CraneWow Programmer Manual

Introduction

Programming effectively for the digiBASE-E takes some knowledge of COM, including familiarity with the VARIANT class and how to extract its data. Furthermore, some operations must be learned through trial-and-error as specific details of function are not documented.

Communication with the digiBASE-E is accomplished in CraneWow via two COM controls – a "List" drop list and a "Conn" control. The drop list is contained in the "umcbidrop" module, and functions like a conventional drop list. It will automatically procure the master list from the ORTEC MIO Server program, and return the "detector address" of the detector selected. This address can be fed into the Conn control (umcbiconn) to communicate with the device.

The easiest way to include these objects is to use the Visual Studio dialog editor, dragging the objects into a dialog box. The Conn control generates a small box containing a version number, while the List control generates a familiar-looking drop box.

VARIANT and SAFEARRAY

A VARIANT is a data type used by COM to pass data safely between different processes. It is a VARIANT that the GetRawData() procedure of the Conn control returns. Layered in the object is the SAFEARRAY that CraneWow uses to extract the digiBASE-E data stream.

Extracting the SAFEARRAY from the VARIANT first requires identification of the VARIANT (example v). There is a value v.vt that contains a variable type identifier. For the purposes of CaneWow, v.vt must have the VT_ARRAY and VT_I4 types. VT_ARRAY indicates that the VARIANT contains a SAFEARRAY, and VT-I4 indicates that it is an array of four-byte words.

If v.vt indicates VT_ARRAY, v.parray will be the SAFEARRAY contained within the VARIANT. SafeArrayGetElement() can then be used to obtain an element of that array. Note that instead of assigning the return of this function, one must provide a pointer to which the element should be copied; instead of an index, one provides a pointer to a variable containing the index desired.

digiBASE-E Data Format

The raw data of the digiBASE-E consists of four-byte words. When the data comes in as polled by GetRawData(), it contains 2048 of these words, with the zeroth one containing the number of valid words. Thus, most of the 2047 remaining channels may be filled with gibberish or empty, and must be ignored.

The valid words make up part of a data stream that can be saved to disk and parsed at a later date or on the fly. The first two bits of each word make up a flag that indicates the word type. Thus, there are four word types:

0 (00): Sync pulse signal time stamp.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	r	r	r	r	r	r	r	r	r	r	r	r	p	p	p	p	p	p	p	p	p	p	p	p	p	p	p	p	p	p

r: real time (29-18)

p: prescale (17-0)

The real time portion of the stamp, found at bits 12-0, will reset every 80 stamps, rolling over every 8 seconds. This time stamp is in units of 100ms. Prescale is in the units of 80ns since the last real time stamp. This is inserted into the data stream each time a pulse is propagated around the sync loop by the SyncMaster digiBASE-E. If Sync or SyncMaster gating are not used, this word will not appear.

1 (01): Live time stamp.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t

t: live time (29-0)

The live time is given as a tick count of 10ms per tick. The live time stamp accompanies a real time stamp and represents the live time at that instant.

2 (10): Real time stamp.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t	t

t: real time (29-0)

The real time is recorded as a tick count updated every 10ms . This simply counts the time since data acquisition was engaged for the digiBASE-E. Note that the real time and live time stamps will not trigger at the very beginning of the data stream. That is, there will not be any real time or live time stamps reading zero time.

3 (11): ADC word.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	r	d	d	d	d	d	d	d	d	d	d	d	d	p	p	p	p	p	p	p	p	p	p	p	p	p	p	p	p	p

r: memory routing bit

d: channel or bin number (28-17)

p: prescale (16-0)

The memory routing bit is not used by CraneWow. Bits 28-17 contain the channel number, or energy bin, that the event is attached to. The remaining bits contain a prescale time stamp in units of 80ns. This is the number of ticks since the last real time stamp, and can pinpoint the event's time in the data stream. When recreating spectra, this represents one count in the energy bin listed. Counting all the ADC words of a given channel in a span of time will allow construction of a familiar spectrum.

