

# USER MANUAL

ProDAQ Data Acquisition Function Cards

## **ProDAQ 3411 24-Channel Enhanced ADC Function Card**

PUBLICATION NUMBER: 3411-XX-UM-0100



Copyright, © 2014, Bustec Production, Ltd.

Bustec Production, Ltd.  
Bustec House, Shannon Business Park, Shannon, Co. Clare, Ireland  
Tel: +353 (0) 61 707100, FAX: +353 (0) 61 707106

## **PROPRIETARY NOTICE**

**This document and the technical data herein disclosed, are proprietary to Bustec Production Ltd., and shall not, without express written permission of Bustec Production Ltd, be used, in whole or in part to solicit quotations from a competitive source or used for manufacture by anyone other than Bustec Production Ltd. The information herein has been developed at private expense, and may only be used for operation and maintenance reference purposes or for purposes of engineering evaluation and incorporation into technical specifications and other documents, which specify procurement of products from Bustec Production Ltd.. This document is subject to change without further notification. Bustec Production Ltd. Reserve the right to change both the hardware and software described herein.**

**Table of Contents**

<b>1.</b>	<b>Introduction</b>	<b>7</b>
<b>2.</b>	<b>Installation</b>	<b>9</b>
2.1.	Unpacking and Inspection	9
2.2.	Reshipment Instructions	9
2.3.	Preparing the ProDAQ Module	10
2.4.	Installing a ProDAQ Function Card	11
2.5.	Removing a ProDAQ Function Card	13
<b>3.</b>	<b>Theory of Operation</b>	<b>15</b>
3.1.	General Description	15
3.2.	Programmed Conversion	15
3.3.	Multiplexed Conversion	16
3.3.1.	Channel Mask	16
3.3.2.	Scanning	16
3.3.3.	Data Conversion	16
3.3.4.	The on-board FIFO	18
3.4.	Trigger	18
<b>4.</b>	<b>The VXI<i>plug&amp;play</i> Driver</b>	<b>19</b>
4.1.	Installation	19
4.2.	The Soft Front Panel	19
4.2.1.	Configuration	20
4.2.2.	Data Acquisition	21
4.2.3.	The Graph Controls	22
<b>5.</b>	<b>Programming the ProDAQ 3411</b>	<b>23</b>
5.1.	Connecting to the Function Card	23
5.2.	Setting Gain and Filter	24
5.3.	Acquiring single Samples	24
5.4.	Acquiring a Waveform	25
5.5.	Asynchronous Acquisition	26
5.6.	Calibration	27
<b>Appendix A:</b>	<b>Front Panel Connector</b>	<b>29</b>
<b>Appendix B:</b>	<b>Register Description</b>	<b>31</b>
<b>A.1</b>	<b>Address Map</b>	<b>31</b>
<b>A.2</b>	<b>Detailed Register Description</b>	<b>32</b>
A.1.1	FCID Register	32
A.1.2	GCSR Register	32
A.1.3	FCLLEN Register	33
A.1.4	OTRI Register	33
A.1.5	ITRI Register	34
A.1.6	DIVCLK Register	34
A.1.7	MODE Register	35

A.1.8	REPCOUNT Register	35
A.1.9	PATA Register	35
A.1.10	PATB Register	36
A.1.11	GAIFIL Register	36
<b>Appendix C:</b>	<b>Specifications</b>	<b>37</b>

**Table of figures**

<i>Figure 1 - Removing the ProDAQ Module Cover</i>	10
<i>Figure 2 - The ProDAQ Module Assembly</i>	12
<i>Figure 3 - ProDAQ 3411 Block Diagram</i>	15
<i>Figure 4 - Scan Timing</i>	16
<i>Figure 5 - Conversion Timing</i>	17
<i>Figure 6 - Multiple Channel Timing</i>	17
<i>Figure 7 - Function Card Selection</i>	19
<i>Figure 8 - ProDAQ 3411 Soft Front Panel User Interface</i>	20
<i>Figure 9 - Front End Configuration Dialog</i>	20
<i>Figure 10 - Data Acquisition Configuration Dialog</i>	21
<i>Figure 11 - Data Display</i>	21
<i>Figure 12 - Opening a Session</i>	23
<i>Figure 13 - Acquiring single Samples</i>	24
<i>Figure 14 - Acquiring a Waveform</i>	25
<i>Figure 15 - Starting the Asynchronous Acquisition</i>	26

## Reference Documents

Title	Number
ProDAQ 3120 Hardware Manual	3120-XX-HM
ProDAQ 3150 Hardware Manual	3150-XX-HM

## Glossary

<b>ADC</b>	Analog-to-Digital Converter
<b>OTD</b>	Open Thermocouple Detection
<b>MB</b>	Motherboard
<b>FC</b>	Function Card

## 1. Introduction

The ProDAQ 3411 Enhanced ADC function card provides 24 differential inputs with an input range of up to  $\pm 10V$ . The channels are multiplexed into a gain and filter stage with a software selectable gain of 1 to 1000 and three different filter settings to choose from. A 16-bit Analog-to-Digital Converter sampling with 100 kHz converts the signal. The result can be used directly or buffered in an on-board FIFO for asynchronous operation.

For calibration, the internal ground as well as the voltage reference bus from the ProDAQ motherboard can be included in the measurement as a 25<sup>th</sup> and 26<sup>th</sup> channel.

The ProDAQ 3411 function card is one of a range of function cards designed to provide full functionality when installed in one of the range of ProDAQ motherboard modules such as the ProDAQ 3120.



## 2. Installation

### 2.1. Unpacking and Inspection

The ProDAQ module is shipped in an antistatic package to prevent any damage from electrostatic discharge (ESD). Proper ESD handling procedures must always be used when packing, unpacking or installing any ProDAQ module, ProDAQ plug-in module or ProDAQ function card:

- Ground yourself via a grounding strap or similar, e.g. by holding to a grounded object.
- Discharge the package by touching it to a grounded object, e.g. a metal part of your VXIbus chassis, before removing the module from the package.
- Remove the ProDAQ module from its carton, preserving the factory packaging as much as possible.
- Inspect the ProDAQ module for any defect or damage. Immediately notify the carrier if any damage is apparent.

