

DM6420 Data Acquisition Board Driver for Windows 95/98/NT

User's Manual

Real Time Devices USA, Inc.

Table of Contents

INTRODUCTION	1
INSTALLING OF THE DRIVER AND EXAMPLE PROGRAMS.....	3
THE DM6420 BOARD DRIVER	4
THE DRIVER API FUNCTIONS.....	5
USING THE DRIVER WITHOUTH THE WINDOWS-GUI.....	6
DM6420 FEATURES	7
MEASUREMENT SCENARIOS.....	7
CHANNEL-GAIN CIRCUITRY	7
INTERRUPTS.....	10
DIRECT MEMORY ACCESS (DMA)	12
DRIVER API FUNCTION GROUPS.....	13
DRIVER INITIALIZATION FUNCTIONS	13
GENERAL BOARD CONTROL FUNCTIONS	13
FIFO MANIPULATION.....	13
DMA FUNCTIONS.....	13
A/D CONVERTER.....	13
D/A CONVERTER.....	14
PACER CLOCK HANDLING	14
BURST CLOCK HANDLING	14
ABOUT COUNTER FUNCTIONS	14
SAMPLE COUNTER FUNCTIONS	14
DIGITAL I/O FUNCTIONS.....	14
INTERRUPT HANDLING	14
AUTOINCREMENT DATA HANDLING	15
USER TIMER-COUNTER FUNCTIONS.....	15
CHANNEL-GAIN TABLE/LATCH MANIPULATION	15
EXTERNAL TRIGGER CONFIGURATION FUNCTIONS.....	15
ERROR HANDLING FUNCTIONS	15
ALPHABETICAL DRIVER API FUNCTIONS REFERENCE.....	17
EXAMPLE PROGRAMS REFERENCE.....	57
WIN32 CONSOLE APPLICATIONS	57
WIN32 WINDOWS APPLICATIONS	58

Introduction

The DM6420 data acquisition board Windows 95/98/NT driver has designed for programmers who write Windows-based application programs with using the RTD's DM6420 board.

The driver provides an Application Programming Interface with a lot of function calls to perform all the data acquisition tasks of the board users.

The board driver based on the BlueWater System's WinRT device driver kit.

There are example programs to demonstrate the various board features and the usage of the driver API. The example programs are written in Microsoft Visual C++ ver. 5.0.

Installing of the Driver and Example Programs

Before installing the driver and example program files, you need to install the DM6420 board in your PC. Please follow the instructions of the manufacturer, how to install the board in a computer.

On the installation diskette 1 you can find the **Setup.exe** program, which installs on your PC the DM6420 board driver and example programs. The setup program automatically detects your operating system (Windows 95/98 or NT) and installs the appropriate files on your PC.

After starting the setup, please follow the instructions on the screen to install the programs. You can select the directory where to install the files.

The setup also adds to the 'Start menu' under the 'Programs' folder of your Windows system the 'Real Time Devices USA' folder. It contains shortcuts to the example programs, the driver configuration utility and the readme.txt file.

The example programs will be installed in the target directory under the 'Examples' folder. This folder contains the example program sources for the DM6420 board, and the project files for Microsoft Visual C++ users to rebuild the programs. The example programs are compiled with the Microsoft Visual C++ ver. 5.0. In case of different version of Visual C is installed on your PC, please rebuild the executable files.

Uninstallation: you can uninstall the DM6420 driver and example programs under the 'Control panel' with the 'Add/Remove Programs' tool.

The DM6420 board driver

The RTD's DM6420 board driver handles the hardware through the BlueWater's WinRT driver. The WinRT driver provides the low-level access to the board, and the RTD's Drvr6420.dll provides the device API for the programmers, and communicates with the hardware through the WinRT driver.

The RTD board drivers has multi-board feature. It means that it is possible to use more than one board in the system.

The base address of the DM6420 board can be set by DIP switch. If you are using multiple boards in your system, you need assign a different base address to each board. The IRQ and DMA channels are set programmatically. When you install a WinRT driver for the board, you must select the IRQ and DMA channel, and during the operating of the board must be used this IRQ and DMA channels.

Each board requires a unique WinRT device to handle the hardware. Also the second DMA channel of the DM6420 board requires a unique device. The maximum number of devices on Windows NT is 32, on Windows 95/98 is 10.

During the installation the Setup.exe program set the default driver values:

Board 1 : 0x300 base address, DMA channel 5, DMA buffer size 0x4000 and IRQ 11

Board 1 2'nd DMA channel: DMA channel 6, DMA buffer size 0x4000.

If this settings are not proper to your PC configuration, you can modify the settings with the **Drvr_cfg.exe** program. With this program you can also add new WinRT devices when you are using multiple boards.

The Driver Api Functions

The resources on the DM6420 board can be accessed from Windows through the driver API (Application Programming Interface) functions. The executable code of these functions is located in the Drvr6420.dll file. To write applications using the API functions you must include the Reg6420.H header file, and link the program with the Drvr6420.lib import library file. In the example programs you can find different examples how to use the driver.

Using the Driver without the Windows-GUI

This driver is based on the Win32 system, and can run only under Windows. However, the user, who is not familiar with the Windows Graphical User Interface, can use the driver API functions in Win32 console applications, which has an MS-DOS like text-mode interface.

In console applications is not necessary to use the Windows graphical environment, but all the driver's API functions are accessible.

The Microsoft Visual C++ compiler supports writing console applications.

