



**Digital Acquisition Developers Guide
For
National Instruments Acquisition Module**

(Document Number 700-0000062-R00)

10503 Timberwood Circle
Suite 120
Louisville, KY 40223
(502) 423-7225

Digital Acquisition Developers Guide
Document Number 700-0000062
May 2006

Copyright Lumitron 2006
All Rights Reserved.

DISCLAIMER

All copyrights in this manual, and the hardware and software described in it, are the exclusive property of Lumitron, Inc. and its licensors. Claim of copyright does not imply waiver of Lumitron's or its licensor's other rights in the work. See the following Notice of Proprietary Rights.

NOTICE OF PROPRIETARY RIGHTS

This manual and the related hardware and software are confidential trade secrets and the property of Lumitron and its licensors. Use, examination, reproduction, copying, transfer and/or disclosure to others of all or any part of this manual and the related documentation is prohibited except with the express written consent of Lumitron.

The information in this document is subject to change without notice. Lumitron makes no representations or warranties with respect to the contents of this manual and specifically disclaims any implied warranties of merchantability or fitness for a particular purpose. Lumitron Inc. assumes no responsibility for errors or omissions in this document.

Lumitron
10503 Timberwood Circle
Suite 120
Louisville, KY 40223
(502) 423-7225
FAX (502) 423-7064

Table of Contents

1 INTRODUCTION.....	1
2 GENERAL REQUIREMENTS.....	2
3 CFRAMEGRABBER CLASS REFERENCE.....	3
3.1 Class Data Members	3
3.2 CFrameGrabber.....	3
3.3 ~CFrameGrabber.....	4
3.4 FrmGrabCardReady.....	4
3.5 InitializeHardware.....	5
3.6 GetInterface	5
3.7 GetSession	5
3.8 GetIIDNames.....	6
3.9 CaptureStart	6
3.10 CaptureStop.....	7
3.11 CaptureStatus.....	7
3.12 Ni_CreateDisplayImage	7
3.13 Ni_ShowDisplayImage.....	8
3.14 Ni_InitShowCapture.....	8
3.15 Ni_GetCaptureHeader.....	9
3.16 Ni_SumCaptureData	9
3.17 Ni_GetCaptureData	10
3.18 Ni_SumSquaresCaptureData.....	12
3.19 Ni_CaptureStart.....	13
3.20 Ni_CaptureStop.....	13
3.21 Ni_CaptureStatus.....	14
3.22 Ni_InitAcqModule.....	14

3.23 Ni_DisplayError	14
3.24 LoadNIModule	14
3.25 LoadIIDNames	15
4 CFRMGRABCONFIG CLASS REFERENCE	16
4.1 Class Data Members	16
4.2 CFrmGrabConfig	16
4.3 ~CFrmGrabConfig	17
4.4 OnInitDialog	17
4.5 OnCbnSelchangeComboFrmgrabberId	17
APPENDIX A – NI FUNCTION MAPPING REFERENCE.....	19
APPENDIX B – FRAME GRAB CAPTURE INFORMATION STRUCTURES	20
APPENDIX C - FRAME I/O STRUCTURE.....	21
APPENDIX D – DIGITAL HEADER STRUCTURE	22

1 Introduction

This reference manual has been written to assist the developer in creating custom digital acquisition solutions for Lumitron's line of cameras based on the 320A Embeddable Camera Electronics System hardware as it integrates with a National Instruments NI-1426 Camera Link module.

The manual describes and shows examples of the MSVC++ CFrameGrabber and CFrmGrabConfig class objects as they are utilized in the CCI Camera Link software (P/N: 600-0000027). The classes can be inserted into a users existing project with just a few minor modifications or can be used a reference for creating a custom frame grabber class for a different manufacturer.

This manual has been written with the assumption that the user is knowledgeable about Microsoft Windows OS based applications and their development.

Source code referenced in the manual and include with the CD were developed using Microsoft Visual Studio .NET 2003.

2 General Requirements

Several components are required to begin developing digital acquisition software using Lumitron's interface libraries. They include software and hardware components from both Lumitron and National Instruments.

