IIADC Module, C and Fortran Libraries for the ADdsp Analogue to Digital Converter Expansion Card

for Acorn Risc OS-Based Computer Systems

User Guide and

Programmer's Reference Manual

Intelligent Interfaces Ltd

Issue 2 - April 1997

All information is given by Intelligent Interfaces in good faith. However, it is acknowledged that there may be errors or omissions in this User Guide. Intelligent Interfaces welcomes comments and suggestions relating to the IIADC Module and this User Guide.

All correspondence should be addressed to:-

Technical Enquiries
Intelligent Interfaces Ltd
P O Box 80
Eastleigh
Hampshire
SO53 2YX

Tel: 023 8026 1514
Fax: 087 0052 1281
Email: support@intint.demon.co.uk

This User Guide is intended only to assist the reader in the use of the IIADC Module and, therefore, Intelligent Interfaces shall not be liable for any loss or damage whatsoever arising from the use of any information or particulars in, or any error or omission in, this User Guide, or any incorrect use of the IIADC Module or the ADdsp Analogue to Digital Converter Expansion Card.

Contents

Introduction

Each ADdsp Analogue to Digital Converter Expansion Card provides any Acorn Risc OS-based computer, with an expansion backplane and Risc OS 3.1 or later, with eight analogue input channels, an eight bit digital input port and an eight bit digital output port.

The single-ended analogue input channels, each with an input voltage range of -5 to +5 volts, are multiplexed into a single sample and hold amplifier and a fast 12-bit successive approximation analogue to digital converter. A First In First Out (FIFO) memory is used to buffer the output data from the analogue to digital converter. The FIFO interrupts the processor when it is half full (every 256 bytes) in order to reduce the number of interrupts. This reduces the proportion of time during analogue data acquisition that the processor spends servicing interrupts. In older computers (pre-Risc PC) to ensure that there is sufficient time to service interrupts, it is necessary to select a lower band width screen mode, eg changing from mode 31 to mode 27, during analogue data acquisition.

Full software support is provided by the IIADC Module and example programs in both BASIC, C and Fortran are provided on the distribution disc.

This document is divided into two parts:-

i) a User Guide which describes how to fit the expansion card and install the software.

ii) a Programmer's Reference Manual which describes the software support provided by the IIADC Module, the IIADCLib and XIIADCLib C LIbraries and the IIADCLib Fortran Library.

Before fitting the expansion card check that the following items in addition to this document have been received. If any item is missing contact the supplier.

ADdsp Analogue to Digital Converter Expansion Card
Software Distribution Disc

Installation

1 Fitting the Expansion Card

The card can be fitted in any Acorn Computer with an expansion backplane and Risc OS 3.1 or later. To fit the card in an A300 series, A400 series, A540, A5000, Risc PC or A7000 computer:-

1 Switch off the power to the Computer.
2 Disconnect the Computer from the mains supply.
3 The card can be fitted in any unused expansion card slot.
4 Remove the blanking plate from the rear of the Computer and retain the two screws.
5 Fit the card and secure it in position using the two screws retained at stage 4 (if required, fit a joiner and blanking plate).
6 Reconnect the Computer to the mains supply.
7 Switch on the power to the Computer.
8 Confirm that the card has been fitted correctly by pressing F12 and typing

```
*Podules
```

This should list the ADdsp Analogue to Digital Converter Expansion Card as

```
DSP data acquisition system - Irlam Instruments Ltd
```

together with any other cards fitted. Press <Return> to return to the Desktop.

2 Installing the Software

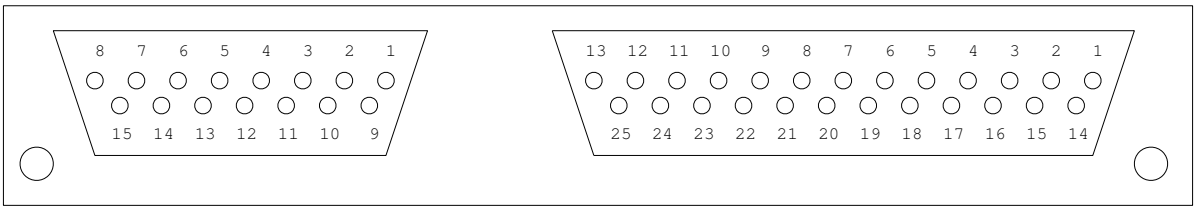
1. It is recommended that an ADC directory is created on the destination filing system (normally the hard disc of the computer).

2. Double click on IX64/arc to display the contents of the archive.

3. Select all the directories and drag them to the ADC directory of the destination filing system.

4. It is essential that the IIADC module is loaded before any application which carries out analogue to digital conversion. Opening the IIADC directory causes the !IIADC application to automatically load the IIADC module from the modules directory within the application.

