminal settings and includes special commands to facilitate firmware updates. However, this software is only available on Windows and users of MacOS and Linux will require an alternative third-party terminal emulator.

RSILink
: A utility that transfers files over USB much faster than via terminal. This application is available on all major platforms.

Some RSI instruments include a USB-to-serial Interface Control (IC) board made by FTDI [4] .Drivers for this IC are required. Users of MacOS and Linux will find that default drivers are included with their operating systems and no driver installations are required. Windows does not include default drivers but Windows Update can perform an automatic installation of these drivers when connected to the Internet. For more information please refer to the RSILink User Guide.

The license to use Motocross is granted only to legal owners of a CF2 computer. For details please view the document *Motocross License.pdf*, which can be found in the RSILink program folder. A help document is available from the main menu after you start Motocross.

[4] Your instrument has this board if your deck cable does not have a RS-232 9-pin serial connector.

4.1 Minimum Requirements

All operating systems will enable you to interact with your instrument. However, for simplicity, the following section assumes you are using Windows. Prior to the installation of the communication tools described above, please ensure that your computer meets the following requirements:

- MS Windows XP or higher
- Internet connection for installation on Windows 7.
- One available USB 2.0 port
- One available RS-232 9-pin compatible serial communications port [5] or serial to USB adaptor and a USB port. (Only one USB port is needed if you do not have a Serial Connector)

5 ODAS5-IR

The instrument's Persistor runs an operating system called PicoDOS from which the ODAS5-IR.app application runs. In version 4.0, the software is installed in the FLASH memory embedded within the Persistor computer, as opposed to on the CF card, as in previous versions of ODAS5-IR. Consequently, there is no longer a `ODAS5IR.PXE` file.

5.1 PicoDOS

PicoDOS borrows many features from other popular DOS variants, such as MS-DOS. The default communication parameters are 9600 Baud, 8 data bits, 1 stop bit, no Parity. The default baud rate and other communication parameters can be changed using the PicoDOS `baud` command. Your instrument will be configured with a deck cable that allows you to connect the COM port and USB port on your PC to the Persistor CF2 computer. Listing 3 shows the typical "sign on" message that you can observe in your terminal window when you power your instrument on.

Listing 3: CF2 sign-on message

```
------------------------------------------------------------------
    ODAS Data Acquisition 5IR: Apr 25 2018, 16:22:30
    Persistor CF2 SN:6774   BIOS:4.03   PicoDOS:4.03
------------------------------------------------------------------

C:\>
```

[5] Most USB Serial converters are not fully compatible with MotoCross. RSI recommends the Keyspan USA-19HS or a device based on a FTDI chipset.

The first command passed to the PicoDOS operating system builds the FAT (File Allocation Table). **This can take up to 45 s for a large (∼64 GB) card with many files**. This delay must be considered for long-term and intermittent deployments. After the initial building of the FAT, the operating system responds rapidly to commands.

The different PicoDos commands can be seen in Listing 4. For more information on a specific command, type the command followed by /?. The following subsections provide details on the most commonly used commands.

Listing 4: PicoDOS command summary after typing `help`

```
C:\>help

~~~~~~~~~~~~~~~~~~~~  ToPico custom commands   ~~~~~~~~~~~~~~~~~~~~
ODAS5IR     Data logging software     ODASAUV     Data logging via AUV
RESTORE     Format CF card for ODAS   PICO        Back to real PicoDOS
SNIPPET     Extract Data segment      SHUTDOWN    Power off instrument
APP         run flash app [args...]   ATTRIB      [+ - RASH] [d:][p][name]
BACKROM     [d:][path] [/SAVI]        BAUD        [newrate] [/Q/P/E/O/N/2]
BOOT        [PICO][PBM][APP]          CAPTURE     [d:][p]fn [/Dx/B/N/E]
CCC         Card Change [delay secs]  CHDIR       [drive:][path]
CHKDSK      [d:][p][fn] [/F][/I]      COPY        source dest [/V]
DUMP        file[start[,end]]         DATE        [mdy[hms[a|p]]] /IEUMCP]
DEL         disabled -- see help      DIR         [d:][p][fn] [/PWBLV4A:a]
ERASE       disabled -- see help      FDISK       [/Pnn/M/Sdev/@/F/Rnn/Q]
FORMAT      [drv:][/V[:label]][/Q/E]  GO          args... | addr /A | /Fn
LO          [ofs][;Bx[+]] [;G]        MOUNT       [V:][DEV[-n][/D/P/N/V/Q]
MKDIR       [drive:][path]            MD          display [range]
MM          modify [address]          ML          disassemble [range]
MON         Reset to PBM              PATH        [[d:]path[;...]] [/P]
PBM         Reset to PBM              PROMPT      [text] [/P]
PR          pin read    <1..50>       PC          pin clear   <1..50>
PS          pin set     <1..50>       PT          pin toggle  <1..50>
PM          pin mirror  <1..50>       TIME        [hh:mm:ss [a|p]] [/M/C]
TYPE        [drv:][pth][name]         REN         [d:][p]oldname newname
RESET       (hard reset)              RMDIR       [drive:][path]
SAVE        file[start][end]          SD          sect.dump[d:][range][/C]
SET         [var=[str]] [/SLFE?]      XS          [/X][/C][/Q]file
XR          [/X][/C][/Q][file]        YS          [/G][/Q]file[,file...]
YR          [/G][/Q]                  VER         Firmware versions
```

Several of the commands above are not useful for data acquisition purposes and could possibly damage your Persistor. We recommend using exclusively the commands described below

5.2 Restore

Formatting the CF card can be safely performed using the `restore` system command. An example `restore` session is shown in Listing 5. This command saves files within the root directory, formats the CF card, and then recovers the saved files. Lastly, a DATA directory is created and a copy of USBL is placed onto the CF card.

The `restore` command will delete all files in the DATA directory of your CF card. You are advised to download all data files [a] to your computer before using this command. For instructions to download data files, refer to Section 8

[a] You may also want to download the autoexec.bat and configuration file

Listing 5: Output of the `restore` command.

```
C:\> restore

WARNING:  This command will delete all data files.

Formatting the CF card will delete all data.
Files located in the root folder will be saved.

    Continue?  [Y] ? Y

Formatting the card...done!

Writing files back to the card...