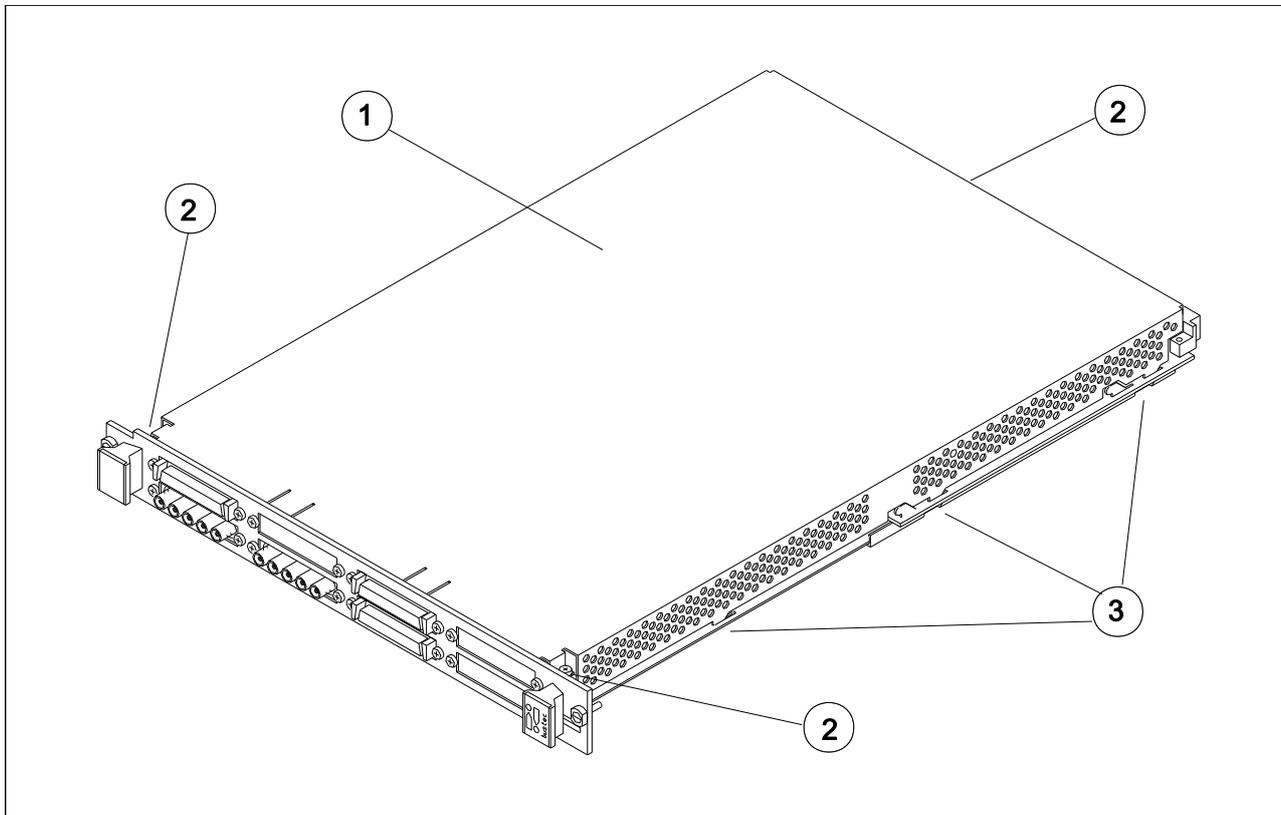
### 2.2. Reshipment Instructions

Use the original packing material when returning a ProDAQ module to Bustec Production Ltd. or calibration or servicing. The original shipping carton and the instrument's plastic foam will provide the necessary support for safe reshipment.

If the original anti-static packing material is unavailable, wrap the ProDAQ module in anti-static plastic sheeting and use plastic spray foam to surround and protect the instrument. Reship in either the original or a new shipping carton.

### 2.3. Preparing the ProDAQ Module

To install a ProDAQ Function Card into one of the ProDAQ Motherboards, you need to remove the modules top cover:



1 - Module Cover

2 - Cover Screws

3 - Cover Hooks

Figure 1 - Removing the ProDAQ Module Cover

To remove the top cover, remove the one countersunk screw in the back and the two panhead screws towards the front panel (②), that hold the cover in place. Remove the cover by sliding it out of its position towards the VXIbus connectors and up. Take special care about the hooks (③) holding it into place. Try not to lift the cover straight up. See Figure 1 for the location of the screws.

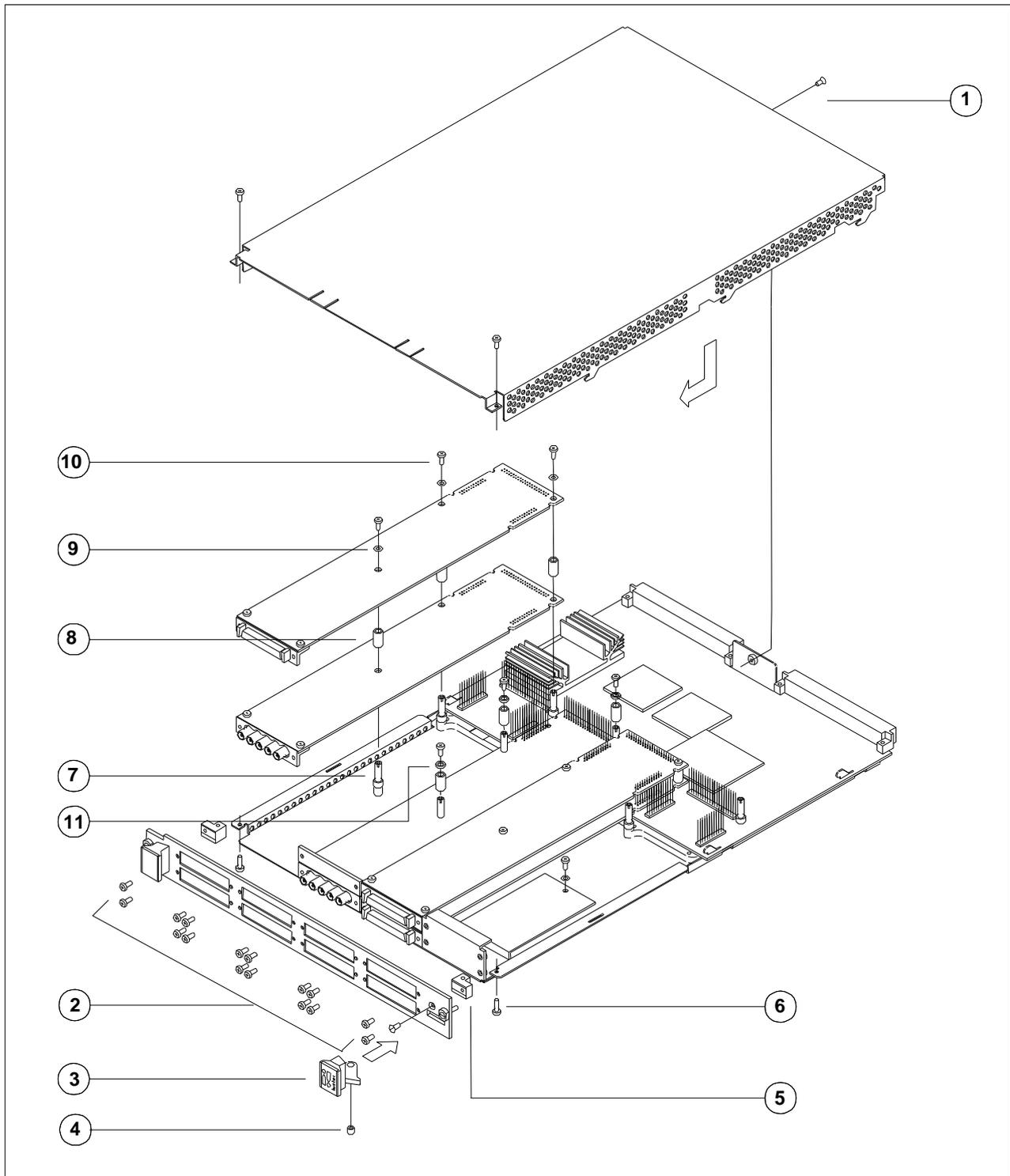
To re-install the cover, slide it back into its position by placing the small hooks over their holes and moving the cover down and forward. Secure the top cover using the two panhead screws and one countersunk screw (②).

## 2.4. Installing a ProDAQ Function Card

The ProDAQ Function Cards are arranged inside the ProDAQ Module in four stacks of two cards each. The function cards are mounted face down, e.g. the front-panel connectors as well as the motherboard connectors are underneath the PCB.

To install a ProDAQ Function Card in any of the possible positions, use the following procedure (See figure 2 for reference):

- Remove the top cover of the module as described earlier in this chapter (Fig. 2, Pos. 1).
- Remove all screws on the front-panel holding installed function cards or double filler panels in place (Fig. 2, Pos. 2). Screws holding single filler panels don't need to be removed.
- Remove the two panhead screws that mount the front panel to the modules bottom cover (Fig. 2, Pos. 6).
- Please take special care of the module handles and the rings (Fig. 2, Pos. 3 and 4), which are also fixed by those screws. The mounting angle (Fig. 2, Pos. 5) stays fixed to the front panel.
- Remove the front panel by moving it forward carefully so as to avoid bending the installed function cards.
- Choose the stack and position (lower or upper) where you want to mount the function card. If the stack, in which the function card should be installed, is covered by a double filler panel, you have to remove it before installing the function card.
- Remove the three 2.5mm panhead screws and the crinkle washers from the stack's standoffs (Fig. 2, Pos. 9 and 10 for example).
- If you want to install a function card in the upper position of a stack without having a function card in the lower position, you need to mount both spacers (Fig. 3, Pos. 11) on each standoff. If the stack is already populated with a function card in the lower position, mount only the bigger spacer (Fig. 2, Pos. 8) onto each standoff.
- Place a bayonet (supplied) on each standoff. Align the function card over these and slide carefully down. The function card should be held parallel to the modules bottom cover all the time during its way down.
- Fix the function card by mounting the three 2.5mm panhead screws and the crinkle washers onto each standoff. If you install a function card in the lower position of a stack, you need first to mount both spacers (Fig. 2, Pos. 11) onto each standoff.
- Re-mount the modules front-panel. If there is only one function card mounted in a stack, cover the remaining opening in the front panel by a single filler panel.
- Re-mount the modules top cover.