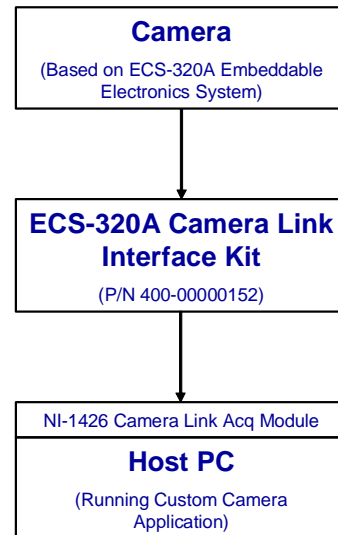
The following is a list of files that Lumitron provides with the Digital Acquisition Developers Guide (P/N: 600-0000029):

- FrameGrabber.cpp Class Source File
- FrameGrabber.h Class Header File
- FrmGrabConfig.cpp Class Source File
- FrmGrabConfig.h Class Header File
- NI Camera Files for Lumitron Cameras

In addition to the above files it will also be necessary to have National Instruments Image Acquisition software (NI-IMAQ) installed on the target platform. The NI software is required during runtime due to the explicit linking of the NI-IMAQ functions called in the CFrameGrabber class. The current release of NI-IMAQ is 3.5.1 and is available through the installation of their Vision Acquisition 8.0 software (provided with NI module).

The target frame grabber module that this manual references is the NI-1426 Camera Link. Although it is available in 16 or 32 MB versions; the 16 MB version typically is sufficient.

The diagram to the right shows a typical connection using a Lumitron Camera Link based camera, Camera Link interface kit, and host computer with NI acquisition module.



3 CFrameGrabber Class Reference

The object members listed in this section make up the class interface for the CFrameGrabber. The NI functions are linked explicitly at run time as long as the library (imaq.dll – v3.5.1) is present and each of the required functions exists in the library.

Appendix A shows contains the list of functions that are mapped into the CFrameGrabber class. Please refer to the NI-IMAQ documentation for further support concerning those functions.

3.1 Class Data Members

<u>Type</u>	<u>Identifier</u>	<u>Description</u>
HMODULE	m_hIMAQ	Handle to the IMAQ DLL module
bool	m_bFrmGrabReady	True if there is frame grabber access
int	m_nActFGIndex	Index to array of available interface files for camera configuration
int	m_LastBufNum	Index of last buffer number read
int	m_NumNiBuffers	Number of buffers reserved for acquisition
UWord32	m_nRoiTop	Acquisition region of interest top most pixel (from selected NI cfg file)
UWord32	m_nRoiLeft	Acquisition region of interest left most pixel (from selected NI cfg file)
UWord32	m_nRoiFullHeight	Acquisition region of interest image height – includes header line (from selected NI cfg file)
UWord32	m_nRoilmgHeight	Acquisition region of interest image height – no header (from selected NI cfg file)
UWord32	m_nRoilImageWidth	Acquisition region of interest image width (from selected NI cfg file)
ptr_cci_FrameGrabShow	m_pFGInfo	Pointer to frame grab information structure (see Appendix B)
char*	m_ImaqBuffer[]	Array of buffer pointers for acquisitions
BYTE*	m_InBuffer	Pointer to an acquisition input buffer
BYTE*	m_OutBuffer	Pointer to an acquisition output buffer
CStringArray*	m_pIID	Pointer to array of NI interface ID file names
SESSION_ID	m_SessionID	NI acquisition session ID
INTERFACE_ID	m_InterfaceID	NI acquisition interface ID
BUFLIST_ID	m_BufferListID	NI acquisition buffer list ID

3.2 CFrameGrabber

Description:

Public class constructor routine: will initialize object data members to default values and/or create any resources required.

Function Declaration:

CFrameGrabber()

Parameters:

None.

Returns:

None.

Example:

```
// Create pointer to framegrabber object
m_pFrameGrabber = new CFrameGrabber;
```

3.3 ~CFrameGrabber

Description:

Class destructor routine: will release any resources associated with the class object.

Function Declaration:

~CFrameGrabber()

Parameters:

None.

Returns:

None.

Example:

Destructor called automatically as application terminates.

3.4 FrmGrabCardReady

Description:

Public routine that will return 'true' if the frame grabber is present and properly initialized.

Function Declaration:

inline bool FrmGrabCardReady()

Parameters:

None.

Returns:

True if initialized.

Example:

```
// Enable/Disable controls if we have no frame grabber
if (theApp.m_pFrameGrabber->FrmGrabCardReady())
{
    m_chkCapture.EnableWindow(TRUE);

    // Get the window for the created button
    pWnd = GetDlgItem(IDC_BUTTON_IMAGE);
    if (pWnd != NULL)
    {
        // Save pointer to output window
        m_sGrabInfo.pDisplayWnd = pWnd->GetSafeHwnd();
    }
}
```

```
    }  
  
    // Create the Lead Tools windows  
    Create_LT_Bmp_Windows();  
  
    // Initialize capture thread related objects  
    InitCaptureThreads();  
}  
else  
{  
    // Disable capture button  
    m_chkCapture.EnableWindow(FALSE);  
}
```

3.5 InitializeHardware

Description:

Public routine to initialize the acquisition module. The provided index should correspond to a National Instruments interface file located in the m_pIID string array. Currently the software only enumerates NI ID files that begin with "LTron".