C:\>dir

 Volume in drive C has no label
 Volume Serial Number is 3E87-5666
 Directory of C:\

DATA            <DIR>          03-08-00  12:42p
USBL.RUN             36,512    03-08-00  12:42p
SETUP.CFG             3,659    03-08-00  12:42p
AUTOEXEC.BAT             15    03-08-00  12:42p

         3 file(s)          40,186 bytes
         1 dir(s)    2,037,448,704 bytes free
C:\>
```

Using the `restore` command is a quick way to recover from a corrupted FAT without having to upload configuration files or executables. Root level files are recovered, USBL is copied from Flash memory to the CF card, and a DATA directory is created. Each of these tasks can be performed manually but they are automated by the `restore` command. USBL is available immediately after a `restore` so that other files can be transferred using RSILink.

Although the content of the CF card root directory is saved, if you log file is large you may want to save it as well. Transferring the log file back to the CF card after using the `restore` command will preserve the continuity of the naming scheme of your data files.

5.3 Snippet

The `snippet` command extracts and saves a segment of a data file. This is used to inspect a short sample (a snippet) of a data file to check the quality of the acquired data. The command's syntax is:
`snippet inputs [/S n] [/B m] [/O output] [/Q] [/V] [/E] [/?].`
The different arguments are as follows:

- Input: input file [6]
- S: start at record "n" [7] (defaults to the mid point of a file)
- B: the snippet segment will have "m" records (defaults to 30)
- Output: output file [6](will default to snip.p)
- Q: Query record count
- V: verbose mode
- E: Print error codes
- ?: Display this message

When using a Kongsberg SeaGlider, snippet files are automatically generated from the middle of each data file

5.4 USBL

The `usbl` command is used in conjunction with RSILink to transfer files to and from your instrument via a USB connection. Information regarding both the `usbl` command and the RSILink software can be found in the RSILink User Guide (also accessible through the RSILink *Help* button.) Listing 6 shows the expected output when using the `usbl` command.

[6] The path must be specified if the file is not in the current directory
[7] If n is negative, the start will be calculated from the end of the file.

When exiting USBL, the Persistor is reset. This will boot your Persitor into the ODAS5-IR software and will launch the *autoexec.bat* file if present. Therefore, remember to quit data acquisition (pressing any key during the count down) to avoid creating unnecessary data files.

Listing 6: USBL command line output

```
C:\> usbl

USBL Version 4.0,       2018/09/11
Persistor CF2 SN:6774       BIOS:4.6
PicoDOS: 4.3
USB Link Active.
CIOPuts diverted!
Press 'q' to quit
```

5.5 Del & Erase

Both the `del` and `erase` commands have been removed from the data acquisition software. To recover storage space, the entire drive must be formated (using the `restore` command). This is done to prevent the CF card from becoming fragmented which happens when files are deleted and this eventually leads to data loss and FAT corruption.

Note that the `del` command is still available outside the ODAS5-IR software. This is accessible by typing: `pico`, then deleting files and returning to ODAS5-IR by typing: `app` once the files have been deleted. However, we strongly advise against using this `del` command for reasons mentioned above.

5.6 Capture

The `capture` command is a PicoDOS feature to quickly create ASCII text files on the CF card. This command is quite useful for creating simple files such as the *autoexec.bat* file. Larger files, such as configuration files, should be uploaded through RSILink (see Section 8.1).

See PicoDOS help for information on the `capture` command. Type: `help capture` at the PicoDOS command prompt for help information.

The downside to using the `capture` command is that not all terminals are compatible with `capture` in their default configuration. **When using `capture` to write the autoexec.bat file, you must press enter at the end of your code.**

5.7 Others

Other common commands called by users are described below:

`cd`	change directory
`dir`	list directory contents
`type`	show contents of file

5.8 Odas5ir

The `odas5ir` command launches the data acquisition application which has two modes of operation. The first is the data acquisition mode, in which the instrument samples the sensors and generates a data file. The second mode is calibration mode, which displays the statistics of a range of channels (more information on calibration mode can be found in Section 5.9). The different command line options are summarized in the Listing 7. **Note that option flags are case insensitive**.

Listing 7: Command Line Option Summary

```
C:\> odas5ir
typical usage:
ODAS5IR -F <setup file> [-C <all|0..255>]
    -A : fAst serial instrument bus
    -B : maintain Backup of current data file
    -C <all|0..255[:0..255]> : specify channel(s) to Calibrate
    -D : Display verbose channel output
    -E : enable Echo mode [USE WITH CAUTION]
    -F <config File> : configuration file defaults to 'SETUP.CFG'
    -H : show this Help message
    -L : system cLock speed in kHz (default 3000)
    -N : skip 10s countdown before starting DAQ
    -R : specify sample Rate (only in calibrate mode -C)
    -S : specify number of Samples(only in calibrate mode -C)
    -T : maximum duration of data file (default: no limit) [records]
    -V : display current battery Voltage
    -X : acquire but do not save data
    -Z : maximum siZe of data file (default: 1024MB) [KB]
```

Explanation of command line option flags:

`-A : use fast serial instrument bus`

The default serial instrument bus speed is acceptable for most users. In situations where more speed is required, the instrument can be configured to double the rate of internal communications. All boards must be configured to use a fast serial instrument bus so it is not recommended that users change this setting on their own. Users should use this option only when the instrument is configured accordingly.

`-C <all|0..255> :  specify channel (or all) to calibrate`

Enable calibration mode on specified channel - or “all” for all channels. Calibrate mode samples the specified channels and displays statistics calculated from the resulting data. Channel statistics are shown in the following format:

```
ch: x   min:  y   max:  t  mean:  u   stdev:  v
```

where *x* corresponds to the channel number, *y* shows the minimum value, *t* shows the maximum value, *u* shows the calculated mean, and *v* displays the standard deviation. All statistical values are given in counts.

`-D : display verbose channel output`
Print statistics from observed data to the console. Used to quickly verify channels are working as expected. Can be used by autonomous vehicles to verify instrument integrity. One line is printed for each record. See section 6 for a description of the output. Note that if this option results in bad buffers, increase the system clock (option `-L`).

`-F <config file> :  configuration file for data acquisition`
The configuration file used for data acquisition and calibration. When not specified, the default file "setup.cfg" is used.

`-H : show this help message`

`-L : system clock speed in kHz (default 3000)`
The rate of the Persistor system clock. The lowest suitable value is 3000 (*i.e.* 3 MHz) which minimizes power consumption and is used as the default value. The 3 MHz) is recommended unless the instrument is sampling using a fast serial instrument bus. The maximum speed available is 16 MHz.

`-N : skip 10s countdown before starting DAQ`
Start data acquisition immediately by skipping the default 10 s delay. The delay allows the user to abort data acquisition if required.

`-R : specify sample rate (only in calibrate mode)`
Sampling rate used when calibrating a channel in samples per second. Default value is 512.

`-S : specify number of samples (only in calibrate mode)`
Number of samples used when calibrating a channel. Default value is 512.

`-T : maximum duration of data file (default:  no limit) [records]`
Limit duration of data files to T records. There is no default duration limit. The duration is specified in records. In most cases, one record is one second.

`-V : display battery voltage`
Read, convert, and display the battery voltage. The `V_Bat` channel of type `voltage` is used when converting into physical units. The program exits after displaying the battery voltage.

`-X : disable saving of data file`
Sample data but do not save to the CF card. This option is used when you have an alternative method of saving the data.

`-Z : maximum size of data file (default:  1024MB) [KB]`

Limit the size of resulting data files. Defaults to the maximum file size within a FAT32 partition, or 1 GB. Data files should typically be kept under 100 MB to facilitate data processing.

Examples of data acquisition commands are presented in section 2.2

The `odas5ir` command requires at least one flag to be set, otherwise it will simply output the help menu.

5.9 Calibration Mode

Calibration mode involves sampling the sensors and generating statistics on the resulting data. This mode is typically used with test probes to verify that the electronic components are working correctly. Despite the name "*calibration*", this mode does not perform a calibration of the instrument.

The `-C <all|0..255>` parameter is used to place the instrument into calibration mode. See Listing 7 for a list of options in calibration mode.

Listing 8 demonstrates launching `odas5ir` in calibration mode and shows the resulting output.

Listing 8: Example Calibration Output