- |                          |                          |                          |
|--------------------------|--------------------------|--------------------------|
| 1 - 2.5mm Panhead Screws | 2 - 2.5mm Panhead Screws | 3 - Module Handle        |
| 4 - Ring                 | 5 - Mounting Angle       | 6 - 2.5mm Panhead Screws |
| 7 - Standoff             | 8 - Spacer               | 9 - Crinkle Washer       |
| 10 - 2.5mm Panhead Screw | 11 - 2mm Spacer          |                          |

Figure 2 - The ProDAQ Module Assembly

## 2.5. Removing a ProDAQ Function Card

Removing a ProDAQ Function Card is exactly the reverse operation then installing it. After removing the top cover and the front panel as described previously, remove the three roundhead screws that fix the function card(s) on the standoffs.

Take special care when removing the function card(s) not to bend the motherboard connectors.

After removing the function card(s), install the correct combination of spacers on the standoffs. If a stack is populated with only one function card, each of the standoffs needs to be mounted with both spacers to cover the distance between the cards as well as the PCB thickness of the missing card. If a stack is populated with two function cards, only the bigger spacer must be mounted.

Fix any remaining function card again by mounting the three panhead screws on the standoffs, re-mount the front panel and the modules cover.



## 3. Theory of Operation

### 3.1. General Description

The ProDAQ 3411 function card houses a 1:26 differential multiplexer, which is used to switch the signals from the 24 input channels and two internal channels to a gain and filter stage before converting them with a high-speed 16-bit Analog-to-Digital converter (see Figure 3 - ProDAQ 3411 Block Diagram).

A 16-bit on-board FIFO allows to constantly acquire data and generate asynchronous events to a controlling program when data is available.

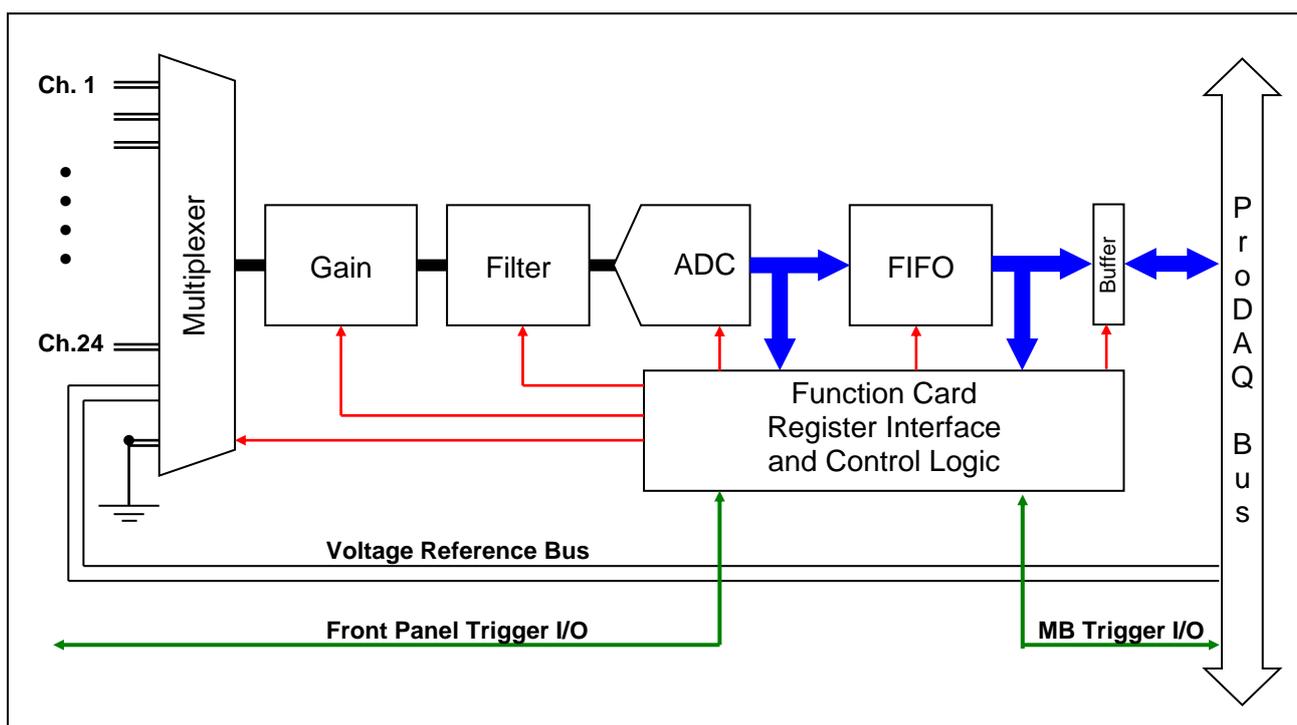


Figure 3 - ProDAQ 3411 Block Diagram

The programmable gain amplifier in the gain stage allows the user to select gains of 1, 10, 100, and 1000.

The filter stage can be switched to provide filter frequencies of 10Hz, 100Hz and 1000Hz. In addition, a bypass setting allows to directly feed the signal into the ADC without filtering.

**Remark:** Even with the filter stage switched to bypass, the multiplexer and gain stage need a minimum settling time of 51 microseconds to allow the signal at the ADC input to be stable to within 0.01% accuracy.

### 3.2. Programmed Conversion

The ProDAQ 3411 allows to separately acquire single samples on one channel at a time. By using the **ADCconv** registers, the control logic will automatically switch the multiplexer to the requested channel and start the ADC conversion. The result can be read through the **ADCvalue** register.

### 3.3. Multiplexed Conversion

To acquire consecutive samples from multiple input channels, the ProDAQ 3411 features a data acquisition mode where the control logic automatically switches the input multiplexer between the enabled channels and moves the converted values of such a scan into the on-board FIFO. The FIFO logic allows generating asynchronous signals to the application when data is available.

#### 3.3.1. Channel Mask

To specify which channels should take part in a scan a channel mask is used. The channel mask consists of two 16-bit registers (**PATA** and **PATB**), where the single bits represent the 24 input channels and the two internal channels. The two internal channels are channel 25, which is internally connected to analog ground and channel 26, which is connected to the voltage reference bus from the motherboard. Bits 0 to 15 in the **PATA** register represent channel 1 to 16, while the lower 10 bits of the **PATB** register represent channels 17 to 26. Setting a bit to one ("1") in the channel mask enables the respective channel; setting a bit to zero ("0") disables the channel.

#### 3.3.2. Scanning

When the data acquisition is started, the control logic switches the input multiplexer to the first channel enabled in the channel mask, waits for a specified time to allow the gain and filter stage to settle to this new signal level and starts the ADC to convert the signal. The resulting data word is moved into the on-board FIFO. Then the multiplexer is switched to the next enabled channel, the data is converted and so on (see Figure 4 - Scan Timing).

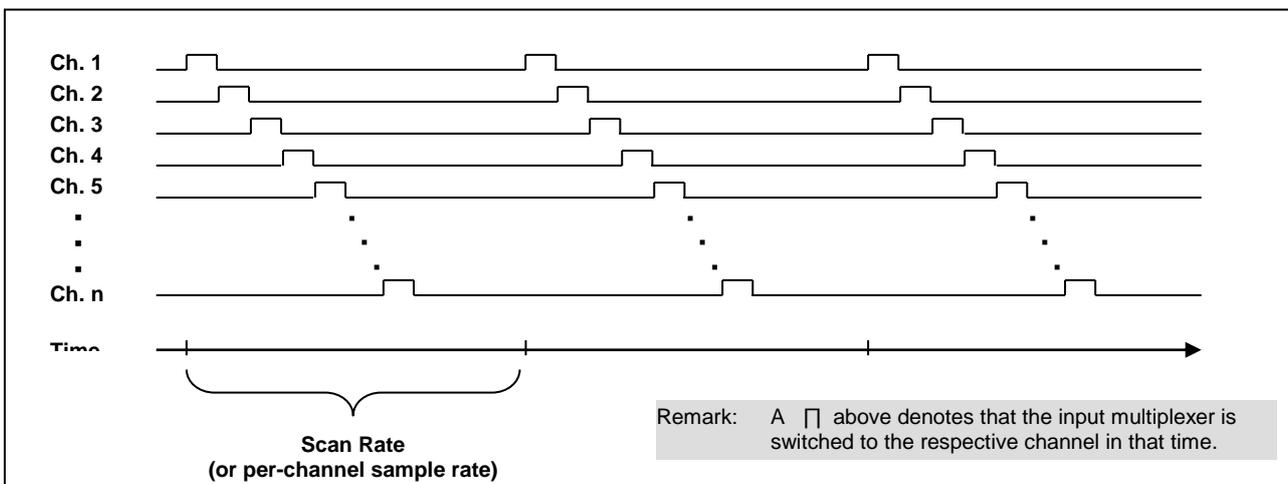


Figure 4 - Scan Timing

The timing between these scans, the scan rate, can be generated by using the on-board clock, a trigger signal from the motherboard or the front-panel input or can be generated by a software command. Each pulse of either one of these sources will start a scan over the enabled channels.