DM6420 Features

Measurement Scenarios

Through selecting different options for A/D Conversion Trigger (SetConversionSelect6420), Burst Clock Start Trigger (SetBurstTrigger6420), and creating different Channel-Gain Tables, you have innumerable sampling scenarios. The following bullets try to enumerate only the most frequently used measurement setups.

❑ ***Single Conversion***

In this mode, a single channel is sampled whenever StartConversion6420 is called. The Channel Gain Latch (see SetChannelGain6420) specifies the channel to sample. This is the easiest scenario of all. It can be used in a variety of applications, such as sample every time a key is pressed on the keyboard, sample with each iteration of a loop, or watch the system clock and sample every five seconds.

❑ ***Multiple Conversions***

In this mode, conversions are continuously performed at the rate of the Pacer Clock, or other selected A/D Conversion Signal rate. The pacer clock can be internal or external. The maximum rate supported by the board is 500KHz. If you use the internal pacer clock, you must program it to run at the desired rate (SetPacerClock6420).

This mode is ideal for filling arrays, acquiring data for a specified period, and taking a specified number of samples.

❑ ***Random Channel Scan***

In this mode, the Channel-Gain Table is incrementally scanned through, with each selected A/D Conversion Signal pulse starting a conversion at the channel and gain specified in the current table entry. Before starting a conversion sequence Channel Gain Table, you need to load the table with the desired data. Then make sure that the Channel-Gain Table is enabled by the function EnableTables6420. This enables the A/D an Digital portion of the Channel Gain Table as well. Each rising edge of selected A/D Conversion Signal starts a conversion using the current Channel Gain data and then increments to the next position in the table. When the last entry is reached, the next pulse starts the table over again.

❑ ***Programmable Burst***

In this mode, a single trigger initiates a scan of the entire Channel-Gain Table. Before starting a burst of the Channel-Gain Table, you need to load the table with the desired data. Then enable the Channel-Gain Table by EnableTable6420.

Burst is used when you want one sample from a specified number of channels for each trigger. The burst trigger starts the Burst Clock and the Burst Clock initiates each conversion. At high speeds, the burst mode emulates simultaneous sampling of multiple input channels. For time critical simultaneous sampling applications, a simultaneous sample-and-hold board can be used (SS8 eight-channel boards are available from Real Time Devices).

❑ ***Programmable Multi-Scan***

This mode - when the A/D Conversion Start Signal is the Burst Clock - lets you scan the Channel Gain Table after a Burst Clock Start Signal. When the Channel Gain Table is empty, the Burst Clock is stopped, and will wait for a new Start Signal.

Channel-Gain Circuitry

Channel-Gain Tables are traditionally for implementing random channel scan analog input on boards where a single A/D converter is multiplexed for 8, 16 or more analog input channels.

The Channel-Gain Circuitry embeds a 1024x24 bit on-board memory (Channel Scan Memory), called *Channel-Gain Table* (CGT) for historical reasons. Every 24-bit row (entry) in a CGT is an instruction executed by the Channel-Gain circuitry. Execution happens at a programmable rate. Channel-Gain Latch (CGL), provided for easy, single channel analog input, can be perceived as a special, single row CGT for the following description. Unless explicitly indicated, explanation holds for the CGL, as well.

The table below pictures the format of a CGT entry:

DO	Skip	Pause	Se/Diff	Range	Gain	Channel
8 bits	1 bit	1 bit	1 bit	2 bits	3 bits	4 bits

Channel Analog Input Channel

Specifies the Analog Input channel to sample. Depending on your configuration you may have 8 differential channels (AIN1...AIN8), or 16 Single-Ended channels (AIN1...AIN16)

Gain Analog Input Gain

This field specifies the gain to apply to the input. Available choices are 1x, 2x, 4x, 8x.

Range Analog Input Range

Specifies one of the three supported input ranges: ± 5 Volts, ± 10 Volts or 0...10Volts.

Se/Diff Analog Input Type

Pause Pause Bit

If this bit is enabled by the SetADPauseEnable6420 function, execution of the Channel-Gain Table stops *after* executing this entry. Execution is resumed with the next CGT entry when the programmed Pacer Clock start trigger occurs.

EXAMPLE: Pause Bit can be used when you have two sequence of entries, each to be executed on a different event (trigger). Suppose that CGT is driven by the Pacer Clock, and the Pacer Clock is started on the External Trigger. The External Trigger comes from a device, whose pulses indicate two different events. Odd pulses indicate an event, on which you want to react by sampling AIN1 and AIN2, on even pulses you want to sample AIN3, AIN4 and AIN5. In this case, you would create a 5 entry CGT:

- Entry #1: AIN1, Pause Bit = 0*
- Entry #2: AIN2, Pause Bit = 1*
- Entry #3: AIN3, Pause Bit = 0*
- Entry #4: AIN4, Pause Bit = 0*
- Entry #5: AIN5, Pause Bit = 1*

In this case, the first pulse on the External Trigger line starts executing the CGT at the rate of the Pacer Clock. After executing the first two entries, execution stops and is waiting for the next External Trigger pulse. The second pulse resumes execution, and entries #3, #4 and #5 are executed at the rate of the Pacer Clock. Execution pauses again, after executing entry #5. A third External trigger pulse continues execution with entry #1, and so on.

NOTE: When the Channel-Gain Latch is used, or in burst mode, Pause Bit is ignored.