3 Connector Pin Designations



Analogue inputs	Digital input and output port	
Pin 1 - Channel 1	Pin 1 - Output Port bit 0	Pin 14 - Input Port bit 0
Pin 2 - Channel 2	Pin 2 - Output Port bit 1	Pin 15 - Input Port bit 1
Pin 3 - Channel 3	Pin 3 - Output Port bit 2	Pin 16 - Input Port bit 2
Pin 4 - Channel 4	Pin 4 - Output Port bit 3	Pin 17 - Input Port bit 3
Pin 5 - Channel 5	Pin 5 - Output Port bit 4	Pin 18 - Input Port bit 4
Pin 6 - Channel 6	Pin 6 - Output Port bit 5	Pin 19 - Input Port bit 5
Pin 7 - Channel 7	Pin 7 - Output Port bit 6	Pin 20 - Input Port bit 6
Pin 8 - Channel 8	Pin 8 - Output Port bit 7	Pin 21 - Input Port bit 7
Pin 9..15 Analogue return 0V	Pin 9..13 Digital return 0V	Pin 22..25 Digital return 0V

4 Example Programs in BASIC

The programs in the directory $.BASIC on the distribution disc are examples of the use of the IIADC SWI calls. The programs ExampleE and ExampleF illustrate the use of sample buffer memory in the RMA (relocatable module area). This is essential for applications written in any language (BASIC, C or FORTRAN) which run in the WIMP environment.

5 Example Programs in C

The programs in the directory $.CLibIIADC on the distribution disc are examples of the use of the C Library IIADCLib and the programs in the directory $.CLibXIIADC on the distribution disc are examples of the use of the C Library XIIADCLib.

6 Example Programs in Fortran

The programs in the directory $.FLibIIADC on the distribution disc are examples of the use of the Fortran Library IIADCLib.

Module IIADC

IIADC SWI's

The module provides the following SWI's which are documented below:-

&048640 IIADC_OneShotConfigConvert
&048641 IIADC_OneShotConfig
&048642 IIADC_ContinuousConfig
&048643 IIADC_ConvertStart
&048644 IIADC_ConvertStop
&048645 IIADC_ConvertStatus
&048646 IIADC_ReadInPort
&048647 IIADC_WriteOutPort
&048648 IIADC_ReadWriteOptions
&048649 IIADC_ReadNumberCards
&04864A IIADC_SelChnsOneShotConfigConvert
&04864B IIADC_SelChnsOneShotConfig
&04864C IIADC_SelChnsContinuousConfig
&04864D IIADC_ClaimRMA
&04864E IIADC_ReleaseRMA
&04864F IIADC_ReadRMAInfo

IIADC_OneShotConfigConvert (SWI &048640)

Purpose
To configure the analogue to digital converter and carry out a one shot acquisition.

On entry:
R0 = card number (logical not expansion slot - 0 to 7)
R1 = sample rate index (0 to 15) see Appendix 1
R2 = number of channels (1, 2, 4 or 8)
R3 = memory address (32 bit word boundary) see Appendix 2
R4 = number of samples per channel (number of 32 bit words)
R5 = number of bits per samples
R6 = input port trigger mask (ls 8 bits only all other bits 0)
R7 = input port trigger pattern (ls 8 bits only all other bits 0)
Acquisition commences when (input port AND trigger mask) = trigger pattern
To disable trigger set R6 = 0 and R7 = 0

On exit:
R0 = preserved
R1 = preserved
R2 = preserved
R3 = preserved
R4 = preserved
R5 = preserved
R6 = preserved
R7 = preserved

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts enabled. Fast interrupts are enabled.

IIADC_OneShotConfig (SWI &048641)

Purpose
To configure the analogue to digital converter for one shot background acquisition.

On entry:
R0 = card number (logical not expansion slot - 0 to 7)
R1 = sample rate index (0 to 15) see Appendix 1
R2 = number of channels (1, 2, 4 or 8)
R3 = memory address (32 bit word boundary) see Appendix 2

R4 = number of samples per channel (number of 32 bit words)
R5 = number of bits per samples

On exit:
R0 = preserved
R1 = preserved
R2 = preserved
R3 = preserved
R4 = preserved
R5 = preserved

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts are enabled. Fast interrupts are enabled.

IIADC_ContinuousConfig (SWI &048642)

Purpose
To configure the analogue to digital converter for continuous background acquisition.

On entry:
R0 = card number (logical not expansion slot - 0 to 7)
R1 = sample rate index (0 to 15) see Appendix 1
R2 = number of channels (1, 2, 4 or 8)
R3 = memory address (32 bit word boundary) see Appendix 2
R4 = number of samples per channel (number of 32 bit words)
R5 = number of bits per samples

On exit:
R0 = preserved
R1 = preserved
R2 = preserved
R3 = preserved
R4 = preserved
R5 = preserved

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts are enabled. Fast interrupts are enabled.

IIADC_ConvertStart (SWI &048643)

Purpose
To start acquisition.

On entry:
R0 = card number
R6 = input port trigger mask (ls 8 bits only all other bits 0)
R7 = input port trigger pattern (ls 8 bits only all other bits 0)
Acquisition commences when (input port AND trigger mask) = trigger pattern
To disable trigger set R5 = 0 and R6 = 0

On exit:
R0 = preserved
R5 = preserved
R6 = preserved

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts are not disabled. Fast interrupts are enabled.