#### 3.3.3. Data Conversion

After the input multiplexer is switched to a channel, the gain and filter stages need enough time to settle to this new analog voltage level. This time is in minimum 51  $\mu\text{s}$  (filter bypass mode) and needs to be taken in consideration when setting up the card for data acquisition (see Figure 5 - Conversion Timing).

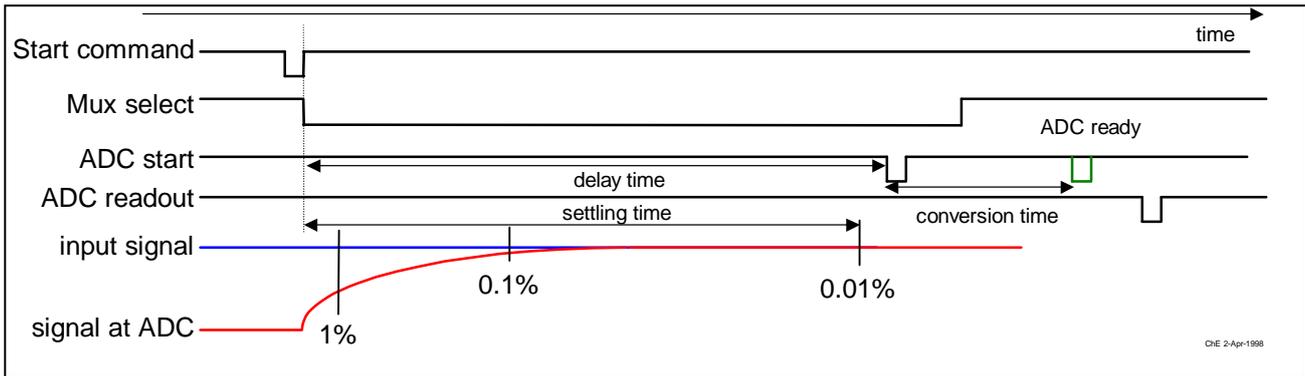


Figure 5 - Conversion Timing

The time available per channel when scanning is  $t = 1 / (\text{scan rate} * n)$ , where the scan rate is the frequency of the clock signal generating the timing for the start of the scans (in Hz) and n is the number of enabled channels. Because of the multiplexing this time can be up to 26 times shorter than the time between two consecutive scans.

Example:

Scan Rate	10 Hz
Number of channels	12
Time per channel	$1 / (10 * 12) = 8.333 \text{ ms}$

The settling time of the filter chosen must always be shorter than the time available per channel in a scan. To allow for maximum accuracy, the delay between the moment the multiplexer is switched to a channel and the start of the ADC conversion can be adjusted using the DELAY field in the **DIVCLK** register.

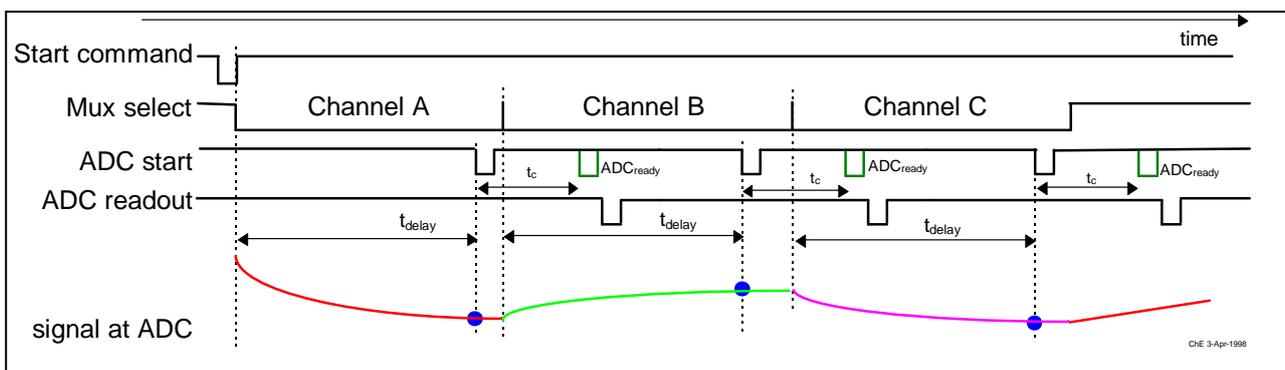


Figure 6 - Multiple Channel Timing

The internal clock for the scan rate timing is generated using a binary counter running at 312.5 kHz or 9.765 kHz. This allows the sample rate to be programmed either in multiples of 3.2  $\mu\text{s}$  or 102.4  $\mu\text{s}$ , resulting in frequencies of 10416.667 Hz down to 38.147 Hz.

The scan timing can also be generated using a trigger signal from either the front-panel trigger input or the function card trigger in from the motherboard. Which source generates the scan timing is chosen by the settings in the input trigger register (**ITRI**).

### 3.3.4. The on-board FIFO

The ProDAQ 3411 features an on-board FIFO to buffer the data generated during data acquisition. The FIFO control logic is able to generate signals when the amount of data in the FIFO reaches a certain level. Using the output trigger control register (**OTRI**), these signals can be routed to the motherboard, where they can be used to generate an asynchronous event either as VXIbus interrupt or a VXIbus trigger to inform the application of the data available.

Signal	Description
FIFO full	The FIFO is filled completely with data words.
FIFO almost full	The FIFO is nearly full. More than (FIFO size –128) words are in the FIFO.
FIFO half full	The FIFO is more than half full.
FIFO almost empty	The FIFO is nearly empty. Less than 128 words are in the FIFO
FIFO empty	The FIFO is completely empty.

Each of these signals can be enabled to generate an asynchronous event. For applications constantly acquiring data, the usage of either the 'FIFO half full' or the 'FIFO almost full' flag is recommended.

During a scan, no data is generated for channels not enabled in the data acquisition. Therefore, the number of scans fitting into the FIFO before any of the signals is generated depends on the number of channels enabled.

### 3.4. Trigger

The ProDAQ 3411 can receive trigger signals via the function card trigger line from the motherboard or from the front-panel trigger input. The trigger signal can be used either to gate or clock the data acquisition. If the trigger signal is used to gate the data acquisition, scans are performed during the time the gate is active. In the clocked mode, either the internal clock source or an external trigger is used to clock the scans.

Output trigger signals via the motherboard function card trigger out line or the front-panel trigger output can be generated from internal event such as the FIFO level signals (see 3.3.4) or the scan clock generated by the internal clock divider. In addition, the front-panel input trigger or the input trigger from the motherboard can be routed to the trigger output (motherboard or front-panel).

## 4. The VXIplug&play Driver

### 4.1. Installation

The ProDAQ 3411 24-Ch. ADC function card is supplied with a *VXIplug&play* driver. To install the driver, run the “Setup.exe” application coming with it and follow the instructions presented. Make sure that no other ProDAQ software is running when you start the setup.

The installation program will by default perform a complete installation. It will install the driver files in the directory defined by the %VXIPNPPATH% environment variable and shortcuts into the VXIPNP program group of the start menu. To choose a different path and/or custom installation options is not recommended and may result in malfunctioning of the soft front panel and any application trying to use the driver.

### 4.2. The Soft Front Panel

The purpose of soft front panel application is to demonstrate the instrument’s abilities. After the start of the soft front panel application, the user will be presented with a dialog box showing all available ProDAQ 3411 instruments in a system, allowing the selection of one instrument to connect to (see Figure 7 - Function Card Selection). The soft front panel is not designed to handle more than one instrument at a time. If there is only one instrument available, the dialog box will not appear and the soft front panel application will automatically establish the communication to this instrument.



Figure 7 - Function Card Selection

If no ProDAQ 3411 is available in your system, the soft front panel application can be run in demo mode, allowing to operate all controls as if connected to a 3411.