Skip Skip Bit

When the Skip Bit is set, the entry is skipped, which means that the A/D conversion is performed but the resulting sample is not written into the A/D FIFO. This feature provides a way to sample multiple channels at different rates without saving unwanted data.

EXAMPLE: In this example, we want to sample AIN1 in every second and AIN4 in every three seconds. For this end, we must create CGT with six entries:

Entry #1: AIN1, Skip Bit = 0
Entry #2: AIN4, Skip Bit = 1
Entry #3: AIN1, Skip Bit = 0
Entry #4: AIN4, Skip Bit = 1
Entry #5: AIN1, Skip Bit = 0
Entry #6: AIN4, Skip Bit = 0

Next, we set the Pacer Clock to run at 2 Hz (0.5 seconds). This allows us to sample each channel once per second, the maximum sampling rate required by one of the channels (pacer clock rate = number of different channels sampled x fastest sample rate).

The first Pacer Clock pulse starts an A/D conversion according to the parameters set in the first entry of the Channel-Gain Table, and each successive clock pulse incrementally steps through the table entries. The first clock pulse takes a sample on AIN1. The second pulse looks at the second entry in the table and sees that the Skip Bit is set. Sample is taken, but is not stored in the FIFO. The third pulse takes a sample on AIN1 again, the fourth pulse skips the next entry, and the fifth pulse takes our third reading on AIN1. On the sixth pulse, the Skip Bit is disabled, AIN4 is sampled and sample is stored to the FIFO. Then the sequence starts over again with entry #1. Samples are not stored when they are not wanted, saving memory and eliminating the need to throw away unwanted data.

NOTE: When the Channel-Gain Latch is used, Skip Bit is ignored.

DO 8-Bit Digital Table

The digital portion of the Channel-Gain Table, also referred to as Digital Table, can be used to control input expansion boards such as the TMX32 Analog Input Expansion board. The expansion board is driven at the same speed as the A/D conversions are performed, with no software overhead.

EXAMPLE: Let us consider the following simple example on driving an analog input expansion board.

In this example, we have a TMX32 expansion board connected to AIN1 on the DM6420. We have three signals to sample, one is connected to the first channel of the expansion board (EAIN1), the second is connected to the fourth channel of the expansion board (EAIN4) and the third is connected directly to AIN2 of the DM6420.

We need to create the following Channel-Gain Table:

Entry #1: AIN1, gain=1, DO=0
Entry #2: AIN1, gain=4, DO=3
Entry #3: AIN2, gain=1, DO=3

Execution, starting with entry #1, samples AIN1 and simultaneously outputs 0 on Digital Port 1. This will cause the expansion board to switch to EAIN1.

Entry #2 will sample AIN1, which is now connected to EAIN1, and simultaneously outputs 3 on Digital Port 1. As a result, the expansion board switches to EAIN4.

Next, entry #3 samples AIN2 and outputs 3 on Digital Port 1, which makes the expansion board to switch (again) to EAIN4.

When executing entry #1 again, AIN1 is sampled which is now connected to EAIN4, and so on.

NOTE: If you only need to use the A/D part of the table, you do not have to program the Digital Table. However, if you only want to use the Digital part of the table, you must program the A/D part of the table.

NOTE: When the Channel-Gain Latch is used, Digital Table is ignored.

When using the Channel Gain Table, you should group your entries to maximize the throughput of your module. Low-level input signals and varying gains are likely to drop the throughput rate because low level inputs must drive out high level input residual signals. To maximize throughput:

- Keep channels configured for a certain range grouped together, even if they are out of sequence.
- Use external signal conditioning if you are performing high speed scanning of low level signals. This increases throughput and reduces noise.
- If you have room in the channel-gain table, you can make an entry twice to make sure that sufficient settling time has been allowed and an accurate reading has been taken. Set the skip bit for the first entry so that it is ignored.
- For best results, do not use the channel-gain table when measuring steady-state signals. Use the single convert mode to step through the channels.

Interrupts

Controller can receive interrupt request from up to 15 sources. These 15 sources cover the most important internal signals of the board plus 2 external signals:

- A/D Sample Counter Countdown

Interrupt is generated when the A/D Sample Counter counts down to zero.

This interrupt can be used to count more than 65535 samples by counting the turnovers of the Sample Counter.

- ❑ A/D Start Convert
Interrupt is generated when a conversion is started.
- ❑ A/D End Of Convert
Interrupt is generated when an end of convert is issued by the A/D converter.
- ❑ A/D FIFO Write
Interrupt is generated when sample enters the A/D FIFO.
This interrupt can be used for reading and processing samples real-time.
- ❑ A/D FIFO Half Full
Interrupt is generated when the A/D FIFO is half full.
- ❑ A/D DMA Done
Interrupt is generated when the A/D DMA done flag goes high.
- ❑ CGT Reset
Interrupt is generated when the Channel-Gain Table recycles execution to the first table entry.
This interrupt can be used for reading and processing a burst of samples from different channels real-time.
- ❑ CGT Pause
Interrupt is generated when Channel-Gain Table execution is paused waiting for a new trigger.
- ❑ External Pacer Clock
Interrupt is generated when the external pacer clock line is pulsed.
- ❑ External Trigger
Interrupt is generated when the external trigger line is pulsed.
- ❑ Digital Interrupt
Interrupt is generated when the Advanced Digital Trigger signals a Digital Interrupt.
This interrupt can be used to detect (and react on) certain patterns on Digital Input Port 0.
- ❑ User Timer/Counter 0 Out
- ❑ User Timer/Counter 0 Out, inverted
- ❑ User Timer/Counter 1 Out
Interrupt is generated on the ticks of User T/C0 (i.e., when the counter counts down to zero).
This interrupt gives you a general-purpose means of measuring real time, frequency, or counting events. It is also intended to use for Pulse output generation.
- ❑ Digital Input FIFO Half Full
Interrupt is generated when the Digital Input FIFO is half full.
- ❑ Digital Input FIFO Write
Interrupt is generated when sample enters the Digital Input FIFO.