IIADC_ConvertStop (SWI &048644)

Purpose
To stop background acquisition.

On entry:
R0 = card number

On exit:
R0 = preserved

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts are not disabled. Fast interrupts are enabled.

IIADC_ConvertStatus (SWI &048645)

Purpose
To return the status of the acquisition.

On entry:
R0 = card number
R6 = input port check mask (ls 8 bits only all other bits 0)
R7 = input port check pattern (ls 8 bits only all other bits 0)
Return the status when (input port AND check mask) = check pattern
To disable check set R6 = 0 and R7 = 0

On exit:
R0 = preserved
R1 = buffer pointer - the memory address where the interrupt service routine will next store data
R2 = buffer flag - clear (0) if first half of buffer being filled
set (1) if second half of buffer being filled
R3 = error count - no errors (0)
R4 = convert flag - clear (0) if acquisition has not been started or has finished
set (1) if acquisition is in progress
R5 = byte from the input port

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts are not disabled. Fast interrupts are enabled.

IIADC_ReadInPort (SWI &048646)

Purpose
To read a byte from the input port.

On entry:
R0 = card number

On exit:
R0 = preserved
R1 = byte from the input port

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts are not disabled. Fast interrupts are enabled.

IIADC_WriteOutPort (SWI &048647)

Purpose
To write a byte to the output port.

On entry:
R0 = card number
R1 = byte

On exit:
R0 = preserved

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts are not disabled. Fast interrupts are enabled.

IIADC_ReadWriteOptions (SWI &048648)

Purpose
To read/write the card's options.

R1 = 0 to read the version number
R1 = 255 read the card swi_cnt array

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts are disabled. Fast interrupts are enabled.

IIADC_ReadWriteOptions (SWI &048648 R1 = 0)

Purpose
To read the version number of the IIADC module.

On entry:
R0 = card number
R1 = 0 to read the version number
R2 = -1

On exit:
R2 = the version number (top 16 bits = major, bottom 16 bits = minor)

IIADC_ReadWriteOptions (SWI &048648 R1 = 254)

Purpose
Read the card Interrupt count, Bytes counted with incorrect alignment and buffer pointer.

On entry:
R0 = card number
R1 = 254 read the card int_cnt and error_byte_cnt
R2 = -1 to read

On exit:
R2 = card int_cnt
R3 = card buf_ptr
R4 = card chns_status (replicated in byte3, byte2 and byte1)
R5 = card error_byte_cnt0
R6 = card error_byte_cnt1
R7 = card error_byte_cnt2
R8 = card error_byte_cnt3

IIADC_ReadWriteOptions (SWI &048648 R1 = 255)

Purpose
Read the card swi_cnt array.

On entry:
R0 = card number
R1 = 255 read the card swi_cnt array
R2 = -1 to read
R3 = swi_no

On exit:
R2 = card swi_no swi_cnt

IIADC_ReadNumberCards (SWI &048649)

No equivalent

Purpose
To read the total number of IIADC cards.

On exit:
R0 = number of IIADC cards

Interrupts
The interrupt status is unaltered from that on entry. Fast interrupts are enabled.

IIADC_SelChnsOneShotConfigConvert (SWI &04864A)

Purpose
To configure the analogue to digital converter and carry out a one shot acquisition from selected channels.

On entry:
R0 = card number (logical not expansion slot - 0 to 7)
R1 = sample rate index (0 to 15) see Appendix 1
R2 = channel status (bits 0-7 0 = discard 1 = store)
R3 = memory address (32 bit word boundary) see Appendix 2
R4 = number of samples per channel (number of 32 bit words)
R5 = number of bits per samples
R6 = input port trigger mask (ls 8 bits only all other bits 0)
R7 = input port trigger pattern (ls 8 bits only all other bits 0)
Acquisition commences when (input port AND trigger mask) = trigger pattern
To disable trigger set R6 = 0 and R7 = 0

On exit:
R0 = preserved
R1 = preserved
R2 = preserved
R3 = preserved
R4 = preserved
R5 = preserved
R6 = preserved
R7 = preserved

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts enabled. Fast interrupts are enabled.

IIADC_SelChnsOneShotConfig (SWI &04864B)

Purpose
To configure the analogue to digital converter for one shot background acquisition from selected channels.

On entry:
R0 = card number (logical not expansion slot - 0 to 7)
R1 = sample rate index (0 to 15) see Appendix 1
R2 = channel status (bits 0-7 0 = discard 1 = store)
R3 = memory address (32 bit word boundary) see Appendix 2
R4 = number of samples per channel (number of 32 bit words)
R5 = number of bits per samples

On exit:
R0 = preserved
R1 = preserved
R2 = preserved
R3 = preserved
R4 = preserved
R5 = preserved

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts are enabled. Fast interrupts are enabled.

IIADC_SelChnsContinuousConfig (SWI &04864C)

Purpose
To configure the analogue to digital converter for continuous background acquisition from selected channels.