After initializing the ProDAQ 3411 function card, during which a splash screen is displayed, the soft front panel window will appear (see Figure 8 - ProDAQ 3411 Soft Front Panel User Interface). Using the controls it allows you to configure the card and then acquire and display waveforms.

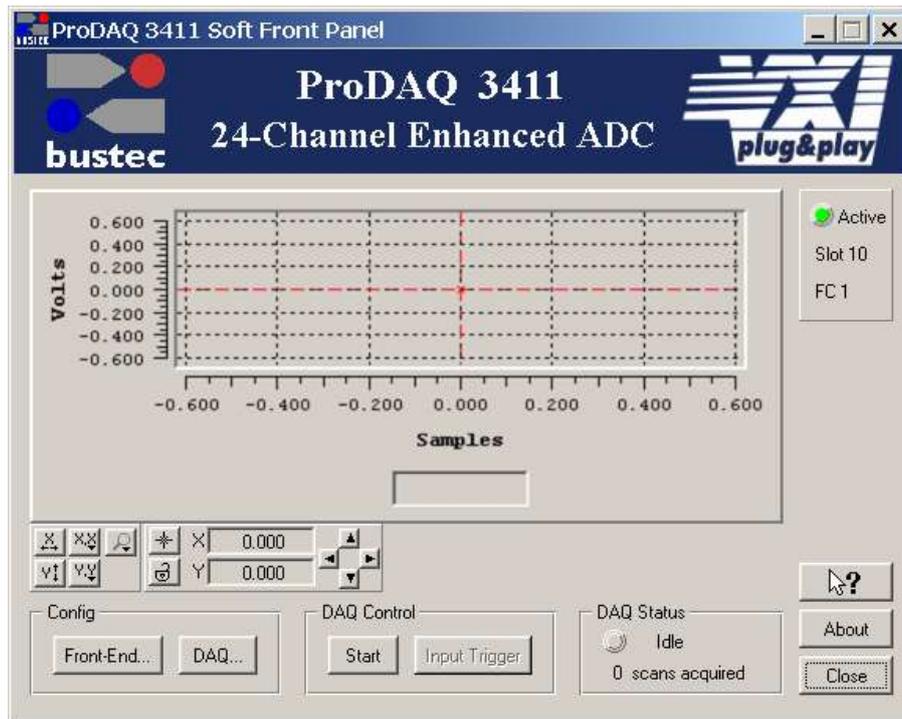


Figure 8 - ProDAQ 3411 Soft Front Panel User Interface

#### 4.2.1. Configuration

The “Config” field in the lower left corner of the panel holds two buttons, which allow opening additional dialogs to configure the function cards front end and the data acquisition parameters. The controls in the “ProDAQ 3411 Front-End Cfg” dialog are equivalent to the function call `bu3411_setFEconf()` as implemented by the driver (see 5.2 - Setting Gain and Filter).

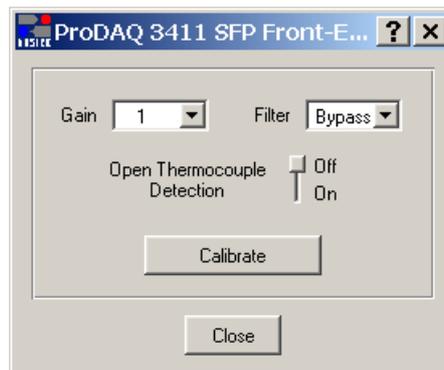


Figure 9 - Front End Configuration Dialog

In addition the button “Calibrate” allows you to start the calibration of the ProDAQ 3411 as implemented by the driver function `bu3411_calibrate()`.

The button “DAQ...” opens the “ProDAQ 3411 SFP DAQ Configuration” dialog, which allows you to choose the parameters for the data acquisition as used by the driver functions `bu3411_acquireWaveform()`, `bu3411_acquireWaveforms()` and `bu3411_startAcquisition()`.

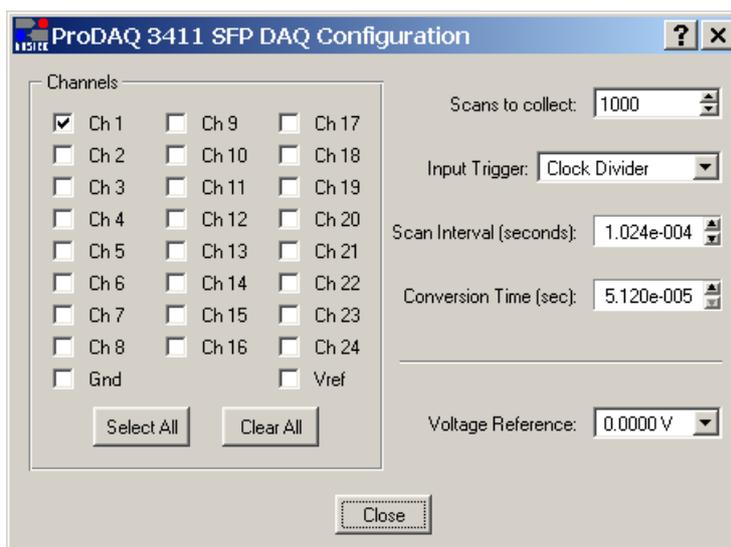


Figure 10 - Data Acquisition Configuration Dialog

The check boxes in the “Channels” field allow define the channel mask, enabling or disabling a channel for the data acquisition. The other controls allow you to choose the number of scans to collect, the source for the scan clock (“Input Trigger”) and to adjust the acquisition timing.

#### 4.2.2. Data Acquisition

The data acquisition is started by selecting the button “Start” in the “DAQ Control” field. The soft front panel will use the function `bu3411_acquireWaveforms()` with the parameter defined in the DAQ configuration dialog to acquire waveforms for the selected channels and displays them:

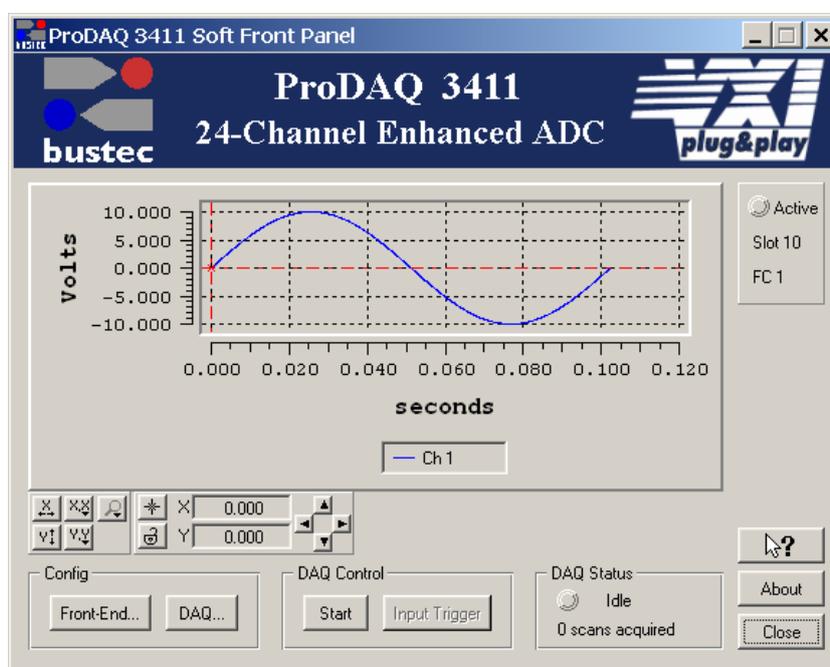


Figure 11 - Data Display

If the input trigger for the data acquisition was set to “Software”, the button “Input Trigger” can be used to generate a software trigger to start the scans.

### 4.2.3. The Graph Controls

The graph controls below the data display can be used to change the scaling of the display or the precision of the labels shown. They allow you to zoom in and out for a more detailed display. A cursor can be activated selecting a point in the display by using the left mouse button. The cursor coordinates are shown below the graph.

<u>Control</u>	<u>Description</u>
	Switches the X scale of the display to autoscale mode.
	Switches the Y scale of the display to autoscale mode.
	Displays a pull down menu allow to select the format and precision of the labels on the X scale.
	Displays a pull down menu allow to select the format and precision of the labels on the Y scale.
	Displays a pull down menu allowing to enable interactive zoom either in horizontal or vertical direction only, as an area or let you disable the zoom mode.
	Brings the current position of the cursor to the center of the display.
	Locks the cursor to the data displayed. The cursor will automatically snap to the nearest data point on any of the displayed waveforms.