Function Declaration:

```
void InitializeHardware(UWord32 nIIDIndex)
```

Parameters:

UWord32 nIIDIndex: index into array of NI interface files that will be used to select the camera configuration.

Returns:

None.

3.6 GetInterface

Description:

Public inline routine to return the current interface ID value.

Function Declaration:

```
inline INTERFACE_ID GetInterface()
```

Parameters:

None.

Returns:

The currently active interface ID.

3.7 GetSession

Description:

Public inline routine to return the current session ID value.

Function Declaration:

```
inline SESSION_ID GetSession()
```

Parameters:

None.

Returns:

The currently active session ID.

3.8 GetIIDNames

Description:

Public routine to retrieve the list (string array) of available acquisition interface files. The class only searches for file names that begin with "LTron" (see paragraph 3.25).

Function Declaration:

```
inline CStringArray* GetIIDNames()
```

Parameters:

None.

Returns:

Pointer to a CStringArray with list of ID names.

Example:

```
// Fill up combobox with ID files
if (m_pFG != NULL)
{
    // Get pointer to ID files
    pNames = m_pFG->GetIIDNames();

    // Reset/clear combo box
    m_cbxFrameGrabber.ResetContent();

    // Loop thru all ID files
    for (nIndex = 0; nIndex < pNames->GetCount(); nIndex++)
    {
        m_cbxFrameGrabber.AddString(pNames->GetAt(nIndex));
    }

    // Set cursor to stored index
    m_cbxFrameGrabber.SetCurSel(m_nFGIndex);
}
```

3.9 CaptureStart

Description:

Public routine to update the objects data structure and then call the internal class function "Ni_CaptureStart" to begin acquisition.

Function Declaration:

```
void CaptureStart(ptr_cci_FrameGrabShow pFGInfo)
```

Parameters:

ptr_cci_FrameGrabShow pFGInfo: pointer to structure with information about the acquisition that is to be performed.

Returns:

None.

Example:

```
cci_FrameGrabShow m_sCalGrabInfo;
```

```
// Initialize desired capture settings
m_sCalGrabInfo.NiInfo.captCB = Ni_NucCfgDummyCallback;
m_sCalGrabInfo.NiInfo.numBuffers = 16;
m_sCalGrabInfo.NiInfo.skipCount = FRAME_SKIP_COUNT;
m_sCalGrabInfo.NiInfo.acquireCount = 16;

// Stop after number of frames grabbed
m_sCalGrabInfo.NiInfo.acqCommand = IMG_CMD_STOP;

// Call the capture start object
theApp.m_pFrameGrabber->CaptureStart(&m_sCalGrabInfo);

// Delay until done
while (theApp.m_pFrameGrabber->CaptureStatus() != 0)
;
```

3.10 CaptureStop

Description:

Public routine to end the acquisition process by calling the internal class function "Ni_CaptureStop".

Function Declaration:

```
void CaptureStop()
```

Parameters:

None.

Returns:

None.

3.11 CaptureStatus

Description:

Public routine to check the status of the current acquisition. Typically this would be used to check when acquisition of a fixed number of grabs is completed.

Function Declaration:

```
UWord32 CaptureStatus()
```

Parameters:

None.

Returns:

Non-zero if busy acquiring. .

Example:

See paragraph 3.9 above.

3.12 Ni_CreateDisplayImage

Description:

Public routine to check for the last buffer read and copy the data to appropriate buffers for additional processing. Image data copied to the object input buffer (m_InBuffer) will be passed to the user supplied automatic gain/leveling routine for creating an image for display.

Function Declaration:

```
bool Ni_CreateDisplayImage()
```

Parameters:

None.

Returns:

True if successful.

Example:

```

/*****
void CNucConfigView::ShowCapture()
{
    // Create the image (AGC version)
    if (theApp.m_pFrameGrabber->Ni_CreateDisplayImage())
    {
        // Show the image
        theApp.m_pFrameGrabber->Ni_ShowDisplayImage();
    }
}

```

3.13 Ni_ShowDisplayImage

Description:

Public routine used to display an image through the National Instruments display function based on values stored in the “pFGInfo” member structure. Typically this would be called after calling the “Ni_CreateDisplayImage” to generate an image suitable for display.