On entry:
R0 = card number (logical not expansion slot - 0 to 7)
R1 = sample rate index (0 to 15) see Appendix 1
R2 = channel status (bits 0-7 0 = discard 1 = store)
R3 = memory address (32 bit word boundary) see Appendix 2
R4 = number of samples per channel (number of 32 bit words)
R5 = number of bits per samples

On exit:
R0 = preserved
R1 = preserved
R2 = preserved
R3 = preserved
R4 = preserved
R5 = preserved

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts are enabled. Fast interrupts are enabled.

IIADC_ClaimRMA (SWI &04864D)

Purpose
To claim sample buffer memory from the RMA.

On entry:
R0 = card number (logical not expansion slot - 0 to 7)
R2 = number of channels (1, 2, 4 or 8)
R4 = number of samples per channel (number of 32 bit words)

On exit:
R0 = preserved
R1 = preserved
R2 = preserved
R3 = memory address (32 bit word boundary) see Appendix 2
R4 = preserved
R5 = preserved

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts are enabled. Fast interrupts are enabled.

IIADC_ReleaseRMA (SWI &04864E)

Purpose
To release sample buffer memory from the RMA.

On entry:
R0 = card number (logical not expansion slot - 0 to 7)

On exit:
R0 = preserved

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts are enabled. Fast

interrupts are enabled.

IIADC_ReadRMAInfo (SWI &04864F)

Purpose
To read information on the sample buffer memory claimed from the RMA

On entry:
R0 = card number (logical not expansion slot - 0 to 7)

On exit:
R0 = preserved
R1 = preserved
R2 = number of channels (1, 2, 4 or 8)
R3 = memory address (32 bit word boundary) Returns 0 if no memory claimed from the RMA
see Appendix 2
R4 = number of samples per channel (number of 32 bit words)

Interrupts
The interrupt status on exit is restored to that on entry. During the SWI interrupts are enabled. Fast interrupts are enabled.

The llADCLib C Library

void llADC_OneShotConfigConvert(int card, int sample_rate_index, int no_channels, int *mem_ptr, int no_samples_per_channel)

Purpose
To configure the analogue to digital converter and carry out a one shot acquisition.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int no_channels = number of channels (1, 2, 4 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)

void llADC_OneShotConfig(int card, int sample_rate_index, int no_channels, int *mem_ptr, int no_samples_per_channel)

Purpose
To configure the analogue to digital converter for one shot background acquisition.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int no_channels = number of channels (1, 2, 4 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)

void llADC_ContinuousConfig(int card, int sample_rate_index, int no_channels, int *mem_ptr, int no_samples_per_channel)

Purpose
To configure the analogue to digital converter for continuous background acquisition.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int no_channels = number of channels (1, 2, 4 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)

void llADC_ConvertStart(int card, int trigger_mask, int trigger_pattern)

Purpose
To start acquisition when (input port AND trigger mask) = trigger pattern.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int trigger_mask = input port trigger mask
int trigger_pattern = input port trigger pattern

void llADC_ConvertStop(int card)

Purpose
To stop acquisition.

On entry:
int card = card number (logical not expansion slot - 0 to 7)

void llADC_ConvertStatus(int card, void *buf_ptr, int *buf_flag, int *error_cnt, int *convert_flag)

Purpose
To return the status of the acquisition.

On entry:
int card = card number (logical not expansion slot - 0 to 7)

On exit:

void *buf_ptr = address of buffer pointer -	where the interrupt service routine will next store data
int *buf_flag = address of buffer flag -	clear (0) if first half of buffer being filled set (1) if second half of buffer being filled
int *error_cnt = address of error count -	no errors (0)
int *convert_flag = address of convert flag -	clear (0) if acquisition has not been started or has finished set (1) if acquisition is in progress

void llADC_ReadInPortConvertStatus(int card, int check_mask, int check_pattern, void *buf_ptr, int *buf_flag, int *error_cnt, int *convert_flag, int *byte_ptr)

Purpose
To return the status of the acquisition when (input port AND check mask) = check pattern.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int check_mask = input port check mask
int check_pattern = input port check pattern

On exit:

void *buf_ptr = address of buffer pointer -	where the interrupt service routine will next store data
int *buf_flag = address of buffer flag -	clear (0) if first half of buffer being filled set (1) if second half of buffer being filled
int *error_cnt = address of error count -	no errors (0)
int *convert_flag = address of convert flag -	clear (0) if acquisition has not been started or has finished set (1) if acquisition is in progress

int *byte_ptr = address of byte from the input port

int llADC_ReadInPort(int card)

Purpose
To read a byte from the input port.