## 5. Programming the ProDAQ 3411

This chapter shows how to program the ProDAQ 3411 function card using the *VXIplug&play* driver. Complete examples can be found in the “Examples” subdirectory of the driver. All functions are explained in detail in the help file coming with the driver.

### 5.1. Connecting to the Function Card

To initialize the driver and connect to the ProDAQ motherboard, the standard *VXIplug&play* initialization function `bu3411_init()` is used (see Figure 12, ①). (Please refer to the *VXIplug&play* standard VPP-4.3, section 4.3 for a detailed description of the address string used.)

After initializing the driver and connecting to the motherboard, the driver must be told which one of the eight possible function cards on a ProDAQ motherboard to work with. This is done by the function `bu3411_fcSelect()`. It takes as an argument the session established via the function `bu3411_init()`, the function card number and a boolean value specifying whether to reset the selected function card (see Figure 12, ②).

```
#include <visa.h>
#include <bu3411.h>

main (int argc, char **argv)
{
    ViStatus status;
    ViSession session;
    ViChar descr[256];

    ① if ((status = bu3411_init("VXI0::2::INSTR", VI_TRUE, VI_TRUE, &session)) != VI_SUCCESS)
    {
        viStatusDesc (rm_session, status, descr);
        printf ("Error: bu3411_init() failed due to %s\n", descr);

        return -1;
    }

    ② if ((status = bu3411_fcSelect(session, 1, VI_TRUE)) != VI_SUCCESS)
    {
        viStatusDesc (instr_session, status, descr);
        printf ("Error: bu3411_fcSelect failed due to %s\n", descr);

        return -1;
    }

    /* OR: */

    if ((status = bu3411_paramInit("VXI0::2::INSTR", 1, VI_TRUE, VI_TRUE, &session)) != VI_SUCCESS)
    ③ {
        viStatusDesc (rm_session, status, descr);
        printf ("Error: bu3411_paramInit() failed due to %s\n", descr);

        return -1;
    }

    /* ... */
}
```

Figure 12 - Opening a Session

For your convenience, the driver contains a new function called `bu3411_paramInit()`, which combines the functionality of the `bu3411_init()` and `bu3411_fcSelect()` functions by extending the argument list of the standard initialization function with a parameter specifying the function card number (see Figure 12, ③).

For the driver functions to work properly, you will either have to use the function `bu3411_paramInit()` to open a session with the device, or you will have to call the function `bu3411_fcSelect()` after calling the function `bu3411_init()` and before any other driver function is called.

To close a session with the ProDAQ 3411 24-Ch. ADC function card, the standard VXIplug&play function `bu3411_close()` must be used.

## 5.2. Setting Gain and Filter

The filter and gain stages on the ProDAQ 3411 function card are configured using the function `bu3411_setFEconf()`. It takes as arguments the session to the instrument, a value for the gain setting, a value for the filter setting and a flag, which switches on or off the OTD (Open Thermocouple Detection). The gain can be set to 1, 10, 100 and 1000 using macros predefined in the include file `bu3411.h`. The filter can be set to 10 Hz, 100 Hz, 1000 Hz and bypass. Again, macros predefined in `bu3411.h` should be used (see Figure 13, ①).

## 5.3. Acquiring single Samples

The VXIplug&play driver for the ProDAQ 3411 24-Ch. ADC function card provides two functions for acquiring single samples from the inputs. For acquiring a sample from a single channel, the function `bu3411_sampleChannel()` is used. It takes a channel number and returns the measured value in volts. The function `bu3411_sampleChannels()` can be used to acquire a single sample from more than one channel at a time. It takes a channel mask as parameter and returns the measured values in an array (see Figure 13, ② and ③).

```

/* ... */

/* set the gain to 10, the filter to 10 Hz and disable OTD */
① if ((status = bu3411_setFEconf (session, bu3411_GAIN_10, bu3411_FILT_10HZ, 0)) < VI_SUCCESS)
{
    bu3411_error_message (rm_session, status, descr);
    printf ("Error: bu3411_setFEconf () failed due to %s\n", descr);

    return -1;
}

/* measure channel 1 */
② if ((status = bu3411_sampleChannel (session, 1, &val)) != VI_SUCCESS)
{
    bu3411_error_message (rm_session, status, descr);
    printf ("Error: bu3411_sampleChannel failed due to %s\n", descr);

    return -1;
}

/* measure channels 1, 2, 4 and 8 */
③ if ((status = bu3411_sampleChannels (session, 0x8B, arr_val)) != VI_SUCCESS)
{
    bu3411_error_message (rm_session, status, descr);
    printf ("Error: bu3411_sampleChannels () failed due to %s\n", descr);

    return -1;
}

/* ... */

```

Figure 13 – Acquiring single Samples

The channel mask is a 32-bit integer, whose first 26 bits are used to specify whether a channel shall be included in the scan or not. Bit 0 represents channel 1, bit 1 channel 2 and so on. Bit 24 represents channel 25, which is connected internally to Ground, and bit 25 channel 26, which is internally routed to the voltage reference bus from the motherboard.

The value for each channel has a fixed position in the output array, independent whether the respective channel is specified for sampling by the channel mask or not. For all channels not specified, a value of 0.0 (zero) is written to the position of the channel in the array.

## 5.4. Acquiring a Waveform

To acquire a consecutive number of samples from a single channel or several channels, the functions `bu3411_acquireWaveform()` and `bu3411_acquireWaveforms()` can be used. The functions take either a channel number or a channel mask similar to the functions `bu3411_sampleChannel()` and `bu3411_sampleChannels()` as an argument to specify which channel or group of channels to read from. In addition a number of samples to specify the consecutive number of samples that will be read per channel, a sample rate and an output array used to store the waveform(s):

```

{
    ViSession session;
    ViInt32 mask;
    ViReal64 waveform[1024];

    /* .... */

    ① /* acquire a waveform of 1024 samples from channel 3 at 1 kHz */
    if ((status = bu3411_acquireWaveform (session, 3, 1024, 1000.0, waveform)) < VI_SUCCESS)
    {
        bu3411_error_message (rm_session, status, descr);
        printf ("Error: bu3411_acquireWaveform() failed due to %s\n", descr);

        return -1;
    }

    /* acquire waveforms from channels 1-8, 18, 20, Ground and Volt. Ref. */
    ② mask = 0x020900FF;
    if ((status = bu3411_acquireWaveforms (session, mask, 1024, 1000.0,
                                           bu3411_GROUP_BY_CHANNEL, waveform)) != VI_SUCCESS)
    {
        bu3411_error_message (rm_session, status, descr);
        printf ("Error: bu3411_acquireWaveforms() failed due to %s\n", descr);

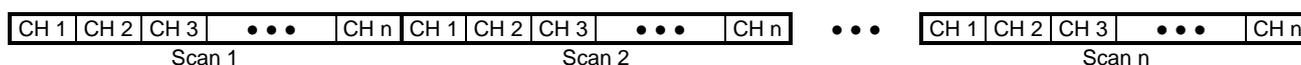
        return -1;
    }

    /* ... */
}

```

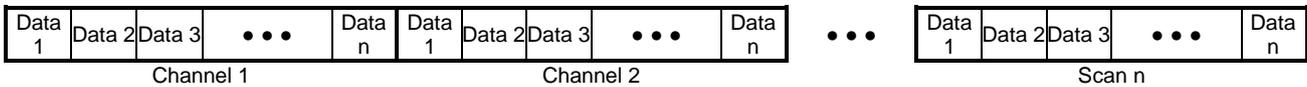
Figure 14 – Acquiring a Waveform

The function `bu3411_acquireWaveforms()` has an additional argument specifying the arrangement of the data in the output array. The ADC is performing scans by switching the input multiplexer to each channel with the speed defined by the given sample rate. So the arrangement of the data as read from the on-board FIFO is



In opposite to the function `bu3411_sampleChannels()`, the values for the different channels have no specific location in the output stream. The number of values per scan depends on the number of channels enabled in the channel mask. If for example channels 1-8, 18, 20, Ground and Vref as in the above example are enabled, each scan delivers 12 values.