```
C:\>  odas5ir -f setup.cfg -c all
Persistor CF2 SN:6774   BIOS:4.03   PicoDOS:4.03

ODAS5IR: 3.5: 2014-05-12 17:26:26 Pacific Daylight Time build 85 (nomod)
free sectors: 2035648

Checking powerok
fsz = 2941
num rows: 2
found section: 'P' on channel: 10
found section: 'v_bat' on channel: 32
stop recording after release: disabled
max time trigger release: disabled
max pressure not specified, defaulting to 500 dBar
pressure release: enabled - max pressure: 500
pressure ch detected at index 10


Calibrating - Type 'Q' to exit.

created input buffer of size: 1024 bytes
ch:   0  min:     +4  max:     +6  mean:     +5.0  stdev:     0.20
ch:   1  min:     +2  max:     +8  mean:     +4.7  stdev:     1.28
ch:   2  min:     +4  max:    +11  mean:     +7.2  stdev:     1.43
ch:   4  min:    -46  max:    -45  mean:    -45.3  stdev:     0.47
ch:   6  min:    -47  max:    -45  mean:    -45.9  stdev:     0.34
ch:   8  min:     -7  max:    +16  mean:     +4.8  stdev:     6.71
ch:   9  min:    -10  max:    +20  mean:     +6.0  stdev:     8.95
ch:  10  min:    -79  max:    -77  mean:    -78.2  stdev:     0.45
ch:  11  min:    -82  max:    -75  mean:    -79.6  stdev:     1.71
ch:  32  min: +26638  max: +26658  mean: +26650.1  stdev:     5.13
ch: 255  min: +32752  max: +32752  mean: +32752.0  stdev:     0.00
```

During calibration, all channels are interpreted as raw 16-bit signed integer values and displayed accordingly. Some instruments have sensors that output unsigned values. This can generate unexpected results because values above 2^{15} will be reported as being negative. This does not represent an error with the instrument.

Sensors that generate 32-bit values can also display seemingly erroneous data when used in calibrate mode. This is expected and should be ignored. RSI staff can verify that the displayed values are acceptable upon user request.

Each instrument has channels 0 and 255, but not all instruments have channels iden-

tical to those from the example presented in Listing 8 which provides examples of outputs. Channel 0 is the result of sampling the instrument's analog ground plane. Values close to 0 are expected with standard deviation values less than 1. Channel 255 is the special character channel and used for aligning the data stream. A value of 32752 and standard deviation equal to 0 is expected.

On a final note, the calibration results are not logged and must be recorded when observed or captured from the RS-232 console.

Calibration Examples The following examples demonstrate the calibration mode of ODAS5-IR.

`odas5ir -f ocean.cfg -c 5`
: Start calibrate mode on channel 5 using settings from the configuration file "ocean.cfg". Uses the default value of 512 for number of samples and sample rate.

`odas5ir -f survey.cfg -c all -r 256 -s 1024`
: Start calibrate mode on all channels listed in the configuration file "survey.cfg" using a sampling rate of 256 samples/sec and collecting 1024 samples for each channel. The sampling of each channel will take a minimum of 4 s.

6 Console Output

Status messages are output to the console when running ODAS5-IR. One message is printed for each record that is received. You can verify that an instrument is recording data successfully by checking that the messages contain reasonable values.

Messages are formatted in one of two ways. By default, simplified messages are generated. These messages contain sufficient information to monitor the increasing record number and provide the pressure in physical units. ODAS5-IR can also generate verbose output when it is called with the `-D` parameter.

Verbose output is difficult to read by eye but can be used by autonomous controllers to verify the health of an instrument.

Note that the error messages are interleaved with the status messages. This can cause problems for scripts that are parsing the output and expect a fixed format. Scripts that parse the output should be able to ignore input that does not match the expected format.

ODAS5-IR startup sequence
Calling commands in the ODAS5-IR terminal will generate a startup sequence as a series of statements regarding the status of the instrument, as displayed in Listing 9.

Lines 6 through 10 are meant for instruments with a release system, NOT tethered instruments. For user with a release systems, please see your instruments manual for instructions to select those parameters accordingly.

Listing 9: ODAS5-IR startup sequence

```
Checking powerok
fsz = 2941
num rows: 2
found section: 'P' on channel: 10
found section: 'v_bat' on channel: 32
stop recording after release: disabled
max time trigger release: disabled
max pressure not specified, defaulting to 500 dBar
pressure release: enabled - max pressure: 500
pressure ch detected at index 10
```

6.1 Simplified Console Output

The default output of ODAS5-IR is presented in listing 10.

Listing 10: An example of the Default Simplified Console Output

```
starting DAQ
free sectors: 2045280
created input buffer of size: 10240 bytes
L    0 N    1 p -1.05
L    1 N    2 p -1.05
L    2 N    3 p -1.05
L    3 N    4 p -1.05
L    4 N    5 p -1.05
L    5 N    6 p -1.05
L    6 N    7 p -1.05
L    7 N    8 p -1.05
L    8 N    9 p -1.05
L    9 N   10 p -1.05
L   10 N   11 p -1.05
```

The first three lines are start-up status messages. The subsequent lines, those that start with `L`, represent data records. Each line is formatted using,

```
L    <LAST_RECORD_COUNT> N    <NEXT_RECORD_COUNT> p <PRESSURE>
```

where `LAST_RECORD_COUNT` and `NEXT_RECORD_COUNT` are record numbers for the last and next records. The current pressure is displayed following the `p`. This pressure value is in units of dbar and is calculated by using the coefficients in the `pressure` section of the configuration file. Information about the configuration file's section and parameters can be found in Appendix A. If these coefficients are erroneous, the pressure value will be wrong. Note that in the example above, pressure is reported to be negative due to inaccurate calibration coefficients.

The pressure coefficient can be adjusted immediately or later during data processing, because only raw data are recorded.

6.2 Verbose Console Output

The verbose output option was added to support instruments on autonomous vehicles. These vehicles often need to verify the instrument's health during their deployment. It is possible for probes to foul or break during deployment thereby requiring the mission to be aborted prematurely.

Although designed to be read by machines, the verbose output is human readable and can provide useful information to an operator. Listing 11 provides an example of verbose output.

Listing 11: Example Verbose Console Output