On entry:
int card = card number (logical not expansion slot - 0 to 7)

On exit:
returns a byte from the input port

void llADC_WriteOutPort(int card, int byte)

Purpose
To write a byte to the output port.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int byte = byte

void llADC_Diagnostic(int card, intint_cnt, int *buf_ptr, int *error_byte_cnt0, int *error_byte_cnt1, int *error_byte_cnt2, int *error_byte_cnt3)

Purpose
Reserved for use by Intelligent Interfaces.

void llADC_SelChnsOneShotConfigConvert(int card, int sample_rate_index, int chns_status, int *mem_ptr, int no_samples_per_channel)

Purpose
To configure the analogue to digital converter and carry out a one shot acquisition from selected channels.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int chns_status = channel status (bits 0-7 0 = discard 1 = store)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)

void llADC_SelChnsOneShotConfig(int card, int sample_rate_index, int chns_status, int *mem_ptr, int no_samples_per_channel)

Purpose
To configure the analogue to digital converter for one shot background acquisition from selected channels.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int chns_status = channel status (bits 0-7 0 = discard 1 = store)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)

void llADC_SelChnsContinuousConfig(int card, int sample_rate_index, int chns_status, int *mem_ptr, int no_samples_per_channel)

Purpose
To configure the analogue to digital converter for continuous background acquisition from selected channels.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int chns_status = channel status (bits 0-7 0 = discard 1 = store)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)

void llADC_SingleChnOneShotConfigConvert(int card, int sample_rate_index, int channel, int *mem_ptr, int no_samples_per_channel)

Purpose
To configure the analogue to digital converter and carry out a one shot acquisition from a single channel.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int channel = number of channel (1, 2, 3, 4, 5, 6, 7 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2

int no_samples_per_channel = number of samples per channel (number of 32 bit words)

void llADC_SingleChnOneShotConfig(int card, int sample_rate_index, int channel, int *mem_ptr, int no_samples_per_channel)

Purpose

To configure the analogue to digital converter for one shot background acquisition from a single channel.

On entry:

int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int channel = number of channel (1, 2, 3, 4, 5, 6, 7 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)

void llADC_SingleChnContinuousConfig(int card, int sample_rate_index, int channel, int *mem_ptr, int no_samples_per_channel)

Purpose

To configure the analogue to digital converter for continuous background acquisition from a single channel.

On entry:

int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int channel = number of channel (1, 2, 3, 4, 5, 6, 7 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)

void llADC_Mean(int no_channels, int *mem_ptr, int no_samples_per_channel, int *mean_ptr)

Purpose

To calculate the mean of the values placed in memory by an acquisition.

On entry:

int no_channels = number of channels (1, 2, 4 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)
int *mean_ptr = memory address (32 bit word boundary)

The mean of the values (32 bit integers in 2's complement format) are placed in memory in ascending channel order starting at this address. The mean of the values with 12 significant bits range from 0xFFFFF800 (-5V) to 0x000007FF (+5V)

The XIIADCLib C Library

_XIIADC_error *XIIADC_OneShotConfigConvert(int card, int sample_rate_index,int no_channels, int *mem_ptr, int no_samples_per_channel, int no_bits_per_sample)

Purpose
To configure the analogue to digital converter and carry out a one shot acquisition.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int no_channels = number of channels (1, 2, 4 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)
int no_bits_per_sample = number of bits converted 8 or 12

_XIIADC_error *XIIADC_OneShotConfig(int card, int sample_rate_index, int no_channels, int *mem_ptr, int no_samples_per_channel, int no_bits_per_sample)

Purpose
To configure the analogue to digital converter for one shot background acquisition.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int no_channels = number of channels (1, 2, 4 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)
int no_bits_per_sample = number of bits converted 8 or 12

_XIIADC_error *XIIADC_ContinuousConfig(int card, int sample_rate_index, int no_channels, int *mem_ptr, int no_samples_per_channel, int no_bits_per_sample)

Purpose
To configure the analogue to digital converter for continuous background acquisition.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int no_channels = number of channels (1, 2, 4 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)
int no_bits_per_sample = number of bits converted 8 or 12

_XIIADC_error *XIIADC_ConvertStart(int card, int trigger_mask, int trigger_pattern)

Purpose
To start acquisition.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int trigger_mask = input port trigger mask
int trigger_pattern = input port trigger pattern

_XIIADC_error *XIIADC_ConvertStop(int card)

Purpose
To stop acquisition.

On entry:
int card = card number (logical not expansion slot - 0 to 7)

_XIIADC_error *XIIADC_ConvertStatus(int card, void *buf_ptr, int *buf_flag, int *error_cnt, int *convert_flag)

Purpose
To return the status of the acquisition.

On entry:
int card = card number (logical not expansion slot - 0 to 7)

On exit:

void *buf_ptr = address of buffer pointer -	where the interrupt service routine will next store data
int *buf_flag = address of buffer flag -	clear (0) if first half of buffer being filled set (1) if second half of buffer being filled
int *error_cnt = address of error count -	no errors (0)
int *convert_flag = address of convert flag -	clear (0) if acquisition has not been started or has finished set (1) if acquisition is in progress

void IIADC_ReadInPortConvertStatus(int card, int check_mask, int check_pattern, void *buf_ptr, int *buf_flag, int *error_cnt, int *convert_flag, int *byte_ptr)

Purpose
To return the status of the acquisition when (input port AND check mask) = check pattern.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int check_mask = input port check mask
int check_pattern = input port check pattern

On exit:

void *buf_ptr = address of buffer pointer -	where the interrupt service routine will next store data
int *buf_flag = address of buffer flag -	clear (0) if first half of buffer being filled set (1) if second half of buffer being filled
int *error_cnt = address of error count -	no errors (0)
int *convert_flag = address of convert flag -	clear (0) if acquisition has not been started or has finished set (1) if acquisition is in progress

int *byte_ptr = address of byte from the input port

int XIIADC_ReadInPort(int card)

Purpose
To read a byte from the input port.