To service interrupts with the DM6420 Driver is very simple.

Actually, when you install your interrupt handler, it is not a real Interrupt Service Routine, since you can not do that under Windows. Your handler is an ordinary routine, which is called by the real Interrupt Service Routine implemented by the DM6420 driver. The driver's Interrupt Service Routine schedules your interrupt handler for execution as a separate thread, acknowledges the interrupt to the board's interrupt controller and returns. After returning, the scheduled thread starts executing your interrupt handler.

Direct Memory Access (DMA)

DMA transfers data between peripheral device and PC memory without using the processor as an intermediate. This method allows very fast data transfer rates.

The DM6420 driver provides an interface for the user to handle the DMA channels of the board. The driver handles the dual-DMA mode of the DM6420, too.

The following program steps are necessary to use the DMA:

- ❑ Initialize the board, setup triggering modes.
- ❑ Install the DMA handler.
`InstallDMA6420(DEVICE_NO, &dmasetup, TRUE, 0, 0);`
- ❑ Program DMA channel.
`SetADDMA6420(DEVICE_NO, ADDMAChannel, 0);`
- ❑ Perform data collection.
- ❑ Close DMA channel.
`DeInstallDMA6420(DEVICE_NO, 0);`

To see how to use the DMA channel on the DM6420 board, see the dma and wdma example programs.

Driver API Function Groups

Driver Initialization Functions

OpenBoard
CloseBoard

General Board Control Functions

InitBoard6420
ClearBoard6420
ClearRegister6420
ReadStatus6420
LoadControlRegister6420
LoadTriggerRegister6420

FIFO Manipulation

ClearADFIFO6420
IsADFIFOEmpty6420
IsADFIFOFull6420
ClearDINFIFO6420
IsDINFIFOEmpty6420
IsDINFIFOHalf6420
IsDINFIFOFull6420
ReadADDDataWithMarker6420
ReadADDDataMarker6420
ReadDINFIFO6420

DMA Functions

InstallDMA
DeInstallDMA
StartDMA
FlushDMA
SetADDMA6420
ClearADDMADone6420
IsADDMADone6420
IsFirstADDMADone6420

A/D Converter

IsADHalted6420
IsADConverting6420
SetADSampleCounterStop6420
SetADPauseEnable6420
ReadADDData6420
StartConversion6420
SetConversionSelect6420
SetStartTrigger6420
SetStopTrigger6420

SetTriggerRepeat6420

D/A Converter

LoadDAC6420

Pacer Clock Handling

SetPacerClock6420

IsPacerClockOn6420

SetPacerClockSource6420

Burst Clock Handling

SetBurstClock6420

IsBurstClockOn6420

SetBurstTrigger6420

About Counter Functions

IsAboutTrigger6420

Sample Counter Functions

LoadADSampleCounter6420

Digital I/O Functions

IsDigitalIRQ6420

LoadDINConfigRegister6420

ConfigDINClock6420

DINClockEnable6420

SelectRegister5812

ClearChip5812

SetPort0Direction5812

SetPort1Direction5812

LoadMask5812

LoadCompare5812

ReadDIO5812

ReadCompareRegister5812

SelectClock5812

WriteDIO5812

Interrupt Handling

InstallCallbackIRQHandler

InstallCounterIRQHandler

RemoveIRQHandler

GetIRQCounter

ClearIRQ16420

ClearIRQ26420

IsIRQ16420
IsIRQ26420
LoadIRQRegister6420
SetIRQ16420
SetIRQ26420
ClearIrq5812
EnableIrq5812
SelectIrqMode5812
IsChipIRQ5812

Autoincrement Data Handling

GetAutoincDataByte
GetAutoincDataWord

User Timer-Counter Functions

SelectTimerCounter6420
ClockMode6420
ClockDivisor6420
SetUserClock6420
ReadTimerCounter6420
DoneTimer6420

Channel-Gain Table/Latch Manipulation

ClearChannelGainTable6420
ResetChannelGainTable6420
EnableTables6420
ChannelGainDataStore6420
ReadChannelGainDataStore6420
SetChannelGain6420
LoadADTable6420
LoadDigitalTable6420

External Trigger Configuration Functions

SetTriggerPolarity6420

Error Handling Functions

GetErrorStatus6420

Alphabetical Driver API Functions Reference

ChannelGainDataStore6420

Description:

This routine enables the Channel Gain Data Store feature of the board.

Parameters:

hBoard:	device handle
Enable:	0 = Disable 1 = Enable

ClearADFIFO6420

Description:

This routine is used to clear all the data from the A/D FIFO.

Parameters:

hBoard:	device handle
---------	---------------

ClearADDMADone6420

Description:

This routine is used to clear the A/D DMA done status bit.

Parameters:

hBoard:	device handle
---------	---------------

ClearBoard6420

Description:

This routine is used to clear board.

Parameters:

hBoard:	device handle
---------	---------------

ClearChannelGainTable6420

Description:

This routine is used to clear both the AD Table and the Digital Table.