```
L   0 N   1 p -1.05 f3 c0,5,0 c1,4,1 c2,7,1 c4,-27,0 c6,-22,0 c8,4,2
c9,6,3 c10,-86,0 c11,-86,1 c32,23843,4 c255,32752,0
L   1 N   2 p -1.05 f3 c0,5,0 c1,4,1 c2,7,1 c4,-27,0 c6,-22,0 c8,4,2
c9,6,3 c10,-86,0 c11,-86,0 c32,23837,10 c255,32752,0
L   2 N   3 p -1.05 f3 c0,5,0 c1,4,1 c2,7,1 c4,-27,0 c6,-22,0 c8,4,2
c9,6,3 c10,-86,0 c11,-86,2 c32,23837,10 c255,32752,0
L   3 N   4 p -1.05 f3 c0,5,0 c1,4,1 c2,7,1 c4,-27,0 c6,-22,0 c8,4,2
c9,6,3 c10,-86,0 c11,-87,1 c32,23838,10 c255,32752,0
L   4 N   5 p -1.05 f3 c0,5,0 c1,4,1 c2,7,1 c4,-27,0 c6,-22,0 c8,4,2
c9,6,3 c10,-86,0 c11,-86,0 c32,23838,10 c255,32752,0
L   5 N   6 p -1.05 f3 c0,5,0 c1,4,1 c2,7,1 c4,-27,0 c6,-22,0 c8,4,2
c9,6,3 c10,-86,0 c11,-86,0 c32,23837,11 c255,32752,0
```

Lines displayed in listing 11 conforms to the following structure;

```
L <LAST> N <NEXT> p <PRES> f <EXT> ( c<CH>,<AVG>,<STD> )*
```

LAST: record number of the most recently observed record

NEXT: record number of the current / next record

PRES: pressure converted into physical units of [dbar]

EXT: file number / extension of current data file

CH: channel id, integer value between 0 and 255

AVG: average channel value as a 16-bit signed integer

STD: average absolute difference between observed value and calculated average. This is similar to standard deviation but its computation is simpler.

The average and deviation values are calculated by assuming that the data words consist of 16-bit signed numbers. If the channels are unsigned or part of a 32-bit word, then the calculated values may seem erroneous. Most data are 16-bit signed numbers.

Test sensors were installed in the instrument during the generation of listing 11. This results in clean thermistor (`c4` and `c6`), shear (`c8` and `c9`), and pressure (`c10` and `c11`) data. The battery voltage channel, (`c32`) shows more deviation which is expected from a power source. Channel 255 is used for data synchronization is referred to as the special character. The value of this channel is always 32752[8]. Each line wraps due to the limited width of this document. The actual channel assignments are provided with your instrument, but the ones given above are typical.

[8] which equals 0x7FF0 in hexadecimal format

7 Persistor CF2 Date and Time

The Persistor CF2 has an internal time of day (TOD) clock. This clock is used to set the file time when files are created on the CF card. The time is also used when timestamps are added to log files.

The TOD clock is powered by the CR123 battery on the power supply board. Should this battery be depleted, or the board disconnected, the TOD clock will have to be reset. A warning message is provided should the date be before the year 2000 (see listing 12).

Listing 12: PicoDOS will remind you when date/time is uninitialized

```
------------------------------------------------------------------
    ODAS Data Acquisition 5IR: Apr 25 2018, 16:22:30

    Persistor CF2 SN:6774   BIOS:4.03   PicoDOS:4.03
------------------------------------------------------------------

The clock needs setting if it's not February 17, 1972!
```

7.1 Timezones, Daylight Savings Time, UTC

The concept of time is simple enough but things become complex when one starts to consider timezones, daylight savings time, and instruments that can cross timezone boundaries while recording data.

To make things as simple as possible, RSI adheres to the policy; **all instruments shall have clocks set to UTC time.** Rockland's processing software assumes the files are created in UTC.

7.2 Setting Time using RSILink

RSILink includes a *Set Time* button that, when pressed, sets the Persistor CF2 time to the UTC value of the current computer time. **We strongly recommend using this button to set time as opposed to other methods.**

In addition, when you download files with RSILink, the file modification time is adjusted to match the UTC creation time by the Persistor. The operating system on your personal computer is fully aware of UTC. Thus, when you display a file, it is correctly tagged with the time for the time-zone of your computer.

For example, if you use RSILink to set the Persistor time, record a data file at 13:45 (local time), the file will have a time stamp of 1:45 PM when it is downloaded to your PC.

7.2.1 Setting Time from the Terminal

The `date` command is used to set the date and time from a terminal. Type `help date` for more information on this command. Example usage is provided in listing 13.

Listing 13: PicoDOS date command help and usage.

```
date [mm-dd-yy[yy] [hh:mm:ss [a|p]]] [/I] [/E] [/U] [/M] [/C] [/P]

Set the real time clock from the date (and optionally time) based
on parameters in the form: [mm/dd/yy[yy] [hh:mm:ss [a|p]].

Type DATE without parameters to display the current date settings

  /U          Expect USA date format (default):  mm/dd/yy
  /I          Expect ISO date format:            yy/mm/dd
  /E          Expect European date format:       dd/mm/yy
  /M          Military 24 hour time format
  /C          Civilian 12 hour time format
  /P          Prompt for date and time