On entry:
int card = card number (logical not expansion slot - 0 to 7)

On exit:
returns a byte from the input port

void XIIADC_WriteOutPort(int card, int byte)

Purpose
To write a byte to the output port.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int byte = byte

_XIIADC_error *XIIADC_Diagnostic(int card, int *int_cnt, int *buf_ptr, int *error_byte_cnt0, int *error_byte_cnt1, int *error_byte_cnt2, int *error_byte_cnt3)

Purpose

Reserved for use by Intelligent Interfaces.

_XIIADC_error *XIIADC_SelChnsOneShotConfigConvert(int card, int sample_rate_index, int chns_status, int *mem_ptr, int no_samples_per_channel, int no_bits_per_sample)

Purpose

To configure the analogue to digital converter and carry out a one shot acquisition from selected channels.

On entry:

int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int chns_status = channel status (bits 0-7 0 = discard 1 = store)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)
int no_bits_per_sample = number of bits converted 8 or 12

_XIIADC_error *XIIADC_SelChnsOneShotConfig(int card, int sample_rate_index, int chns_status, int *mem_ptr, int no_samples_per_channel, int no_bits_per_sample)

Purpose

To configure the analogue to digital converter for one shot background acquisition from selected channels.

On entry:

int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int chns_status = channel status (bits 0-7 0 = discard 1 = store)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)
int no_bits_per_sample = number of bits converted 8 or 12

_XIIADC_error *XIIADC_SelChnsContinuousConfig(int card, int sample_rate_index, int chns_status, int *mem_ptr, int no_samples_per_channel, int no_bits_per_sample)

Purpose

To configure the analogue to digital converter for continuous background acquisition from selected channels.

On entry:

int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int chns_status = channel status (bits 0-7 0 = discard 1 = store)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)
int no_bits_per_sample = number of bits converted 8 or 12

_XIIADC_error *XIIADC_SingleChnOneShotConfigConvert(int card, int sample_rate_index, int channel, int *mem_ptr, int no_samples_per_channel, int no_bits_per_sample)

Purpose

To configure the analogue to digital converter and carry out a one shot acquisition from a single channel.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int channel = number of channel (1, 2, 3, 4, 5, 6, 7 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)
int no_bits_per_sample = number of bits converted 8 or 12

_XIIADC_errorXIIADC_SingleChnOneShotConfig(int card, int sample_rate_index, int channel, int *mem_ptr, int no_samples_per_channel, int no_bits_per_sample)

Purpose
To configure the analogue to digital converter for one shot background acquisition from a single channel.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int channel = number of channel (1, 2, 3, 4, 5, 6, 7 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)

_XIIADC_error *XIIADC_SingleChnContinuousConfig(int card, int sample_rate_index, int channel, int *mem_ptr, int no_samples_per_channel)

Purpose
To configure the analogue to digital converter for continuous background acquisition from a single channel.

On entry:
int card = card number (logical not expansion slot - 0 to 7)
int sample_rate_index = sample rate index (0 to 15) see Appendix 1
int channel = number of channel (1, 2, 3, 4, 5, 6, 7 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)
int no_bits_per_sample = number of bits converted 8 or 12

void XIIADC_Mean(int no_channels, int *mem_ptr, int no_samples_per_channel, int *mean_ptr)

Purpose
To calculate the mean of the values placed in memory by an acquisition.

On entry:
int no_channels = number of channels (1, 2, 4 or 8)
int *mem_ptr = memory address (32 bit word boundary) see Appendix 2
int no_samples_per_channel = number of samples per channel (number of 32 bit words)
int *mean_ptr = memory address (32 bit word boundary)
The mean of the values (32 bit integers in 2's complement format) are placed in memory in ascending channel order starting at this address. The mean of the values with 12 significant bits range from 0xFFFFF800 (-5V) to 0x000007FF (+5V)

The llADCLib Fortran Library

SUBROUTINE llADCOneShotConfigConvert(icard, isamplerateindex, inochannels,idataarray,inosamplesperchannel)

Purpose
To configure the analogue converter and carry out a one shot acquisition.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)
isamplerateindex (integer) = sample rate index (0 to 15) see Appendix 1
inochannels(integer) = number of channels (1, 2, 4 or 8)
idataarray (integer) see Appendix 2
inosamplesperchannel (integer) = number of samples per channel (number of 32 bit words)

SUBROUTINE llADCOneShotConfig(icard, isamplerateindex, inochannels,idataarray,inosamplesperchannel)

Purpose
To configure the analogue to digital converter for a one shot background acquisition.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)
isamplerateindex (integer) = sample rate index (0 to 15) see Appendix 1
inochannels(integer) = number of channels (1, 2, 4 or 8)
idataarray (integer) see Appendix 2
inosamplesperchannel (integer) = number of samples per channel (number of 32 bit words)