Parameters:

hBoard: device handle

ClearChip5812_6420

Description:

This routine clears the selected DIO chip.

Parameters:

hBoard: device handle
SelectedChip: 0, 1, 2

ClearDINFIFO6420

Description:

This routine is used to clear the Digital Input FIFO.

Parameters:

hBoard: device handle

ClearIRQ16420

Description:

This routine is used to clear the IRQ 1 circuitry and status bit.

Parameters:

hBoard: device handle

ClearIRQ26420

Description:

This routine is used to clear the IRQ 2 status bit.

Parameters:

hBoard: device handle

ClearIrq5812_6420

Description:

This routine clears the selected DIO chips IRQ status bit.

Parameters:

hBoard: device handle

SelectedChip: 0, 1, 2

ClearRegister6420

Description:

This routine is used to write the clear register with one command and issue a clear to the board.

Parameters:

hBoard: device handle
ClearValue: 1 - 65535

ClockDivisor6420

Description:

This routine is used to set the divisor of a designated counter on the 8254 programmable interval timer (PIT). This procedure assumes that the counter has already been set to receive the least significant uchar8 (LSB) of the divisor followed by the most significant uchar8 (MSB).

Parameters:

hBoard: device handle
Timer: 0,1,2
Divisor: 0 - 65535

ClockMode6420

Description:

This routine is used to set the mode of a designated counter in the 8254 programmable interval timer (PIT).

Parameters:

hBoard: device handle
Timer: 0,1,2
Mode: 0,1,2,3,4,5

CloseBoard6420

Description:

This routine is used to close board driver.

Parameters:

DeviceNumber: WinRT device number to load
szBuf: message buffer

Return value:

TRUE on success, FALSE on error

Description:

This routine configures the Digital Input FIFO clock source.

Parameters:

hBoard:	device handle
DIN_Clock:	0 = user T/C out 0
	1 = user T/C out 1
	2 = write A/D FIFO
	3 = external pacer clock
	4 = external trigger
	5 = reserved
	6 = reserved
	7 = reserved

DeInstallDMA6420

Description:

This routine is used to remove DMA handler.

Parameters:

hBoard: device handle
dma_ch_index: DMA channel index (0/1)

Returns:

0 if success
error code if not success

DINClockEnable6420

Description:

This routine enables the Digital Input FIFO clock.

Parameters:

hBoard: device handle
DIN_Clock: 0 = disabled
1 = enabled

DoneTimer6420

Description:

Initialize the timers for high speed to ensure the immediate load.

Parameters:

hBoard: device handle

EnableIrq5812_6420

Description:

This routine enables the selected DIO chips interrupt.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2
Enable:	0 = Disabled 1 = Enabled

EnableTables6420

Description:

This Routine Enables and Disables both the AD and Digital Tables.

Parameters:

hBoard:	device handle
Enable_AD_Table:	0 = disable 1 = enable
Enable_Digital_Table:	0 = disable 1 = enable

FlushDMA6420

Description:

This routine flushes the DMA buffer.

Parameters:

hBoard: device handle

GetAutoincDataByte6420

Description:

This routine is used to get the data with autoincrement driver mode.
It uses the INP_B command to read data.

Parameters:

hBoard:	device handle
autoincsetup:	parameters for autoincrement command
DataNum:	number of data item to read

Return value:

TRUE on success, FALSE on error

GetAutoincDataWord6420

Description:

This routine is used to get the data with autoincrement driver mode.
It uses the INP_W command to read data.

Parameters:

hBoard:	device handle
autoincsetup:	parameters for autoincrement command
DataNum:	number of data item to read

Return value:

TRUE on success, FALSE on error

GetErrorStatus6420

Description:

This routine is used to get the error code and string in case of errors.
The function clears the last error code.

Parameters:

hBoard:	device handle
ErrorCode:	Error Code
ErrorString:	Return error string

GetIRQCounter6420

Description:

This routine is used to get the counter incremented by IRQ.

Parameters:

hBoard:	device handle
irqsetup:	return parameters to caller

InitBoard6420

Description:

This Routine Should always be called first.
This clears the board and variables the driver uses.

Parameters:

hBoard: device handle

InstallCallbackIRQHandler6420

Description:

This routine is used to install IRQ handler function.

Parameters:

hBoard: device handle
isrT handler: user callback function

Return value:

TRUE on success, FALSE on error

InstallCounterIRQHandler6420

This routine is used to install IRQ handler function.

Parameters:

hBoard: device handle
irqsetup: return parameters to caller

Return value:

TRUE on success, FALSE on error

InstallDMA6420

Description:

This routine is used to install DMA handler.

Parameters:

hBoard: device handle
dmasetup: return parameters to caller
fill: TRUE if buffer fill is necessary
fill_byte: DMA buffer filler

dma_ch_index: DMA channel index (0/1)

Return value:

TRUE on success, FALSE on error

IsAboutTrigger6420

Description:

This routine checks to see if the about trigger has occurred.

Parameters:

hBoard: device handle

Returns:

1 if trigger has occurred
0 if trigger has not occurred

IsADConverting6420

Description:

This routine checks to see if the AD converting.

Parameters:

hBoard: device handle

Returns:

1 if AD is converting
0 if AD is not converting

IsADDMADone6420

Description:

This routine checks to see if the A/D DMA transfer is done.

Parameters:

hBoard: device handle

Returns:

1 if DMA is done
0 if DMA is not done

IsADFIFOEmpty6420

Description:

This routine checks to see if the A/D FIFO is empty.