Example:
c:\>  date 03-08-2011 12:07:00 /m
Clock reads: Tuesday, March 08, 2011  12:07:00
```

The time of day clock is accurate to ∼100ppm and, like most computer clocks, should be updated periodically.

8 Tranferring Files

Files can be transferred to and from your instrument using five different methods.

1. Using RSILink to download and upload files over a USB link.
2. Using the MotoCross `Load` command to upload binaries or ASCII files over a serial port connection.
3. Using the `capture` command to upload text files.
4. Using (X/Y)MODEM, which is supported by the Persistor. They can download and upload both binary and ASCII files, if this protocol is supported by your terminal.
5. Using an external CF card reader. But, this requires the removal of the card from your instrument.

> ⚠ Using a personal computer to modify the CF card could corrupt the card's FAT and thus, we advise against using this method unless necessary.

8.1 Using RSILink

RSILink provides a platform to download and upload data files from and to the CF card through the USB connection between the host computer and your instrument. Extensive information can be found in the RSILink User Manual.

8.2 Loading Files with MotoCross

MotoCross contains a proprietary `Load` command that can be used to upload executables and ASCII text files over a RS-232 serial port. For instruction to use this command, please refer to section 9.

8.3 PicoDOS Capture command

PicoDOS provides a command to quickly capture text into a file. Additional information regarding the capture command is presented in section 5.6.

8.4 (X/Y)MODEM Support

The XMODEM and YMODEM protocols are designed for transferring binary files over serial communication channels, such as RS-232. MotoCross lacks support for both XMODEM and YMODEM but there are good alternative terminals that natively support both protocols – for example, TeraTerm for Windows.

The PicoDOS command XS, XR, YS, and YR are used for X/YMODEM transfers. Use the PicoDOS HELP command to get more information on these commands. For help with X/YMODEM transfers, see the help section of the terminal software you are using.

If possible, use YMODEM instead of XMODEM. The XMODEM command transfers blocks of fixed size. If the size of a file is not a whole-number multiple of this block size, then XMODEM pads characters to the end of the file to complete the transfer. Padding is not a problem for ASCII files because the end of string character will still be in the correct place. However, binary files will be damaged. Using YMODEM avoids this problem.

X/YMODEM transfers can be slow and difficult to initiate. However, such transfers are also robust and flexible.

You may change the BAUD rate to accelerate the data transfer. A rate of 115 200 is 10 times faster then the default rate of 9600. See the `baud` command in PicoDOS for more information.

8.5 USB External Card Reader

The CF card can be physically removed from your instrument and its files can be read using a CF card reader. You should avoid writing files with a card reader because this may change the configuration parameters of the FAT. Such a change may severely reduce the speed of writing data by the Persistor.

The obvious downside to using an external card reader is that it requires opening the pressure case. Care must be exercised because this could cause leaks. This method should therefore be avoided.

The advantage of using an external card reader is that this method provides by far the greatest transfer speed. If you have to transfer several gigabytes, then using a reader saves you hours.

8.5.1 USB connection

Data can be transferred from the instrument to a host computer using the RSILink application. This application allows for data to be transferred using a USB2.0 connection at a rate of approximately 250 KB/s. RSILink can perform data transfers in both directions, but its primary purpose is to transfer data files from the CF2 to the Host PC. In order to perform High speed USB transfers, two conditions must be met:

1. Start the RSILink utility on the Host PC. Please refer to the RSILink User Guide for more details.
2. Put the CF2 Persistor into USB data transfer mode. This is accomplished by starting the utility called `usbl` from the PicoDOS prompt (see listing 3 for an example of the picoDOS prompt) on your instrument via the serial connection. Details regarding the `usbl` command are provided in section 5.4.

9 Updating System Software

To update your system's software, please refer to the Software Update Guidelines document.

A Configuration Files[9]

A configuration file serves three purposes. First, it contains all settings required to configure the RSI data acquisition software for data collection. Upon start-up, the data acquisition software will read the configuration file and acquire data according to the contents of this file.

Second, the configuration file is used to store information required to convert the recorded raw data into physical units. This allows an RSI instrument to, for example, report the current depth based on the pressure transducer. In addition, after the data file is collected, the task of data processing is greatly simplified because the information required to convert the raw data into physical units is embedded in the data file. When a raw data file is converted into a mat-file, the configuration file that was used for the data acquisition is saved into the string variable `setupfilestr`.

Third, the configuration file can be used to hold user provided information which is available when the data file is processed. While not required by the instrument nor required for conversion into physical units, user provided data can greatly assist in organizing the collected data files and even in automating the process of data analysis. For example, the location of a profile can be added to a configuration file. When processing the resulting data files, reference to the location will be available and can be automatically extracted by your Matlab scripts.

The contents of a configuration string are accessed with the `setupstr` function. This function scans a configuration string for structured information and returns values that match a provided query. You can modify the information in the configuration string using the `extract_setupstr` and `patch_setupstr` functions, to correct some errors present at the time of data acquisition. Examples are provided in section A.3.

A.1 Anatomy of a Configuration File

Configuration files are standard ASCII[10] text files consisting of a basic structure composed of `parameters`, `sections`, and `comments`. Listing 15 shows an example of a configuration file.

Parameters

Information within a configuration file is stored as `parameters`. Parameters consists of two parts delimited by an equal sign - a `name` and a `value`.

```
name=value
```

[9] This section assumes you have some knowledge of Matlab and the ODAS Matlab library.
[10] Configuration files can only contain 7-bit characters - values from 0x00 to 0x7F.

Each line can have a maximum of one parameter. An equal sign separates the name of a parameter from its value. Extra white-space surrounding the name or value is ignored. Thus, "`name=value`" is equivalent to " `name = value` ".

Sections

Groups of parameters are identified by `sections`. A section declaration must be on its own line and consists of the `name` string enclosed by square brackets (`[ ]`).

```
[section]
```

The declaration of a section automatically ends the previous section. Sections are referenced by their *section identifier*, a value constructed in one of two ways. First, sections are scanned for a *parameter* with the name "`name`". If present, this parameter value is used as the section identifier. If not found, the *section* `name` is used as the section identifier.

To ensure sections can be uniquely referenced, each occurrence of a parameter named `name` must be unique. If a section does not contain a `name` parameter, then the name of that section must be unique, because it will be used as the section identifier.

Parameters declared before the first section declaration are automatically assigned to the root section, `[root]`. Parameters in the root section are referenced using the section name `root`, even if the root section is not explicitly declared.

Comments

All characters after a semicolon (`;`) are considered *comments* and are ignored. Comments are still valuable and can assist people in understanding the meaning and value of a parameter.

```
; A generic example showing comments.
[section]     ; Description of section.
date = value  ; This is the date of creation.
```

A.2 Adding Values

Information can be added to a configuration file but the resulting file must adhere to the format described in section A.1. In addition, RSI instruments expect certain sections and parameters to exist so care should be taken when modifying a configuration file. A configuration file that has been modified should be tested by using it for data acquisition *before* a real deployment.

If you want to add new sections to your configuration file, make sure that it has a unique name and never use a parameter with the name "`id`", because that is used in

conjunction with the [matrix] section to identify a channel and to demultiplex your data (unless, of course, that channel exists in your instrument and it is identified in the [matrix] section).

Edit existing [channel] sections to change the coefficients for conversion into physical units, particularly after installing a new sensor. You must not added a section that uses the parameter name and assign it a value that is used by RSI (see Table A.2 for a partial list of names used by RSI). In general, avoid using the parameter name by simply calling it something else, such as ship_name=, for example.

The following example depicts a portion of a configuration file that has been modified by a user to support additional parameters.

Listing 14: Partial configuration file example

```
[matrix]
num_rows = 2
row01    = 255 0 1 2 3
row02    =   0 0 1 2 3

[cruise] ; User provided section with unique name
boat     = Titanic
operator = The Unlucky Captain
date     = April 14, 1912

[channel]
id        = 8
name      = sh1
type      = shear
diff_gain = 1.045
sens      = 0.0638
adc_fs    = 4.096
adc_bits  = 16
SN        = M358  ; User provided probe SN
```

In this example, the user has created a section with the name "cruise". The name cruise is acceptable because it does not conflict with existing section names or name parameters. Inside this section, the parameters "boat", "operator", and "date" have been added.

A parameter has also been added to an existing section. The section with name channel has been appended with parameter "SN = M358". This is valid because the parameter name "SN" will not conflict with any RSI defined channel parameters. See table A.3 for a list of RSI defined channel parameters that should be avoided.

Multiple sections named channel can exist within a configuration file. These sections must contain the unique parameter "name=channel_name" so that the section can be referenced. The name parameter acts as the section identifier.

You find the value of the parameter named "SN" in the above example, by making the

following query:

```
>> probeSN = setupstr( setupfilestr, 'sh1', 'sn' )
probeSN =
    'M358'
```

Notice the value "sh1" is used to identify the section. Parameters with name "name" are unique because they are used as the section identifier.

The following query differs because the section name [cruise] is used as the section identifier:

```
>> boat = setupstr( setupfilestr, 'cruise', 'boat' )
boat =
    'Titanic'
```

A full explanation of the setupstr function can be found in the ODAS Matlab Library 4.01 User Manual.

A.3 Extracting Values

The setupstr function is used to query a configuration string for parameter values. Different results are returned depending on how it is called.

Syntax

```
>> object = setupstr( config_string )
>> sections = setupstr( config_string, 'section' )
>> value = setupstr( config_string, 'section', ...
                                    'parameter_name' )
>> [S P V] = setupstr( config_string, 'section', ...
                                      'parameter_name', ...
                                      'parameter_value' )
```

Note:

1. The section and parameter inputs are interpreted as modified regular expressions. Users do not have to understand regular expressions because queries using ASCII characters and numbers result in direct matches.
2. When section and parameter inputs are given as empty strings, (''), the setupstr function matches all values.
3. All returned values are formatted as cell arrays.

Argument	Description
config_string	the configuration string.
section	the desired section descriptor.
parameter_name	the desired parameter name.
parameter_value	the desired parameter value.
object	structure index containing the contents of the configuration file. Use in place of `config_string` to (optionally) speed up subsequent queries.
S	cell array of matching section descriptors.
P	cell array of matching parameter names.
V	cell array of matching parameter values.

Table A.1: Arguments of the Matlab `setupstr` function and their description

Finding Sections

When called with two string arguments, the `setupstr` function will return a cell array of section identifiers that match the requested section.

```
>> sections = setupstr( config_string, 'section' )
```

The requested section, `'section'`, is matched against each section identifier found in the configuration string. The returned value, "`sections`", is a cell array of matching section identifiers.

This is commonly used to test if a section exists. The function is called with `section` set to a requested section identifier and if an empty cell array is returned, there are no matches. The function can also be used to find sections. Calling the function with an empty string "`''`" for `'section'` returns a cell array containing all section identifiers.

Finding Parameter Values

When called with three string arguments, the `setupstr` function will return a cell array of parameter values that match the requested section and parameter name.

```
>> value = setupstr( config_string, 'section', ...
                                    'parameter_name' )
```

The function returns a cell array containing all parameter values where the parameter name matches `'parameter_name'` and the section identifier matches `'section'`.

Advanced Search

Calling the `setupstr` function with four arguments returns a tuple of three cell arrays. Each matching parameter has their section identifier in "`S`", parameter name in "`P`", and parameter value in "`V`".

```
>> [S P V] = setupstr( config_string, 'section', ...
                                      'parameter_name', ...
                                      'parameter_value' )
```

When queried using regular expressions, the function is useful for performing searches of a configuration string. The following command will search all sections for a parameter named "`id`" set to a valid integer within the configuration string shown in listing 14.

```
>> [S P V] = setupstr( config_string, '', 'id', '[0-9]+' )
S =
    'sh1'
P =
    'id'
V =
    '8'
```

The returned tuple contains values for each matching parameter. In this example, the regular expression "`[0-9]+`" matches all positive integers.

Faster Searches

It is possible to save the configuration string as an indexed structure to be used in future function calls. This indexed structure is used in place of the configuration string while all other arguments stay the same. The time required to perform the first function call does not change but subsequent function calls will be much faster.

```
>> obj = setupstr( config_string );
>> boat = setupstr( obj, 'cruise', 'boat' )
boat =
    'Titanic'
>> user = setupstr( obj, 'cruise', 'operator' );
user =
    'The Unlucky Captain'
```

A.3.1 `setupstr` Examples

Extracting a parameter value

Configuration File Segment:

```
[Cruise_Info]
cruise_name=Gulf of Mexico
```

Matlab Code:

```
>> cruiseName = setupstr( setupfilestr, 'cruise_info', ...
                                        'cruise_name' )
cruiseName =
    'Gulf of Mexico'
>> disp( char(cruiseName) )
Gulf of Mexico
```

Find the parameter value with the name `cruise_name` located within the section `Cruise_Info`. Display the resulting value on the console. The "`char()`" function is used to convert the cell into a string. The arguments are case in-sensitive.

Find the serial number of a shear probe

Configuration File Segment:

```
[channel]
id        = 8
name      = sh1
type      = shear
diff_gain = 1.045
sens      = 0.0638
adc_fs    = 4.096
adc_bits  = 16
SN        = M358  ; User provided probe SN
```

Matlab Code:

```
>> obj = setupstr( setupfilestr );
>> serialNumber = setupstr( obj, 'sh1', 'sn' )
serialNumber =
    'M358'
```

Read the serial number of the shear probe used for the `sh1` channel. This is an optional parameter. The function only work if the parameter is in the configuration file.

The query was made using two function calls. The first call generates an index to speed subsequent searches and the second call queries this index.

Plotting profile locations on a map

This example assumes multiple profiles have been acquired and the resulting data files have been converted into `.mat` files. The `.mat` files are located in the current di-

rectory and each has a variable called `setupfilestr` containing the configuration string.

Configuration File Segment:

```
[profile]
date = 2012/01/01
GPSx = -123.376242
GPSy = 48.44792
```

Matlab Code:

```
>> locationX = [];
>> locationY = [];
>> locationDate = {};
>> offset = 2;              % Point names are offset from points
>>                          % so they do not overlap.
>> files = dir( '*.mat' );
>> for file = files',
>>   load( file.name, 'setupfilestr' );
>>   x = setupstr( setupfilestr, 'profile', 'gpsx' );
>>   y = setupstr( setupfilestr, 'profile', 'gpsy' );
>>   t = setupstr( setupfilestr, 'profile', 'date' );
>>   locationX(end+1) = str2double( char(x) );
>>   locationY(end+1) = str2double( char(y) );
>>   locationDate{end+1} = char( t );
>> end
>> scatter( locationX, locationY )
>> text( locationX+offset, locationY+offset, locationDate )
```

This code results in a scatter plot of profile locations labelled with the profile date. So long as the operator sets the GPS coordinates within the configuration file before each profile, a map of profile locations can be generated directly from the profile data.

Listing 15: Example Configuration File

```
; [root]  This is a comment and is ignored by RSI instruments
; and software. Including comments can be helpful for users -
; this one indicates the start of the root section.
rate     = 512
disk     = c:/data
prefix   = dat_
recsize  = 1
no-fast  = 7
no-slow  = 2

[matrix]
num_rows = 4
row01 = 255  0  1  2  3  5  7  8  9
row02 =   4  6  1  2  3  5  7  8  9
row03 =  10 11  1  2  3  5  7  8  9
row04 =  32  0  1  2  3  5  7  8  9