SUBROUTINE llADCContinuousConfig(icard, isamplerateindex, inochannels,idataarray,inosamplesperchannel)

Purpose
To configure the analogue to digital converter for continuous background acquisition.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)
isamplerateindex (integer) = sample rate index (0 to 15) see Appendix 1
inochannels(integer) = number of channels (1, 2, 4 or 8)
idataarray (integer) see Appendix 2
inosamplesperchannel (integer) = number of samples per channel (number of 32 bit words)

SUBROUTINE llADCConvertStart(icard, itriggermask, itriggerpattern)

Purpose
To start acquisition.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)
itriggermask (integer) = input port trigger mask
itriggerpattern (integer) = input port trigger pattern

SUBROUTINE llADCConvertStop(icard)

Purpose
To stop acquisition.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)

SUBROUTINE llADCConvertStatus(icard, ibufptr, ibufflag, ierrorcnt, iconvertflag)

Purpose
To return the status of the acquisition.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)

On exit:

ibufptr (integer) = buffer pointer -	where the interrupt service routine will next store data
ibufflag (integer) = buffer flag -	clear (0) if first half of buffer being filled
	set (1) if second half of buffer being filled
ierrorcnt (integer) = error count -	no errors (0)
iconvertflag (integer) = convert flag -	clear (0) if acquisition has not been started or has finished
	set (1) if acquisition is in progress

SUBROUTINE IIADCReadInPortConvertStatus(icard, ibufptr, ibufflag, ierrorcnt, iconvertflag, ibyte)

Purpose
To return the status of the acquisition.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)

On exit:

ibufptr (integer) = buffer pointer -	where the interrupt service routine will next store data
ibufflag (integer) = buffer flag -	clear (0) if first half of buffer being filled
	set (1) if second half of buffer being filled
ierrorcnt (integer) = error count -	no errors (0)
iconvertflag (integer) = convert flag -	clear (0) if acquisition has not been started or has finished
	set (1) if acquisition is in progress

ibyte (integer) = byte from the input port

SUBROUTINE IIADCReadInPort(icard, ibyte)

Purpose
To read a byte from the input port.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)

On exit:
ibyte (integer) = byte from the input port

SUBROUTINE IIADCWriteOutPort(icard, ibyte)

Purpose
To write a byte to the output port.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)
ibyte (integer) = byte

SUBROUTINE IIADCDiagnostic(icard, intcnt, ibufptr, ierrorbytecnt0, ierrorbytecnt1, ierrorbytecnt2, ierrorbytecnt3)

Purpose
Reserved for use by Intelligent Interfaces.

SUBROUTINE IIADCSelChnsOneShotConfigConvert(icard, isamplerateindex, ichnsstatus,idataarray,inosamplesperchannel)

Purpose
To configure the analogue to digital converter and carry out a one shot acquisition from selected channels.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)

isamplerateindex (integer) = sample rate index (0 to 15) see Appendix 1
ichnsstatus (integer) = channel status (bits 0-7 0 = discard 1 = store)
idataarray (integer) see Appendix 2
inosamplesperchannel (integer) = number of samples per channel (number of 32 bit words)

SUBROUTINE IIADCSelChnsOneShotConfig(icard, isamplerateindex, ichnsstatus,idataarray,inosamplesperchannel)

Purpose
To configure the analogue to digital converter for a one shot background acquisition from selected channels.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)
isamplerateindex (integer) = sample rate index (0 to 15) see Appendix 1
ichnsstatus (integer) = channel status (bits 0-7 0 = discard 1 = store)
idataarray (integer) see Appendix 2
inosamplesperchannel (integer) = number of samples per channel (number of 32 bit words)

SUBROUTINE IIADCSelChnsContinuousConfig(icard, isamplerateindex, ichnsstatus,idataarray,inosamplesperchannel)

Purpose
To configure the analogue to digital converter for a continuous background acquisition from selected channels.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)
isamplerateindex (integer) = sample rate index (0 to 15) see Appendix 1
ichnsstatus (integer) = channel status (bits 0-7 0 = discard 1 = store)
idataarray (integer) see Appendix 2
inosamplesperchannel (integer) = number of samples per channel (number of 32 bit words)

SUBROUTINE IIADCSingleChnOneShotConfigConvert(icard, isamplerateindex, ichannel,idataarray,inosamplesperchannel)

Purpose
To configure the analogue to digital converter and carry out a one shot acquisition from a single channel.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)
isamplerateindex (integer) = sample rate index (0 to 15) see Appendix 1
ichannel (integer) = number of channel (1, 2, 3, 4, 5, 6, 7 or 8)
idataarray (integer) see Appendix 2
inosamplesperchannel (integer) = number of samples per channel (number of 32 bit words)

SUBROUTINE IIADCSingleChnOneShotConfig(icard, isamplerateindex, ichannel,idataarray,inosamplesperchannel)

Purpose
To configure the analogue to digital converter for a one shot background acquisition from a single channel.