Function Declaration:

```
void Ni_ShowDisplayImage()
```

Parameters:

None.

Returns:

None.

Example:

See paragraph 3.12 above.

3.14 Ni_InitShowCapture

Description:

Low level (private) routine to initialize the input and output buffers, based on “pFGInfo” member structure, used to display acquired data. This routine is automatically called by the “Ni_CaptureStart” routine.

Function Declaration:

```
bool Ni_InitShowCapture()
```

Parameters:

None.

Returns:

True if successful.

3.15 Ni_GetCaptureHeader

Description:

Public routine to copy the header information from the acquired frame into the caller's buffer. The header is the first acquired line of the frame. See Appendix D for information about the format of the header.

Function Declaration:

```
void Ni_GetCaptureHeader(int frameNum, void* pBuf, size_t size)
```

Parameters:

int frameNum: frame number index into array of stored frames (typically 0).

void* pBuf: pointer to destination buffer.

size_t size: size of data (in bytes) to be copied to users buffer.

Returns:

None.

Example:

```
double          dTSWF = (double)m_nTempSensorWF/256.;
cam_InstHeaderOut instHdr = {0};

// Get the latest index
theApp.m_pFrameGrabber->Ni_GetCaptureHeader(
    0,
    (void*)&instHdr,
    sizeof(cam_InstHeaderOut)
);

// Compute front-end effective temperature
m_nEffAdcTemp = (UWord16)floor(
    ((dTSWF * (double)instHdr.adcAResults[2]) + ((1. - dTSWF) *
    (double)instHdr.adcAResults[3])) / 16.0 + 0.5
);
```

3.16 Ni_SumCaptureData

Description:

Public routine to fill a buffer in which each pixel position has been summed across a valid number of acquired frames. The header is not included.

Function Declaration(s):

```
int Ni_SumCaptureData(int numFrames, double* pBuf, int pixels)
```

```
int Ni_SumCaptureData(int numFrames, float* pBuf, int pixels)
```

Parameters:

int numFrames: intended number of frames to perform the summation.

double* pBuf: pointer to destination buffer.

float* pBuf: pointer to destination buffer.

int pixels: number of pixels to sum (typically the image size)

Returns:

Number of valid frames used in summation.

Example:

```

// Perform 16 frame acquisition
...

// Now compute average of the 16 frames

UWord32    c;

int        nNumFrames = 0;

float*     pStart;
float      frameAvg = -1.0;

// See which reference we are compiling
if (reference == COLD_DATA_CAPT)
{
    SAFE_DELETE_AR(m_pCold);

    m_pCold = new float[m_nFpaPixels];
    pStart = m_pCold;
}
else
{
    SAFE_DELETE_AR(m_pHot);

    m_pHot = new float[m_nFpaPixels];
    pStart = m_pHot;
}

// Zero buffer memory
ZeroMemory(pStart, sizeof(float) * m_nFpaPixels);

// Call the hardware specific compile capture routine
nNumFrames = theApp.m_pFrameGrabber->Ni_SumCaptureData(
    16,                // Number of frames to compile
    pStart,            // Pointer to destination buffer
    m_nFpaPixels       // Number of pixels in frame
);

// Now loop to average each pixel value
if (nNumFrames > 0)
{
    // Loop on all pixels in frame
    for (c = 0, frameAvg = 0.; c < m_nFpaPixels; c++)
    {
        // Average individual pixels
        pStart[c] /= (float)nNumFrames;
        // Compute frame sum
        frameAvg += pStart[c];
    }

    // Compute frame average
    frameAvg /= (float)m_nFpaPixels;
}

return (frameAvg);

```

3.17 Ni_GetCaptureData

Description:

Public routine to fill the destination buffer with image data from the selected frame number. The header is not included.

Function Declaration(s):

Digital Acquisition Developer's Guide

```
void Ni_GetCaptureData(int frameNum, UWord16* pBuf, size_t pixels)
```

```
void Ni_GetCaptureData(int frameNum, double* pBuf, size_t pixels)
```

Parameters:

int framesNum: desired frame number of image.

UWord16* pBuf: pointer to destination buffer.

double* pBuf: pointer to destination buffer.

int pixels: number of pixels to sum (typically the image size)

Returns:

None.