Parameters:

hBoard: device handle

Returns:

1 if FIFO is empty
0 if FIFO is not empty

IsADFIFOFull6420

Description:

This routine checks to see if the A/D FIFO is full.

Parameters:

hBoard: device handle

Returns:

1 if FIFO is full
0 if FIFO is not full

IsADHalted6420

Description:

This routine checks to see if the AD is halted.

Parameters:

hBoard: device handle

Returns:

1 if AD is halted
0 if AD is not halted

IsBurstClockOn6420

Description:

This routine checks to see if the burst clock is enabled.

Parameters:

hBoard: device handle

Returns:

1 if Burst Clock is on
0 if Burst Clock is off

IsChipIRQ5812_6420

Description:

This routine checks to see if the selected DIO chip has generated an interrupt.

Parameters:

hBoard: device handle
SelectedChip: 0, 1, 2

Returns:

1 if IRQ has been generated
0 if no IRQ

IsDigitalIRQ6420

Description:

This routine checks to see if the digital I/O chip has generated an interrupt.

Parameters:

hBoard: device handle

Returns:

1 if IRQ has been generated
0 if no IRQ

IsDINFIFOEmpty6420

Description:

This routine checks to see if the Digital Input FIFO is empty.

Parameters:

hBoard: device handle

Returns:

1 if FIFO is empty
0 if FIFO is not empty

IsDINFIFOFull6420

Description:

This routine checks to see if the Digital Input FIFO is full.

Parameters:

hBoard: device handle

Returns:

1 if FIFO is full
0 if FIFO is not full

IsDINFIFOHalf6420

Description:

This routine checks to see if the Digital Input FIFO is half full.

Parameters:

hBoard: device handle

Returns:

1 if FIFO is half full
0 if FIFO is not half full

IsFirstADDMADone6420

Description:

This routine checks to see if the A/D DMA transfer is done on the first channel.

Parameters:

hBoard: device handle

Returns:

1 if DMA is done
0 if DMA is not done

IsIRQ16420

Description:

This routine checks the IRQ 1 status bit.

Parameters:

hBoard: device handle

Returns:

1 if IRQ has been generated
0 if no IRQ

IsIRQ26420

Description:

This routine checks the IRQ 2 status bit.

Parameters:

hBoard: device handle

Returns:

1 if IRQ has been generated
0 if no IRQ

IsPacerClockOn6420

Description:

This routine checks to see if the pacer clock is running.

Parameters:

hBoard: device handle

Returns:

1 if Pacer Clock is on
0 if Pacer Clock is off

LoadADSampleCounter6420

Description:

This routine loads the A/D sample counter.

Parameters:

hBoard: device handle
NumOfSamples: 0 - 65535

LoadADTable6420

Description:

This routine loads the AD Table with the given number of entries.

Parameters:

hBoard: device handle
Num_of_Entry: 1-1024

The struct ADTableRow is defined in REG6420.h

```
typedef struct
{
    uint16 Channel: 0 - 15

    uint16 Gain: 0 = x1
                1 = x2
                2 = x4
                3 = x8
                4 = reserved
                5 = reserved
                6 = reserved
                7 = reserved

    uint16 ADRange: 0 = +/- 5v
                  1 = +/- 10v
                  2 = 0v - 10v
                  3 = reserved

    uint16 Se_Diff: 0 = single ended
                  1 = differential

    uint16 Pause: 0 = disabled
                 1 = enabled

    uint16 Skip: 0 = disabled
                1 = enabled
} ADTableRow;
```

LoadCompare5812_6420

Description:

This routine loads the selected DIO chips compare register.

Parameters:

hBoard: device handle
SelectedChip: 0, 1, 2
Compare: 0 - 255

LoadControlRegister6420

Description:

This routine loads the control register with one write operation.

Parameters:

hBoard: device handle
Value: 0 - 65535

LoadDAC6420

Description:

This routine loads the DAC value.

Parameters:

hBoard: device handle
Channel: 0 = DAC 1
 1 = DAC 2
Data: 0 - 4095

LoadDigitalTable6420

Description:

This routine loads the Digital Table with the given number of entries.

Parameters:

hBoard: device handle
Channel: 0 - 255
Num_of_Chan: 1 - 1024

LoadDINConfigRegister6420

Description:

This routine loads the Digital Input configuration register with one write operation.

Parameters:

hBoard: device handle
Value: 0 - 65535

LoadIRQRegister6420

Description:

This routine loads the interrupt register with one write operation.

Parameters:

hBoard: device handle
Value: 0 - 65535

LoadMask5812_6420

Description:

This routine loads the selected DIO chips mask register.

Parameters:

hBoard: device handle
SelectedChip: 0, 1, 2
Mask: 0 - 255

LoadTriggerRegister6420

Description:

This routine loads the trigger register with one write operation.

Parameters:

hBoard: device handle
Value: 0 - 65535

`OpenBoard6420`

Description:

This routine is used to open board driver.

Parameters:

DeviceNumber: WinRT device number to load
LoadDevice: Indicate to load WinRT driver or not
szBuf: Message buffer
BoardConfig: Return board configuration

Return value:

TRUE on success, FALSE on error

ReadADData6420

Description:

This Routine Reads the Data from the FIFO.

Parameters:

hBoard: device handle

Returns:

Signed 12 bit AD Data.

ReadADDataMarker6420

Description:

This Routine Reads the Data Marker from the FIFO.