COM and Threading

It is important to note that the COM objects used by CraneWow are not threadable. This is the reason for using timers instead of worker threads on a loop. There are, however, threadable procedures available in the mcbio32.dll library.

When threading with ORTEC devices, it is important to run MIOStartup() at the beginning of the thread and MIOCleanup at the thread's conclusion. See the digiBASE-E manual for details.

Selected digiBASE-E Commands

All of these commands can be entered in upper, lower, or any case. They are sent as LPCSTRs via the Comm() function of the m_Conn control. All commands contain a dollar sign ($) with a following letter code corresponding to the response type. Commands that generate number responses have the last three characters as an 8 bit checksum of the rest of the response. As an example, for the SHOW_ACTIVE command, $C00000087 indicates inactive, while $C00001088 indicates active. Note that these are returned as strings by the Conn control.

CLEAR_ALL

This will clear the channels of the digiBASE-E as well as any live or real time presets and counters. It also clears ROI settings.

CLEAR_DATA

This will only clear the channels of the digiBASE-E. It will not clear the flags, presets, or other counters. It does not seem to clear or reset the FIFO.

DISABLE_HV

This will disable the high voltage across the detector.

ENAB_HV

This will enable the high voltage across the detector. The value is set beforehand, likely using MAESTRO or GammaVision, or via SET_HV.

RESET
This resets the digiBASE to its just-powered-on state. This has the effect of clearing the data channels, FIFO, and presets, and disabling the voltage.

SET_DATA_APPLICATION "string1" "string2"
This instructs the digiBASE-E to store a string with another string as its label. This string can later be returned. CraneWow and MAESTRO/GammaVision use the "SampleDescription" string to store a description of the sample. CraneWow adds this to the .dat file header. However, CraneWow accesses this through the SetAppDataName and SetAppDataValue functions of the Conn control. Note that there is a 32 and 128 byte maximum on string1 and string2, respectively.

SET_GATE "gate"
This will set a gate mode to the digiBASE-E. For the purposes of CraneWow, 0 (None), 6 (Sync), and 7 (SyncMaster) are the only relevant modes.

SET_HV "volts"
This sets a number, in volts, as a target for the digiBASE-E. The digiBASE-E has a tendency to hit a few volts higher than this target.

SET_LIVE "time"
This will set the live time counter to a particular time, in 20ms ticks. It will not work while the digiBASE-E is acquiring data.

SET_MODE_LIST
This will set the digiBASE-E to list mode. While in list mode, a first-in-first-out (FIFO) buffer will accrue data. When data is requested from the device, the first 2047 (or fewer) four byte words of data from the FIFO will be reported in the digiBASE-Es data channels, rather than a spectrum.

SET_MODE_PHA
This will set the digiBASE-E to pulse-height analysis mode. While in PHA mode, hits on the detector are stored as counts in bins corresponding to energy ranges. These are reported when requested in the digiBASE-E data channels.

SET_TRUE "time"
This will set the real time of the digiBASE-E to an arbitrary time, in 20ms ticks. It will not work while the digiBASE-E is acquiring data.

SHOW_ACTIVE
This will tell you if the digiBASE-E is acquiring spectral data, giving a one for active and zero for inactive.

SHOW_BT
This will return the digiBASE-E board temperature, in Celsius.

SHOW_CRM

This returns the count rate as measured by the digiBASE-E. This is independent of the data stream and any discriminators, and is a raw measure of events as detected by the crystal. It is in the units of counts per second.

SHOW_DATA_APPLICATION "string1"

This will show the string associated with the label "string1" in the application data memory. CraneWow and MAESTRO/GammaVision use the "SampleDescription" string to store a description of the sample. CraneWow adds this to the .dat file header. However, CraneWow accesses this through the SetAppDataName and SetAppDataValue functions of the Conn control. Note that there is a 32 and 128 byte maximum on string1 and string2, respectively.

SHOW_GATE

This will return a string indicating the gating mode of the digiBASE-E. For the purposes of CraneWow, "None", "Sync", and "SyncMaster" are the only relevant modes.