Example:

```
/* Acquire and display single frame */
void CAcquireView::OnButtonTestAcquire()
{
    SAFE_DELETE_AR(m_pImageData);

    m_pImageData = new WORD[m_nImageWidth * m_nImageHeight];

    if (m_pImageData != NULL)
    {
        BeginWaitCursor();

        // Init frame grab info struct
        m_GrabShow.NiInfo.captCB = Ni_AcqViewDummyCallback;
        m_GrabShow.NiInfo.numBuffers = NUM_RING_BUFS_STD;
        m_GrabShow.NiInfo.skipCount = 0;
        m_GrabShow.NiInfo.acquireCount = 0;
        m_GrabShow.NiInfo.acqCommand = IMG_CMD_STOP;

        // Call the capture start object
        theApp.m_pFrameGrabber->CaptureStart(&m_GrabShow);

        // Delay until done
        while (theApp.m_pFrameGrabber->CaptureStatus() != 0)
            ;

        // Get the captured data
        theApp.m_pFrameGrabber->Ni_GetCaptureData(
            0,
            m_pImageData,
            (size_t)(m_nImageWidth * m_nImageHeight)
        );

        // Stop the acquisition
        theApp.m_pFrameGrabber->CaptureStop();

        // Display (third party routine)
        m_pImageWindow->SetWindowBitmap(
            m_nImageWidth,
            m_nImageHeight,
            WORD_FORMAT,
            (void*)m_pImageData
        );

        // Set the image in the window to 100%
        m_pImageWindow->SetZoomMode(ZOOM_NORMAL, TRUE);

        // Update Zoom Percent
        UpdateZoomPercent();
    }
}
```

```

        EndWaitCursor();
    }
}

```

3.18 Ni_SumSquaresCaptureData

Description:

Public routine to fill a buffer in which each pixel position has been squared and then summed across a valid number of acquired frames. The header is not included.

Function Declaration:

```
int Ni_SumSquaresCaptureData(int numFrames, double* pBuf, int pixels)
```

Parameters:

int numFrames: intended number of frames to perform the summation.

double* pBuf: pointer to destination buffer.

int pixels: number of pixels to sum (typically the image size)

Returns:

Number of valid frames used in summation.

Example:

```

UWord32    pix;

int        nMeanFrames,
          nSSFrames;

double     fFrames,
          fDenominator,
          *pMean,
          *pNoise;

// Clear out the buffer before filling it up
ZeroMemory((void*)pCaptMean, sizeof(double) * imageSize);

// Get the sum of the captured data
nMeanFrames = theApp.m_pFrameGrabber->Ni_SumCaptureData(
    frames,
    pCaptMean,
    imageSize
);

if (nMeanFrames > 0)
{
    // Clear out the buffer before filling it up
    ZeroMemory((void*)pSSquares, sizeof(double) * imageSize);

    // Get the sum of the squares from the captured data
    nSSFrames = theApp.m_pFrameGrabber->Ni_SumSquaresCaptureData(
        frames,
        pSSquares,
        imageSize
    );

    // Make sure we have the same number of frames
    if (nSSFrames == nMeanFrames)
    {
        fFrames = (double)nSSFrames;
        fDenominator = fFrames * (fFrames - 1.);

        // Init pointers

```

Digital Acquisition Developer's Guide

```

pMean = pCaptMean;
pNoise = pSSquares;

// Now compute the mean of each pixel
for (pix = 0; pix < imageSize; pix++)
{
    if (pMean[pix] != 0)
    {
        // Compute SDev for each pixel
        pNoise[pix] = ((fFrames * pNoise[pix]) - (pMean[pix] *
        pMean[pix]))/fDenominator;

        pNoise[pix] = sqrt(pNoise[pix]);
    }
}

return (true);
}
else
{
    return (false);
}
}
else
{
    return (false);
}
}

```

3.19 Ni_CaptureStart

Description:

Low level (private) routine to begin the acquisition process as configured by the “m_pFGInfo” data structure. This routine is called by the user routine “CaptureStart”.

Function Declaration:

```
void Ni_CaptureStart()
```

Parameters:

None.

Returns:

None.

3.20 Ni_CaptureStop

Description:

Low level (private) routine to end the current acquisition process. This routine is called by the user routine “CaptureStop”.

Function Declaration:

```
void Ni_CaptureStop()
```

Parameters:

None.

Returns:

None.

3.21 Ni_CaptureStatus

Description:

Low level (private) routine to check the status of the current acquisition. This routine is called by the user routine "CaptureStatus".

Function Declaration:

```
void Ni_CaptureStatus()
```

Parameters:

None.