Parameters:

hBoard: device handle

Returns:

3 bit Data Marker.

ReadADDataWithMarker6420

Description:

This Routine Reads the Data from the FIFO.

Parameters:

hBoard: device handle

Returns:

signed 12 bit AD Data and 3 bit Data Marker.

ReadChannelGainDataStore6420

Description:

This Routine Reads the Channel/Gain Data from the FIFO when the Channel Gain Data Store feature of the board is enabled.

Parameters:

hBoard: device handle

Returns:

16 bit value: Bottom 8 bits = A/D table value
Upper 8 bits = digital table value

ReadCompareRegister5812_6420

Description:

This routine reads the selected DIO chips compare register.

Parameters:

hBoard: device handle
SelectedChip: 0, 1, 2

ReadDINFIFO6420

Description:

This Routine Reads the Data from the Digital Input FIFO.

Parameters:

hBoard: device handle

Returns:

8 bit value.

ReadDIO5812_6420

Description:

This routine reads the selected DIO chips port.

Parameters:

hBoard: device handle
SelectedChip: 0, 1, 2

Port: 0, 1

ReadStatus6420

Description:

This routine returns the status from the board.

Parameters:

hBoard device handle

Returns:

16 bit unsigned integer

ReadTimerCounter6420

Description:

This routine is used to read the contents of the selected timer/counter.

Parameters:

hBoard: device handle
Timer: 0,1
Clock: 0,1,2

Returns:

uint16

RemoveIRQHandler6420

Description:

This routine is used to uninstall IRQ handler function.

Parameters:

hBoard: device handle

ResetChannelGainTable6420

Description:

This routine is used to reset both the AD Table and the Digital Table pointers to the first location in the table.

Parameters:

hBoard: device handle

SelectClock5812_6420

Description:

This routine sets the selected DIO chips clock source.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2
Clock:	0 = 8 MHz 1 = Programmable clock

SelectIrqMode5812_6420

Description:

This routine sets the selected DIO chips Irq mode.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2
IrqMode:	0 = Event mode 1 = Match mode

SelectRegister5812_6420

Description:

This routine picks the different registers on the selected DIO chip.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2
Select:	0 = Clear mode 1 = Port 0 direction 2 = Port 0 mask 3 = Port 0 compare

SelectTimerCounter6420

Description:

This routine selects one of the four 8254 timer chips.

Parameters:

hBoard:	device handle
Select:	0 = Clock TC (Pacer & Burst clocks)

- 1 = User TC (A/D sample counter & User timer/counters)
- 2 = reserved
- 3 = reserved

SetADDMA6420

Description:

This routine sets the A/D DMA channels.

Parameters:

hBoard: device handle
Channel1: 0 = disabled
 1 = DRQ 5
 2 = DRQ 6
 3 = DRQ 7
Channel2: 0 = disabled
 1 = DRQ 5
 2 = DRQ 6
 3 = DRQ 7

SetADPauseEnable6420

Description:

This routine enables and disables the A/D pause bit.

Parameters:

hBoard: device handle
Enable: 0 = Enable
 1 = Disable

SetADSampleCounterStop6420

Description:

This routine enables and disables the A/D sample counter stop bit.

Parameters:

hBoard: device handle
Enable: 0 = Enable
 1 = Disable

SetBurstClock6420

Description:

This routine sets the burst clock rate.

Parameters:

hBoard: device handle
Burst Rate: 500 KHz or less

Returns:

The actual clock frequency that is programmed.

SetBurstTrigger6420

Description:

This routine selects the burst trigger.

Parameters:

hBoard: device handle
Burst_Trigger: 0 = Software trigger
1 = pacer clock
2 = external trigger
3 = digital interrupt

SetChannelGain6420

Description:

This routine loads the channel/gain latch.

Parameters:

hBoard: device handle
Channel: 0 - 15
Gain: 0 = x1
1 = x2
2 = x4
3 = x8
4 = reserved
5 = reserved
6 = reserved
7 = reserved
Range: 0 = +/- 5v
1 = +/- 10v
2 = 0v - 10v
3 = reserved
Se_Diff: 0 = single ended
1 = differential

SetConversionSelect6420

Description:

This routine selects the conversion mode.

Parameters:

hBoard: device handle
Select: 0 = software trigger
1 = pacer clock
2 = burst clock
3 = digital interrupt

SetIRQ16420

Description:

This routine sets the source and channel for interrupt 1.

Parameters:

hBoard: device handle
Source: 0 = A/D sample counter
1 = A/D start convert
2 = A/D End-of-Convert
3 = A/D Write FIFO
4 = A/D FIFO half full
5 = A/D DMA done
6 = Reset channel/gain table
7 = Pause channel/gain table
8 = External Pacer Clock
9 = External trigger
10 = Digital chip interrupt
11 = User TC out 0
12 = User TC out 0 inverted
13 = User TC out 1
14 = Digital Input FIFO half full
15 = DIN Write FIFO
16 .. 31 = reserved

Channel: 0 = disabled
1 = IRQ 3
2 = IRQ 5
3 = IRQ 9
4 = IRQ 10
5 = IRQ 11
6 = IRQ 12
7 = IRQ 15

SetIRQ26420

Description:

This routine sets the source and channel for interrupt 2.