This is also the arrangement of the data in the output array when the parameter *fillMode* is specified as `bu3411_GROUP_BY_SCAN`. But most of the time it is more convenient to have the data arranged on a per channel basis. Therefore, the function `bu3411_acquireWaveforms()` will rearrange the data while transferring it to the output array when the parameter *fillMode* is specified as `bu3411_GROUP_BY_CHANNEL`. The result is an arrangement like



The complete number of samples as specified by the parameter *numberOfScans* for the first enabled channel is placed into the output array, then the complete number of samples for the second enabled channel and so on.

## 5.5. Asynchronous Acquisition

To acquire data continuously, the ProDAQ 3411 needs to be configured for scanning the input channels and moving the data into the on-board FIFO. The FIFO memory stores the data until the host computer is ready to read out the data. The timing for this asynchronous read-out depends on the amount of data in the FIFO.

The driver function `bu3411_startAcquisition()` can be used to configure the card for scanning and starting the acquisition. The parameter *channelMask* defines which channels should be enabled for scanning. The parameter *inputTriggerSource* selects the source for the scan clock. If the internal clock is chosen, the parameter *scanRateHz* defines the scan rate used. The parameter *conversionTime* can be used to adjust the timing as described in “3.3.3 - Data Conversion”. A special value, defined as `bu3411_ADJUST` in the header file allows the driver to automatically choose the correct value.

```

{
    ViSession session;

    /* .... */

    /*
     * Start the asynchronous acquisition for channels 1-8, 18, 20, Ground and Volt. Ref.
     * using the internal clock at a rate of 100 Hz. The conversion time is automatically
     * adjusted by the driver.
     */

    if ((status = bu3411_startAcquisition (session, 0x020900FF,
                                         bu3411_ITRI_SRC_INT,
                                         100.0, bu3411_ADJUST)) < VI_SUCCESS)
    {
        bu3411_error_message (rm_session, status, descr);
        printf ("Error: bu3411_startAcquisition () failed due to %s\n", descr);

        return -1;
    }

    /* ... */
}

```

Figure 15 – Starting the Asynchronous Acquisition

To read out the acquired data at the right time, the application can either poll the status of the acquisition using the function `bu3411_checkAcquisition()` or install a callback prior to starting

the acquisition using the function `bu3411_installAcquisitionCallback()`. If a callback is installed, the driver will use the FIFO flags to generate an asynchronous event that will activate the callback function.

When data is available, the function `bu3411_readAcquisition()` can be used to read out the data acquired. It takes as parameters the number of scans to read, the output buffer and a fill mode as described above for the functions `bu3411_acquireWaveform()` and `bu3411_acquireWaveforms()`. It also returns the actual number of scans read and the number of scans still in the on-board FIFO. See the example "AsynchAcquisition" coming with the driver for a complete example how to use these functions.

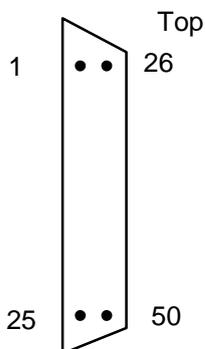
## 5.6. Calibration

The driver provides the function `bu3411_calibrate()` to calibrate the ADC using the on-board ground reference and the motherboard voltage reference. The results of the calibration are stored internally in the driver and automatically applied when using the high-level functions as for example `bu3411_acquireWaveform()` or `bu3411_readAcquisition()`. If the motherboard housing the ProDAQ 3411 function card is not equipped with a voltage reference, the function performs only an offset calibration.



## Appendix A: Front Panel Connector

The front panel connector used on the ProDAQ 3411 is a 50-pin female SCSI with the following pin-out:



Signal	Pin #		Signal
Trigger out	1	26	Trigger in
CH24 -	2	27	CH24 +
CH23 -	3	28	CH23 +
CH22 -	4	29	CH22 +
CH21 -	5	30	CH21 +
CH20 -	6	31	CH20 +
CH19 -	7	32	CH19 +
CH18 -	8	33	CH18 +
CH17 -	9	34	CH17 +
CH16 -	10	35	CH16 +
CH15 -	11	36	CH15 +
CH14 -	12	37	CH14 +
CH13 -	13	38	CH13 +
CH12 -	14	39	CH12 +
CH11 -	15	40	CH11 +
CH10 -	16	41	CH10 +
CH9 -	17	42	CH9 +
CH8 -	18	43	CH8 +
CH7 -	19	44	CH7 +
CH6 -	20	45	CH6 +
CH5 -	21	46	CH5 +
CH4 -	22	47	CH4 +
CH3 -	23	48	CH3 +
CH2 -	24	49	CH2 +
CH1 -	25	50	CH1 +



## Appendix B: Register Description

### A.1 Address Map

All addresses are given in a hexadecimal notation. "FC Address" specifies the address in the internal function card address space of the motherboard. "VXI Offset" is the offset to be used when accessing this register directly via the function card space mapped into the VXI memory space of the motherboard.

FC Address	VXI Offset	Register Name	Access	Function
Control Register				
0	0	FCID_REG	RO	ID register for automatic board identification
1	4	---	-	Not used
2	8	GCSR	RWC	General control and status register
3	C	FCLEN	RO	Size of installed FIFO (2048 or 16384 kWords)
4	10	---	-	Not used
5	14	OTRI	RW	Output trigger control
6	18	ITRI	RW	Input trigger control
7	1C	DIVCLK	RW	Clock divider for use in the stand-alone mode
8	20	MODE	RW	Operation mode register
9	24	REPCOUNT	RW	Repetition Counter
A	28	---	RW	Not used
B	2C	PATA	RW	Channel mask for channels 1 to 16
C	30	PATB	RW	Channel mask for channels 17 to 26
D	34	---		Not used
E	38	GAIFIL	RW	Gain and filter settings
80	200	ADCvalue	RO	Last converted ADC value
C0-D9	300-364	ADCconv	WO	Start ADC conversion for single channel
Memory Space				
4000-FFFF	20000-3FFFC	FIFO	RW	FIFO access area

All registers are 16-bit wide.

## A.2 Detailed Register Description

### A.1.1 FCID Register

The function card ID register contains a unique number used to identify the card.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Initial	1	0	1	0	0	0	0	0	1	1	0	1	1	0	0	0
Content	Function Card ID (0xA0D8)															

### A.1.2 GCSR Register

General control and status register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	RC	RC	RC	RC	RO	RO	RO	RC	--	--	RWC	RW	RWC	RW	RWC	WC
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Content	FIFO_EM	TRIG_I	REPZERO	ADC_TMO	FIFO_AE	FIFO_HF	FIFO_AF	FIFO_FF	ADC_READY	ADC_ACTIVE	SINGLE	TRG_START	ENA	START	STOP	RESET

<b>RESET</b>	Resets the function card.
<b>STOP</b>	Writing a one ("1") to this bit stops the data acquisition. The bit is automatically cleared when the data acquisition has stopped.
<b>START</b>	Writing a one ("1") to this bit starts the data acquisition. The bit is automatically cleared when the data acquisition is finished or has been stopped.
<b>ENA</b>	Enables the data acquisition.
<b>TRG_START</b>	Enables the data acquisition to react on the trigger selected by the ITRI register.
<b>SINGLE</b>	Writing a one ("1") to this bit will start the data acquisition for one scan only.
<b>FIFO_FF</b>	A one ("1") indicates that the FIFO memory is full.
<b>FIFO_AF</b>	A one ("1") indicates that the FIFO memory is almost full (less than 128 words free).
<b>FIFO_HF</b>	A one ("1") indicates that the FIFO memory is half full.
<b>FIFO_AE</b>	A one ("1") indicates that the FIFO memory is almost empty (less than 128 words in FIFO).
<b>ADC_TMO</b>	An ADC timeout occurred.
<b>REPZERO</b>	A one ("1") in this bit indicates that the internal repetition counter is zero. This bit is cleared by reading, but also by starting the data acquisition or reset.
<b>TRIG_I</b>	A one ("1") indicates that an input trigger as selected by the OTRI register was detected. This bit is cleared by reading, but also by starting the data acquisition or reset.
<b>FIFO_EM</b>	A one ("1") indicates that the FIFO memory is empty.