Returns:

Non-zero if busy acquiring.

3.22 Ni_InitAcqModule

Description:

Low level (private) routine to initialize the acquisition module. This routine is called by the user routine "InitializeHardware".

Function Declaration:

```
void Ni_InitAcqModule(UWord32 nIndex)
```

Parameters:

UWord32 nIndex: index into array of NI interface files that will be used to select the camera configuration.

Returns:

None.

3.23 Ni_DisplayError

Description:

Low level (private) routine to generate an acquisition process error message. The error codes are NI specific; refer to the file "niimaq.h" for further information.

Function Declaration:

```
void Ni_DisplayError(Word32 errCode)
```

Parameters:

Word32 errCode: error code to be displayed.

Returns:

None.

3.24 LoadNIModule

Description:

Low level (private) routine to open the NI-IMAQ driver library. This routine is called by the object constructor. If the target dll is not present or the routine fails to load any of the required functions; then no acquisitions can be started. The "FrmGrabCardReady" routine will return false.

Function Declaration:

```
HMODULE LoadNIModule()
```

Parameters:

None.

Returns:

Handle to the opened NI-IMAQ dynamic link library module.

3.25 LoadIIDNames

Description:

Low level (private) routine to enumerate the available NI interface ID files and place the file names into the string array object "m_pIID". Currently the software only enumerates NI ID files that begin with "LTron".

Function Declaration:

```
void LoadIIDNames(void)
```

Parameters:

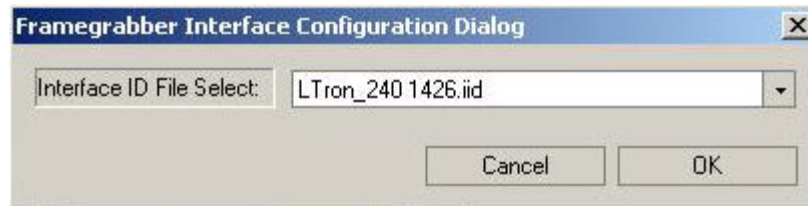
None.

Returns:

None.

4 CFrmGrabConfig Class Reference

The CFrmGrabConfig class described below is one of many possible ways to present the user with an interface control for selecting the desired NI interface for module configuration. The class is derived from the CDialog class and is menu activated in the CCI Camera Link software (see picture below).



4.1 Class Data Members

Type	Identifier	Description
CXtButton	m_btnOK	"OK" button control
CXtButton	m_btnCancel	"Cancel" button control
CXtFlatComboBox	m_cbxFrameGrabber	ComboBox control
CFrameGrabber*	m_pFG	Pointer to CFrameGrabber class object
int	m_nFGIndex	Index of selected file

4.2 CFrmGrabConfig

Description:

Public class constructor routine: will initialize object data members to default values and/or create any resources required.

Function Declaration:

```
CFrmGrabConfig(CFrameGrabber* pFG, int nActIndex = 0, CWnd* pParent = NULL)
```

Parameters:

None.

Returns:

None.

Example:

```

/*****
void CCCIApp::OnFramegrabberSetup()
{
    CDocument* pDoc = GetActiveDocument();

    // Create dialog object pointer
    CFrmGrabConfig* pFGDlg = new CFrmGrabConfig(m_pFrameGrabber, m_nFGIndex);

    // If user selected OK
    if (pFGDlg->DoModal() == IDOK)
    {
        // Get selected index

```

Digital Acquisition Developer's Guide

```
m_nFGIndex = pFGDlg->m_nFGIndex;

// Set the value in registry
SetAppSettingsFrmGrabber();

// Re-Init hardware
CheckForFramegrabber();

// Send updated data to all tab views
if (pDoc)
    pDoc->UpdateAllViews(NULL, cciUpdate_FrameGrabber, NULL);
}

SAFE_DELETE(pFGDlg);
}
```

4.3 ~CFrmGrabConfig

Description:

Class destructor routine: will release any resources associated with the class object.

Function Declaration:

```
~ CFrmGrabConfig()
```

Parameters:

None.

Returns:

None.

4.4 OnInitDialog

Description:

Internal class routine that will return 'true' when the dialog has been properly initialized. Called automatically.

Function Declaration:

```
BOOL OnInitDialog ()
```

Parameters:

None.

Returns:

True when done with initialization.

4.5 OnCbnSelchangeComboFrmgrabberId

Description:

Internal routine that updates the index based on the selected cursor position.

Function Declaration:

```
void OnCbnSelchangeComboFrmgrabberId ()
```

Parameters:

None.