Parameters:

hBoard: device handle
Source: 0 = A/D sample counter
1 = A/D start convert
2 = A/D End-of-Convert

3 = A/D Write FIFO
4 = A/D FIFO half full
5 = A/D DMA done
6 = Reset channel/gain table
7 = Pause channel/gain table
8 = External Pacer Clock
9 = External trigger
10 = Digital chip interrupt
11 = User TC out 0
12 = User TC out 0 inverted
13 = User TC out 1
14 = Digital Input FIFO half full
15 = DIN Write FIFO
16 .. 31 = reserved

Channel: 0 = disabled
1 = IRQ 3
2 = IRQ 5
3 = IRQ 9
4 = IRQ 10
5 = IRQ 11
6 = IRQ 12
7 = IRQ 15

SetPacerClock6420

Description:

This routine sets the pacer clock rate. It will automatically decide whether to use a 16 or 32 bit clock depending on the rate.

Parameters:

hBoard: device handle
ClockRate: 500 KHz or less

Returns:

The actual clock frequency that is programmed.

SetPacerClockSource6420

Description:

This routine sets the pacer clock source.

Parameters:

hBoard: device handle
Source: 0 = Internal
1 = External

SetPort0Direction5812_6420

Description:

This routine sets the selected DIO chips port 0 direction.

Parameters:

hBoard: device handle
SelectedChip: 0, 1, 2
Direction: 0 - 255
 0 = In
 1 = Out

SetPort1Direction5812_6420

Description:

This routine sets the selected DIO chips port 1 direction.

Parameters:

hBoard: device handle
SelectedChip: 0, 1, 2
Direction: 0 = In
 1 = Out

SetStartTrigger6420

Description:

This routine selects the start trigger.

Parameters:

hBoard: device handle
Start_Trigger: 0 = software trigger
 1 = external trigger
 2 = digital interrupt
 3 = User TC Counter 1 out
 4 = reserved
 5 = reserved
 6 = reserved
 7 = gate mode

SetStopTrigger6420

Description:

This routine selects the stop trigger.

Parameters:

hBoard: device handle
Stop_Trigger: 0 = software trigger
 1 = external trigger

- 2 = digital interrupt
- 3 = sample counter
- 4 = about software trigger
- 5 = about external trigger
- 6 = about digital interrupt
- 7 = about user TC counter 1 out

SetTriggerPolarity6420

Description:

This routine sets the external trigger polarity.

Parameters:

hBoard:	device handle
Polarity:	0 = positive edge 1 = negative edge

SetTriggerRepeat6420

Description:

This routine sets the trigger repeat bit.

Parameters:

hBoard:	device handle
Repeat:	0 = Single Cycle 1 = Repeat Cycle

SetUserClock6420

Description:

This routine sets the user timer counters.

Parameters:

hBoard:	device handle
Timer:	0, 1, 2
InputRate:	Input clock to selected timer.
OutputRate:	Desired output rate from selected timer.

Returns:

The actual clock frequency that is programmed.

StartConversion6420

Description:

This routine is used to create software triggers or enable hardware triggers depending on the conversion mode.

Parameters:

hBoard: device handle

StartDMA6420

Description:

This routine starts DMA.

Parameters:

hBoard: device handle
UserLengt: user definable dma length
dma_ch_index: DMA channel index (0/1)

WriteDIO5812_6420

Description:

This routine writes the selected DIO chips port.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2
Port:	0, 1
Data:	0 - 255

Example Programs Reference

Win32 console applications

Name	Feature	Remarks
dma	Single DMA channel acquisition Sampling on pacer clock Start trigger: software Stop trigger: software	The data is displayed numerically on the screen.
speedtst	Sampling on pacer clock Start trigger: software Stop trigger: software Using IRQ handler function	It demonstrates three methods of data acquisition with interrupt: <ul style="list-style-type: none">• callback function• poll the IRQ counter• get data with autoincrement mode

Win32 Windows applications

Name	Feature	Remarks
wdac	Digital/analog conversation.	The user can change the output voltage with the on-screen slider.
wdigital	It programs the digital port 0 to input, port 1 to output.	The user can set the digital output port; the program reads the input port.
wdma	Single DMA channel acquisition. Sampling on pacer clock Start trigger: software Stop trigger: software	The data displayed graphically on the screen.
wdualdma	Dual DMA channel acquisition. Sampling on pacer clock Start trigger: software Stop trigger: software	It demonstrates the high speed gap-free data acquisition. The data stored in file.
whidin	High-speed digital input example. The user timer-counter is programmed to sample the digital input. At digital FIFO half full generated an interrupt, and the data is stored and displayed.	The data displayed graphically and stored in file.
wintrpts	The user timer-counter is programmed to generate interrupts. The analog input sampled when the IT occurs.	The data displayed graphically.
wrandom	Random channel acquisition with CGT. The pacer clock programmed to start conversion with burst clock.	The samples are displayed numerically.
wrepinsw	Sampling on pacer clock. Start trigger: software Stop trigger: software Interrupt on sample counter	On sample counter IT the program reads and displays the samples graphically.
wsmpcnt	Sampling on pacer clock. Start trigger: software Stop trigger: sample counter. Interrupt on sample counter	On sample counter IT the program reads and displays the samples graphically. The user can repeat the measure by pressing a key.
wsofttrig	Single A/D sampling. Start trigger: software Stop trigger: software	The user can sample an analog input by pressing a button. The sample displayed on the screen in Volts.
w2board	Example on how to use two boards.	Demonstration of the driver's multi-board feature.
wtimers	Demonstration of user timer-counters.	The counters are programmed to count the elapsed time in seconds.