### A.1.3 FCLEN Register

Specifies the FIFO size in words.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Content	FIFO size 3411-AA: 0x0800 (2048 words) 3411-AB: 0x4000 (16384 words)															

### A.1.4 OTRI Register

Register to configure the output trigger.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	--	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Content	n/u	inSelect			Pol	FPo	OMP	OMB	SWT	MBT	DivC	iSel		ADC conv	DAQ start	FPin

- FPin                                      A one ("1") selects the front panel trigger input as the source
- ADCconv                                 A one ("1") enables the start of the conversion to generate a signal
- DAQstart                                A one ("1") enables the start of the acquisition to generate a signal
- iSel                                        A one ("1") enables the internal trigger selector
- DivC                                       A one ("1") enables the system clock divider as the trigger source
- MBT                                       A one ("1") enables the trigger from the motherboard
- SWT                                        software trigger
- OMB                                       A one ("1") enables the trigger output to the motherboard as a level
- OMP                                        A one ("1") enables the trigger output pulse to the motherboard.
- FPo                                        A one ("1") enables trigger to the front panel
- Pol                                        allows to change the polarity of the output. A one ("1") selects an active high output, while a zero ("0") select an active low output.
- inSelect                                 allows selection of the trigger output one of the following internal sources

Value	Source enabled
0	Data acquisition ready
1	Repeat counter is zero
2	FIFO full
3	FIFO almost full
4	FIFO almost empty
5	FIFO half full
6	ADC ready
7	Receptive mode counter pulse

### A.1.5 ITRI Register

Configures the input trigger of the function card.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	--	--	--	--	RW	RW	RW	RE	RW	RW	RW	--	--	--	--	--
Initial	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Content	n/u	not used			POL	FPo	Edge	FPin	SWT	MBT	DIVC	not used				

**DIVC** Writing a one ("1") to this bit enables the sample clock as specified in the DIVCLK register as a source for the input trigger.

**MBT** Enables the input trigger from the motherboard as a source for the input trigger.

**SWT** Writing a one ("1") to this bit activates the input trigger (Software Trigger).

**FPin** Enables trigger from front panel

**Edge** always zero ("0")

**FPo** Enables trigger output to the front panel

**POL** Sets the polarity of the output.

### A.1.6 DIVCLK Register

This register holds the divider value for deriving the sample clock from the internal clock source.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Content	DIVCLK								DELAY							

**DELAY** defines the time between the input multiplexer switching to the next channel and the ADC start. Valid range is 2...255 resulting in the following ranges (depending on the setting of the DSC field in the MODE register):

DSC	Range
1.6 $\mu$ s	3.2 $\mu$ s to 408.0 $\mu$ s
25.6 $\mu$ s	51.2 $\mu$ s to 6528 $\mu$ s
409.6 $\mu$ s	819.2 $\mu$ s to 104.448 ms
6553.6 $\mu$ s	13.1072 ms to 1671.168 ms

**DIVCLK** Base clock selection for the data acquisition (scanning). The valid range is 2...255 resulting in the following ranges (depending on the CSC field in the MODE register):

CSC	Range
3.2 $\mu$ s	9.6 $\mu$ s to 819.2 $\mu$ s
102.4 $\mu$ s	307.2 $\mu$ s to 26214.2 $\mu$ s

### A.1.7 MODE Register

The function card ID register contains a unique number used to identify the card.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Operation</b>	RW	RW	RW	--	RW	--	RW	RW	RW	RW	RW	RW	--	--	--	--
<b>Initial</b>	0	0	0	X	0	X	0	0	0	0	0	0	0	0	0	0
<b>Content</b>	FIDC	FIDD	ADAI	n/u	ATC	n/u	DSC		CSC	n/u	FIRS	FIEN	not used			

<b>FIEN</b>	Enables the FIFO for pattern capture/generation.
<b>FIRS</b>	Resets the FIFO.
<b>CSC</b>	Selects the base unit for the data acquisition clock (DIVCLK). A zero ("0") selects a base clock of 3.2 $\mu$ s, a one ("1") a base clock of 102.4 $\mu$ s.
<b>DSC</b>	Selects the base unit for the delay timer (DELAY). A zero ("00") selects a base unit of 1.6 $\mu$ s, a one ("01") a base unit of 25.6 $\mu$ s, a two ("10") a base unit of 409.6 $\mu$ s and a three ("11") a base unit of 6553.6 $\mu$ s.
<b>ATC</b>	Enables the data acquisition in continuous mode.
<b>ADAI</b>	Enables the data acquisition to run infinite. Otherwise it will stop after the number of scans programmed in the REPCOUNT register.
<b>FIDD</b>	Disable the FIFO drop mode (enabled by default).
<b>FIDC</b>	enable FIFO clock.

### A.1.8 REPCOUNT Register

This register holds the repetition counter for the data acquisition.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Operation</b>	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
<b>Initial</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>Content</b>	Repetition Counter															

The repetition counter is a 16-bit down counter used to specify how many scans shall be performed. The number of scans performed is the register value plus one.

### A.1.9 PATA Register

Defines which channel takes part in the data acquisition. The PATA register holds the mask for channels 1 ...16.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Operation</b>	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW						
<b>Initial</b>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>Content</b>	CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1

### A.1.10 PATB Register

Defines which channel takes part in the data acquisition. The PATB register holds the mask for channels 17 ...24, 25 (Ground) and 26 (Voltage Reference).

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	--	--	--	--	--	--	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	0
Content	not used						VREF	GND	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17

### A.1.11 GAIFIL Register

Holds the settings for gain, filter and open thermocouple detection.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	--	--	--	--	--	--	--	--	RW	RW	RW	RW	RW	RW	RW	RW
Initial	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0
Content	not used								OTD	F2	F1	F0	G3	G2	G1	G0

#### G3 – G1

Gain selection:

G3	G2	G1	G0	Gain
0	0	0	0	1
0	0	0	1	10
0	0	1	0	100
0	0	1	1	1000

#### F2 – F0

Filter selection:

F2	F1	F0	Filter
0	0	0	1000 Hz
0	0	1	100 Hz
0	1	0	10 Hz
1	1	1	Bypass

#### OTD

A one ("1") enables the open thermocouple detection.

## Appendix C: Specifications

<b>Number of Input Channels</b>	24
<b>Input Type</b>	Differential
<b>Input Voltage Range</b>	$\pm 10$ mV to $\pm 10$ V
<b>Gain Selection</b>	1, 10, 100, or 1000
<b>Filter Selection</b>	10 Hz, 100Hz, 1000 Hz or Bypass
<b>Dynamic Range</b>	92 dB
<b>Integral Linearity Error</b>	$\pm 0.01\%$ typ. ( $\pm 0.02\%$ max)
<b>Differential Linearity Error</b>	-1 LSB to +1.5 LSB
<b>Total Full Scale Error</b>	< 0.2% (uncalibrated) < 0.02% (calibrated using ProDAQ 3201)  Note: measured at $25^{\circ}\text{C} \pm 1^{\circ}\text{C}$
<b>Settling Time</b>	minimum 51 $\mu\text{s}$ to 0.01%
<b>Input Impedance</b>	> 10M $\Omega$ / 25 pF
<b>Input Protection</b>	max. $\pm 35\text{V}$
<b>Input Coupling</b>	DC
<b>Input Offset Voltage</b>	$\pm 1$ mV typ. (Gain 1) $\pm 5$ mV max. (Gain1)
<b>CMRR</b>	110 dB @ 1kHz (typ.) 80 dB @ 1 kHz (min.)
<b>Thermal Drift</b>	$\pm 1$ LSB / $10^{\circ}\text{C}$
<b>Noise</b>	< 1 $\mu\text{V}$ RMS (Gain 1000, Filter 10 Hz)
<b>FIFO</b>	2 kWord (-AA) or 16 kWord (-AB)
<b>Trigger Input</b>	Motherboard or Front Panel
<b>Trigger Output</b>	Motherboard or Front Panel
<b>Current Consumption</b>	190mA @ 5V 40 mA @ +12V 20 mA @ +15V 10 mA @ -15V  Note: $\pm 15\text{V}$ are derived on the motherboard from $\pm 24\text{V}$ .
<b>Power Consumption</b>	< 1.9 Watt
<b>Connector</b>	50-pin SCSI
<b>Dimensions</b>	230mm x 53mm (9.1inch x 2.1inch)
<b>Weight</b>	< 100 g.

<b>Operating Temperature</b>	0°C to 50°C
<b>Storage Temperature</b>	-40°C to 70°C
<b>Warm-up Time</b>	< 30 min.
<b>MTBF</b>	121980 Hrs.
<b>Software Support</b>	VXI <i>plug&amp;play</i> Driver