Digital Acquisition Developer's Guide

Returns:

None.

Appendix A – NI Function Mapping Reference

```
// Type definition for each of the NI functions to be used by the CFramegrabber object
typedef Word32 (CALLBACK* IMGINTERFACEOPEN)(const char*, INTERFACE_ID*);
typedef Word32 (CALLBACK* IMGSESSIONOPEN)(INTERFACE_ID, SESSION_ID*);
typedef Word32 (CALLBACK* IMGCLOSE)(UWord32, UWord32);
typedef Word32 (CALLBACK* IMGSNAP)(SESSION_ID, void**);
typedef Word32 (CALLBACK* IMGGRABSETUP)(SESSION_ID, UWord32);
typedef Word32 (CALLBACK* IMGSESSIONSTOPACQUISITION)(SESSION_ID);
typedef Word32 (CALLBACK* IMGSETATTRIBUTE)(UWord32, UWord32, UWord32);
typedef Word32 (CALLBACK* IMGGETATTRIBUTE)(UWord32, UWord32, void*);
typedef Word32 (CALLBACK* IMGCREATEBUFLIST)(UWord32, BUFLIST_ID*);
typedef Word32 (CALLBACK* IMGCREATEBUFFER)(SESSION_ID, UWord32, UWord32, void**);
typedef Word32 (CALLBACK* IMGSETBUFFERELEMENT)(BUFLIST_ID, UWord32, UWord32, UWord32);
typedef Word32 (CALLBACK* IMGSESSIONCONFIGURE)(SESSION_ID, BUFLIST_ID);
typedef Word32 (CALLBACK* IMGSESSIONWAITSSIGNALASYNC2)(SESSION_ID, IMG_SIGNAL_TYPE, UWord32,
UWord32, CALL_BACK_PTR, void*);

typedef Word32 (CALLBACK* IMGSESSIONACQUIRE)(SESSION_ID, UWord32, CALL_BACK_PTR);
typedef Word32 (CALLBACK* IMGSESSIONABORT)(SESSION_ID, UWord32*);
typedef Word32 (CALLBACK* IMGDISPOSEBUFFER)(void*);
typedef Word32 (CALLBACK* IMGDISPOSEBUFLIST)(BUFLIST_ID, UWord32);
typedef Word32 (CALLBACK* IMGSESSIONCOPYBUFFER)(SESSION_ID, UWord32, BYTE*, UWord32);
typedef Word32 (CALLBACK* IMG PLOT)(GUIHNDL, void*, UWord32, UWord32, UWord32, UWord32, UWord32,
UWord32, UWord32);

typedef Word32 (CALLBACK* IMGSHOWERROR)(IMG_ERR, char*);
typedef Word32 (CALLBACK* IMGSESSIONSTATUS)(SESSION_ID, UWord32*, UWord32*);
typedef Word32 (CALLBACK* IMGSESSIONGETROI)(SESSION_ID, UWord32*, UWord32*, UWord32*, UWord32*);
typedef Word32 (CALLBACK* IMGSESSIONGETBUFFERSIZE)(SESSION_ID, UWord32*);
typedef Word32 (CALLBACK* IMGINTERFACEQUERYNAMES)(UWord32, char*);
typedef Word32 (CALLBACK* IMGINTERFACERESET)(INTERFACE_ID);

// National Instruments DLL Mapped Functions
// Format: Add ni_ to standard NI function name
IMGINTERFACEOPEN          ni_imgInterfaceOpen;
IMGSESSIONOPEN           ni_imgSessionOpen;
IMGCLOSE                 ni_imgClose;
IMGSNAP                  ni_imgSnap;
IMGGRABSETUP             ni_imgGrabSetup;
IMGSESSIONSTOPACQUISITION ni_imgSessionStopAcquisition;
IMGSETATTRIBUTE          ni_imgSetAttribute;
IMGGETATTRIBUTE           ni_imgGetAttribute;
IMGCREATEBUFLIST         ni_imgCreateBufList;
IMGCREATEBUFFER           ni_imgCreateBuffer;
IMGSETBUFFERELEMENT      ni_imgSetBufferElement;
IMGSESSIONCONFIGURE      ni_imgSessionConfigure;
IMGSESSIONWAITSSIGNALASYNC2 ni_imgSessionWaitSignalAsync2;
IMGSESSIONACQUIRE       ni_imgSessionAcquire;
IMGSESSIONABORT          ni_imgSessionAbort;
IMGDISPOSEBUFFER         ni_imgDisposeBuffer;
IMGDISPOSEBUFLIST       ni_imgDisposeBufList;
IMGSESSIONCOPYBUFFER     ni_imgSessionCopyBuffer;
IMG PLOT                  ni_imgPlot;
IMGSHOWERROR             ni_imgShowError;
IMGSESSIONSTATUS         ni_imgSessionStatus;
IMGSESSIONGETROI         ni_imgSessionGetROI;
IMGSESSIONGETBUFFERSIZE ni_imgSessionGetBufferSize;
IMGINTERFACEQUERYNAMES   ni_imgInterfaceQueryNames;
IMGINTERFACERESET       ni_imgInterfaceReset;
```