[instrument_info]
vehicle = vmp

[cruise_info]
date    = 2011-01-15
cid     = BT201101
cname   = Bermuda Triangle
vessel  = HMS Pinafore

[channel]
id        = 10
name      = P
type      = poly
coef0     = 3.67
coef1     = 0.062076
coef2     = 4.724e-8

[channel]
id        = 32
name      = V_Bat
type      = voltage
G         = 0.1
adc_fs    = 4.096
adc_bits  = 16

; Additional channels definitions required but not shown.
```

Additional information regarding the configuration file, its sections and input parameters can be found in the ODAS Matlab Library User Manual.

ID	Signal	Typical Name
0	ADC ground signal.	Gnd
1	Accelerometer x-axis.	Ax
2	Accelerometer y-axis.	Ay
3	Accelerometer z-axis.	Az
4	Thermistor probe #1 without pre-emphasis.	T1
5	Thermistor probe #1 with pre-emphasis.	T1_dT1
6	Thermistor probe #2 without pre-emphasis.	T2
7	Thermistor probe #2 with pre-emphasis.	T2_dT2
8	Shear probe #1.	Sh1
9	Shear probe #2.	Sh2
10	Pressure transducer without pre-emphasis.	P
11	Pressure transducer with pre-emphasis.	P_dP
12	Voltage across the pressure transducer. Sometimes used for micro-conductivity sensor #1 with pre-emphasis.	PV, C1_dC1
15	EMC drive current reference signal for EM noise removal	EMC_Cur
16,17	Sea-Bird SBE3F thermometer.	SBT
18,19	Sea-Bird SBE4C conductivity sensor.	SBC
32	Voltage of the battery or power source.	V_Bat
40	X-axis from an ADIS dual-axis digital inclinometer.	Incl_X
41	Y-axis from an ADIS dual-axis digital inclinometer.	Incl_Y
42	Temperature from an ADIS dual-axis digital inclinometer.	Incl_T
48, 49	JAC conductivity sensor.	JAC_C
50	JAC thermometer.	JAC_T
52	JAC fluorometer.	Chlorophyll
53	JAC backscatter sensor.	Turbidity
64	Micro-conductivity sensor #1 without pre-emphasis.	C1
65	Micro-conductivity sensor #1 with pre-emphasis.	C1_dC1
66	Micro-conductivity sensor #2 without pre-emphasis.	C2
67	Micro-conductivity sensor #2 with pre-emphasis.	C2_dC2
80	Vertical component of magnetic field from magnetometer.	Mz
81	Horizontal component of magnetic field from magnetometer.	My
82	Horizontal component of magnetic field from magnetometer.	Mx
144	JAC Electromagnetic Velocity Digital output	U_EM
255	This special character should be present at the start of every channel matrix. It is used for error detection and correction.	Sp_Char

Table A.2: Partial list of channel ids and their typical corresponding sensors and signals.

Parameter	Description
`id`	Channel number, or address, as specified in the address matrix. This value must be unique.
`name`	String reference to this channel. This is the name that the channel will have in the mat-file. The channel will be converted into physical units regardless of the value of `name`, but most data processing software makes assumptions about the names of signals. This value must be unique. The RSI recommended names are in table A.2.
`type`	Identifies how a raw signal is converted into physical units. Most types require additional parameters for the conversion into physical units.
`diff_gain`	The gain of the analog differentiator used to apply pre-emphasis to this signal. This value is found within the instrument calibration report. In most cases, this value is already entered into the `setup.cfg`-file that was shipped with your instrument. These channels typically have the name `X_dX` where `X` is the channel name without pre-emphasis. This parameter is only used for channels with pre-emphasis.
`units`	(optional – ODAS-RT only) A string for units of this signal. Used only for real-time display with real-time telemetering instruments. This string should be short for good readability of the display.
`display`	(optional – ODAS-RT) Boolean value that determines if the record-average value of a signal is displayed within the ODAS-RT gui. Default value is `true` for all signals without pre-emphasis. Use `false` to suppress the display of a signal

Table A.3: Parameters for section `[channel]`

B Data and Log File Structure

RSI data acquisition software produces at least two files - a data file and a log file. A second data file is generated, for real-time telemetering instruments, if the data acquisition software (ODAS-RT) is configured to access an external serial device.

B.1 Data File

Data files hold the unprocessed (raw) data observed by an instrument. These binary files are generated by the data acquisition software and are given the extension "`.p`".

B.1.1 Data File Format

Data files are structured as a series of records. The first record is a configuration string constructed by prepending a header to the configuration string. Subsequent records are data records constructed by prepending a header to a data block.

B.1.2 Header

Both configuration and data records start with a 128 byte header comprised of 64, 16-bit words. This header contains the information needed to correctly read the data file. The header from every record is saved into the matrix named `header` in your mat-file with every row representing a single header.

Most entries in the header are self explanatory, see table B.1, but some entries require the additional details described below.

- The date-time entries (items 4 – 10) are set to the time of the completion of a record. This time can fluctuate by about 0.2 s because the buffer status is checked about 5 times per second. The value is truncated to 0.001 s for a real-time instrument and to the nearest second for internally recording instruments.
- Item 16 indicates a "Bad Buffer" because the check of the special character for channel 255 failed. If the check failed, the next record will have the restart (item 17) set to 1. It can take about 0.2 s for the acquisition of data to restart.
- Items 21 and 22 give the actual aggregate sampling rate. It may differ from the requested rate if it is not a whole-number divisor of the 24MHz reference clock. If so, the data acquisition software selects the nearest rate. The reference clock is accurate to 1 part per million. The sampling rate of fast channels is the aggregate sampling rate divided by the number of columns in the address `[matrix]`. The number of columns in the address matrix equals the sum of items 29 and 30.

Location	Description
1	File number - the three-digit number appended to the name of the data file.
2	Record number
3	Record number of the input on the RS-232 port, for real-time telemetering instruments (ODAS-RT only).
4	Year
5	Month
6	Day
7	Hour
8	Minute
9	Second
10	Millisecond
11	Header version (MSB: major version, LSB: minor version)
12	Configuration string size in bytes
13	Product ID (0=legacy 1=odas5ir, 2=odasrt, 3=odas4ir)
14	Build number (unique revision number from SVN)
15	Time zone as minutes from UTC
16	Buffer status (1 if special character check fails, 0 otherwise)
17	Restarted (1 if data acquisition was restarted due buffer status = 1. 0 otherwise)
18	Record header size in bytes (128)
20	Number of records written to the current file.
19	Data record size in bytes (header + data block)
21	Truncated frequency of the sampling clock (Hz)
22	Fractional part of the frequency of the sampling clock (to 0.001 Hz)
23-28	not used
29	Number of fast columns in the address `[matrix]`
30	Number of slow columns in the address `[matrix]`
31	Number of rows in the address `[matrix]`
32-62	not used
63	Profile (0=Vertical, 1=Horizontal). Not used.
64	Data type (0=unknown, 1=little endian, 2=big endian)

Table B.1: Description of fields that constitute a record header.

- Items 29 to 31 are the dimensions of the address `[matrix]`. They are used to de-multiplex the data records into individual channels.
- The profile flag (item 63) indicates the direction an instrument travels. It is no longer used.
- The final item, 64, gives the endian format of the data file. Programs that read a data file directly should use this item to read the data correctly.

B.1.3 Configuration Record

The first record within a data file is the configuration record. The record consists of a header and a copy of the configuration file used for data acquisition. The copy is a single string of ASCII characters – the configuration string. The record number is always 0. The size of this record depends on the size of the configuration string.

To read the configuration record, open the data file and read the first 64 words (128 bytes). Use the last word, item 64, to determine the endian format of the data file. Close the file and reopen it with the correct endian, if necessary. (The function `fopen_odas` does this in one call.) Use items 18 and 12 to read the first header and the configuration string.

B.1.4 Data Record

Data records consist of a header and a data block. They follow the configuration record. The header is described in section B.1.2 and the data block consists of unprocessed data from an instrument. Data words (samples of 2-byte numbers) are stored in the order provided by the address matrix. The number of words within a data block is a multiple of the size of the address matrix. The total size of a data record (header + data block) is stored in item 19 of the record header. All data records will be the same size.

To read the data records, start by reading the configuration record. The file pointer will be at the start of the first record. Use items 18 and 19 to read the header and data for the first data record. Continue reading data records until you reach the end-of-file.

B.2 Log-file

The log-file contains a record of the events that occurred during data acquisition. The log file is named `logfile.txt`. Example events from a log file are shown below:

```
50d8b024 0b 0000 dat_001.p --- Mon Dec 24 11:42:28 2012 - "acquisition start" Setup File:'H:/data/setup.cfg'
50d8b071 0c 004e dat_001.p --- Mon Dec 24 11:43:45 2012 - "acquisition stop" manual stop
50d8b09a 0b 0000 dat_002.p --- Mon Dec 24 11:44:26 2012 - "acquisition start" Setup File:'H:/data/setup.cfg'
50d8b1fd 0d 0162 dat_002.p --- Mon Dec 24 11:50:21 2012 - "bad buffer"
50d8b25e 0c 01c4 dat_002.p --- Mon Dec 24 11:51:58 2012 - "acquisition stop" manual stop
```

B.2.1 Log-file Format

Each line in a log-file describes a single event. The first part of each line is in a machine-readable form to support automated scripts. The second part of each line is in human-readable form.

Events are recorded using the following format:

```
EVENT = {MACHINE_FORMAT} --- {HUMAN_FORMAT}

MACHINE_FORMAT = {time} {event_type} {record#} {data_file_name}
  HUMAN_FORMAT = {human_time} - "{event_desc}" [additional_info]
```

The above fields are defined in Table B.2:

Field	Description
`time`	Time of the event, as given by the UNIX "`time`" function. Value is in hexadecimal format and always uses the first 8 characters.
`event_type`	Type of event identified by a two digit hexadecimal number. See section B.2.2 for a list of event types.
`record#`	The record number during which the event occurred. Formatted as a 4-character hexadecimal value.
`data_file_name`	Name of the data file. If a data file has not been created, the value "`unknown`" is used. This entry always has a width of 10 characters.
`human_time`	Time of the event.
`event_desc`	The event type.
`additional_info`	Optional extra information for the event.

Table B.2: Description of log file entries

B.2.2 Event Types

The types of events that get recorded is instrument dependent (Table B.3).

A typical log file generated by ODAS5IR is detailed in listing 16. Note, some elements were removed to fit the page. From this listing we can conclude:

- The instrument is turned on then immediately receives a start signal. This will be due to the `autoexec.bat` file automatically launching data acquisition.
- The start of data acquisition is delayed by 13 seconds. This represents the 10 second data timeout delay and the time required to read the FAT.
- The manual stop is generated by the user typing "`Q`" into the terminal. A total of 26 `(0x001a - 0x0000)` records were recorded.
- A second data file is started. This time, no "`power applied`" event is logged. This, along with the previous manual stop, implies that data acquisition was started manually via the terminal.

Code	Event Name	Description
0x01	pressure release	Release triggered because the pressure exceeded max_pressure.
0x02	fall rate release	Release triggered because the fall rate was less than $0.2\frac{\text{dbar}}{\text{s}}$ for 4 consecutive records.
0x03	timeout release	Release triggered because the duration of data acquisition exceeded max_time.
0x04	power is bad	Release triggered due to insufficient supply voltage. The *power-good* signal from the power-supply board (PS) is disabled.
0x05	power good signal received	The *power-good* signal from the PS board received. Note, this event is not used in the current revision of the data acquisition software.
0x06	switch is off	The power ON/OFF switch was set to OFF.
0x07	manual stop	Acquisition was stopped manually by an operator.
0x08	marker	Not used.
0x09	maximum file size	Data file reached its maximum allowed size.
0x0a	maximum file time	Data file reached its maximum allowed duration.
0x0b	out of disk space	Disk space was insufficient for data storage.
0x0c	acquisition start	Data acquisition started. The event time corresponds with the creation of a new data file.
0x0d	acquisition stop	Data acquisition stopped.
0x0e	bad buffer	Record contains a bad buffer.
0x0f	10s data timeout	No data observed for 10 seconds. This indicates a serious error.
0x10	start signal	A start signal was received. This occurs when ODAS5IR is first launched but before data acquistion starts.
0x11	power applied	Instrument switched on and power applied. There will be a delay of 1 to 2 seconds between when power is applied and the event is logged.
0x12	fall rate not found	No change in pressure detected 1 hour after data acquisition start. Only applicable to some instruments.
0x13	start descent	A consistent fall rate for the past 60 seconds is detected. Only applicable to some instruments.
0x14	descent stopped	Instrument fall rate has stopped. Calculated over a 30 second interval. Only applicable to some instruments.
0x15	pressure differs too much to be real	Pressure appears to jump from one value to another. Indicator of lost data packets. Only applicable to some instruments.

Table B.3: Event codes that can be in a log file.

- The second data file terminates with a "`power is bad`" event. This event indicates that the power supply is not longer supplying adequate power for data acquisition - but enough to terminate without FAT corruption.

Listing 16: Series of log events generated from a typical use of ODAS5IR.

```
11 0000 unknown   - 09:42:53 - "power applied"
10 0000 unknown   - 09:42:53 - "start signal received"
0b 0000 dat_003.p - 09:43:06 - "acquisition start" Setup File:'SETUP.CFG'
07 001a dat_003.p - 09:43:31 - "manual stop"
0c 001a dat_003.p - 09:43:31 - "acquisition stop" manual stop
10 001a unknown   - 09:43:41 - "start signal received"
0b 0000 dat_004.p - 09:43:53 - "acquisition start" Setup File:'SETUP.CFG'
04 0015 dat_004.p - 09:44:13 - "power is bad"
0c 0015 dat_004.p - 09:44:13 - "acquisition stop" power is bad
```

– End Of Document –