On entry:
icard (integer) = card number (logical not expansion slot - 0 to 7)
isamplerateindex (integer) = sample rate index (0 to 15) see Appendix 1
ichannel (integer) = number of channel (1, 2, 3, 4, 5, 6, 7 or 8)
idataarray (integer) see Appendix 2
inosamplesperchannel (integer) = number of samples per channel (number of 32 bit words)

SUBROUTINE IIADCSingleChnContinuousConfig(icard, isamplerateindex, ichannel,idataarray,inosamplesperchannel)

Purpose

To configure the analogue to digital converter for a continuous background acquisition from a single channel.

On entry:

icard (integer) = card number (logical not expansion slot - 0 to 7)
isamplerateindex (integer) = sample rate index (0 to 15) see Appendix 1
ichannel (integer) = number of channel (1, 2, 3, 4, 5, 6, 7 or 8)
idataarray (integer) see Appendix 2
inosamplesperchannel (integer) = number of samples per channel (number of 32 bit words)

SUBROUTINE IIADCMean(inochannels,idataarray,inosamplesperchannel, imeanarray)

Purpose

To calculate the mean of the values placed in memory by an acquisition.

On entry:

inochannels (integer) = number of channels (1, 2, 4 or 8)
idataarray (integer) see Appendix 2
inosamplesperchannel (integer) = number of samples per channel (number of 32 bit words)
imeanarray (integer) =

The mean of the values (32 bit integers in 2's complement format) are placed in imeanarray in ascending channel order. The mean of the values with 12 significant bits range from 0xFFFFF800 (-5V) to 0x000007FF (+5V)

Appendix 1 - Sample Rate Index

Increasing the number of channels decreases the sample rate per channel because the inputs are multiplexed into one sample and hold amplifier and analogue to digital converter.

Frequency (approximate) Table

Index	Divisor	Channels			
		1	2	4	8
0	1	333.00kHz	166.50kHz	83.25kHz	41.63kHz
1	2	166.50kHz	83.25kHz	41.63kHz	20.81kHZ
2	4	83.25kHz	41.63kHz	20.81kHZ	10.41kHz
3	10	33.30kHz	16.65kHz	8.33kHz	4.16kHz
4	20	16.65kHz	8.33kHz	4.16kHz	2.08kHz
5	40	8.33kHz	4.16kHz	2.08kHz	1.04kHz
6	100	3330.00Hz	1665.00Hz	832.50Hz	416.25Hz
7	200	1665.00Hz	832.50Hz	416.25Hz	208.13Hz
8	400	832.50Hz	416.25Hz	208.13Hz	104.07Hz
9	1000	333.00Hz	166.50Hz	83.25Hz	41.63Hz
10	2000	166.50Hz	83.25Hz	41.63Hz	20.81Hz
11	4000	83.25Hz	41.63Hz	20.81Hz	10.41Hz
12	10000	33.30Hz	16.65Hz	8.33Hz	4.16Hz
13	20000	16.65Hz	8.33Hz	4.16Hz	2.08Hz
14	40000	8.33Hz	4.16Hz	2.08Hz	1.04Hz
15	100000	3.33Hz	1.67Hz	0.83Hz	0.42Hz

Period (approximate) Between Successive Samples from the Same Channel Table

Index	Divisor	Channels			
		1	2	4	8
0	1	3us	6us	12us	24us
1	2	6us	12us	24us	48us
2	4	12us	24us	48us	96us
3	10	30us	60us	120us	240us
4	20	60us	120us	240us	480us
5	40	120us	240us	480us	960us
6	100	0.3ms	0.6ms	1.2ms	2.4ms
7	200	0.6ms	1.2ms	2.4ms	4.8ms
8	400	1.2ms	2.4ms	4.8ms	9.6ms
9	1000	3ms	6ms	12ms	24ms
10	2000	6ms	12ms	24ms	48ms
11	4000	12ms	24ms	48ms	96ms
12	10000	30ms	60ms	120ms	240ms
13	20000	60ms	120ms	240ms	480ms
14	40000	120ms	240ms	480ms	960ms
15	100000	300ms	600ms	1200ms	2400ms

Appendix 2 - Sample Data Format in Memory

The values (32 bit integers in 2's complement format) are placed in memory in ascending channel order.

Irrespective of the number of bits per sample the values range from 0xFFFFF800 (-5V) to 0x0000007FF (+5V).

For 8 bits per sample, the 8 bits of the sample are placed in bits 4-11 of the 32 bit word and the least significant four bits 0-3 of the sample are repeated in the bits 0-3 of the 32 bit word. The sign bit (bit 7 of the sample) is replicated in bits 12-31 of the 32 bit word.

For 12 bits per sample, the 12 bits of the sample are placed in bits 0-11 of the 32 bit word. The sign bit (bit 11 of the sample) is replicated in bits 12-31 of the 32 bit word.

Appendix 3 - Multi-channel Acquisition

The period between adjacent channels (t) is always 3us.

The following example illustrates multi-channel acquisition.

To sample from 4 channels at a sample rate index of 8, the table in Appendix 1 gives the sample rate as 208.13Hz and the period (the time between successive samples from the same channel) as T =4.8ms.