Appendix B – Frame Grab Capture Information Structures

```

/*
   Data Structure that contains information for size, type, output, etc., for
   captures. Structure is passed to FrameGrabber object.
*/
struct _cci_FrameGrabShow
{
    LPVOID      pParent;                // Pointer to parent

    // Pointer to an exposed function that will point to the parent
    // class routine to build the output (display) image
    bool  (*cbViewRoutine)(LPVOID, Word16, Word16, LPVOID, LPVOID);

    int      zoomFctr;                 // Zoom factor (0 is 1:1, 1 is 2:1, ...)
    HWND     pDisplayWnd;              // Output location for capture preview

    cci_ShowCaptNI      NiInfo;      // National Instruments initialization and usage info
};
typedef struct _cci_FrameGrabShow cci_FrameGrabShow, *ptr_cci_FrameGrabShow;

/* NI Capture View Structure */
struct _cci_ShowCaptNI
{
    // Pointer to NI specific capture callback routine
    UWord32      (*captCB)(SESSION_ID sid, IMG_ERR err, IMG_SIGNAL_TYPE type, UWord32 signalID,
void*      userData);

    int      numBuffers;               // Number of buffers to be used in acquisition
    int      skipCount;               // Number of frames to skip between acquires
    int      acquireCount;            // Number of frames to acquire (if not continuous)
    int      acqCommand;              // Command to be supplied to acquisition process
                                        // For Continuous Grab: IMG_CMD_LOOP
                                        // For Sequence Grab: IMG_CMD_STOP
};
typedef struct _cci_ShowCaptNI cci_ShowCaptNI, *ptr_cci_ShowCaptNI;

```

Appendix C - Frame I/O Structure

```
// Structure to pass information for acquiring frame data
struct _img_GetFrameInfo
{
    int         nCamNum;           // Desired camera (0 - 3)
    int         nUnits;           // Desired buffer units (A/D, F, C)
    int         nReserved[6];

    double      dEmissivity;      // Image emissivity
    double      dBkGndTempInF;    // Image Background Temp
    double      dReserved[2];

    short*      pOutShort;        // Pointer to output buffer (short)
    float*      pOutFloat;       // Pointer to output buffer (float)
};
typedef struct _img_GetFrameInfo img_GetFrameInfo, *ptr_img_GetFrameInfo;
```

Appendix D – Digital Header Structure

```

/* Instrumentation Header Data (Output from digital port) Structure */
struct _cam_InstHeaderOut
{
    /* Frame Counter */
    UWord32      count;           /* 2 Words */

    /* Camera Serial Number Information */
    cam_SerNum   camera;         /* 3 Words */

    /* FPA Serial Number Information */
    cam_SerNum   fpa;           /* 3 Words */

    /* Software Version */
    UWord16      swVersion;      /* 1 Word */

    /* Software Build */
    UWord16      swBuild;        /* 1 Word */

    /* FPGA Proc Version */
    UWord16      fpgaVersion[2]; /* 2 Words */

    /* Op Mode Name */
    UWord16      actOpName[4];   /* 4 Words */

    /* NVM Config Data (same as global config data) */
    nvm_GlobalCfg nvmData;       /* 40 Words */

    /* ADC Channel A Register Values (A0, A1, A2, A3) */
    UWord16      adcAResults[4]; /* 4 Words */

    /* ADC Channel B Register Values (B0, B1, B2, B3) */
    UWord16      adcBResults[4]; /* 4 Words */

    /* NUC Mode data structure */
    nuc_Mode      nucMode;       /* 132 Words */

    /* Reserve for user supplied radiometric data */
    UWord16      oemData[32];    /* 32 Words */

    /* Reserved data space */
    UWord16      reserved[30];   /* 30 Words */
};
typedef struct _cam_InstHeaderOut cam_InstHeaderOut, *ptr_cam_InstHeaderOut;

```