SHOW_HV

This will return the current voltage across the detector, as well as a mask indicating the status of the high voltage. The first five characters are the voltage, while the next five are a 16-bit decimal number containing the status mask. Bit 2 of this number indicates whether the high voltage is enabled (1) or disabled (0).

SHOW_LIVE

This will return the live time of the detector's acquisition counted in 20ms ticks.

SHOW_MODE

This will return a three-character string indicating the mode of the detector. For the purposes of CraneWow, PHA (pulse-height analysis) and LIS (list mode) are relevant.

SHOW_TRUE

This will return the real time of the detector's acquisition counted in 20ms ticks.

START

This begins acquisition. Note that the FIFO buffer will fill up if data is not polled regularly from the device. Also note that this does not automatically enable a voltage bias or reset the time counters.

STOP

This ends acquisition. Note that this does not automatically reset the time counters, nor does it automatically turn off the high voltage.

Variables and Functions

CMasterDlg Class

CMasterDlg()
Standard constructor for the CMasterDlg class.. Currently empty.

void OnDespawn(CCraneWowDlg *dying)
This is called when a CCraneWowDlg is closed. The CMasterDlg needs to release the CCraneWowDlg from memory and remove it from the index list. dying is simply a pointer to the CCraneWowDlg being closed.

long GetRT()
The master dialog keeps a timer when the StartAll button is pressed. This can be accessed to correct FIFO reset time seep. This returns the real time set by SetRT in ms.

int SetRT(long RT)
This sets referenceRT to a particular time RT, in ms. The system tick count at that time is also recorded.

BOOL OnInitDialog()
Run when the dialog is first opened. Sets default file name and options for sum histograms.

afx_msg
These message handlers activate when particular buttons are pressed on the master dialog box.

CCraneWowDlg *m_Dlgs[MAX_CRANEWOW_DIALOGS]
This is the master array of CCraneWowDlgs that are opened by the user. Each one is assigned a unique number kept in the same index of m_DlgIndices. CCraneWowDlgs are created as needed when the "Spawn" button is pressed, and deallocated when they are closed.

int m_DlgIndices[MAX_CRANEWOW_DIALOGS]
These numbers are added to MYEVENTID for each CCraneWowDlg's individual data timer.

int m_DlgCount
This is the number of CCraneWowDlgs extant at any time.

int m_DlgIndex
This is the next unique index that should be supplied to a new CCraneWowDlg, to be added to MYEVENTID for that dialog's individual data timer.

bool m_Started
Toggles true when StartAll is pressed, and false when no CCraneWowDlgs are gathering data. Note that this will still be false if CCraneWowDlgs are started individually.

double m_Interval

The interval at which all CCraneWowDlgs are assumed to be generating histograms. It is obtained from the m_iEvery of the zeroth CCraneWowDlg in the m_Dlgs array.

CString m_FileName

The filename beginning string to use for the summed histograms. A number and .txt extension are automatically added to this string to determine the true file name of each histogram.

unsigned long int referenceRT

The data collection real time set to the computer for the purposes of avoiding FIFO reset time seep. Can be used with referenceTicks to determine approximate data collection real time.

unsigned long referenceTicks

The system tick count at which referenceRT was set. Can be used to determine approximate data collection real time.

#defines from CraneWowDlg.h

MAX_CRANEWOW_DIALOGS

This is used to define the maximum number of CraneWow control windows. Currently set to 12, the theoretical maximum for sync mode.

MYEVENTID

This is an event ID for the timers that collect and collate data. Currently set to a pseudorandom number. Each CraneWow dialog has an event ID of MYEVENTID + m_DlgID, so that they are all unique and can be stopped/started individually.

NUMHIST

This defines the maximum number of working histograms. Should not be set to any value other than 3.

TIMERINTERVAL

This defines the interval (in milliseconds) between each GetData() call. Currently set to 3; this may actually be shorter than the GetData() call takes. This value was decided by trial and error to allow the highest throughput. It may need to be adjusted on other systems.

CraneWowDlg Class

CCraneWowDlg(CWnd* pParent = NULL, int newID = 0);

This function is the constructor for the CCraneWowDlg class. pParent is a pointer to the master dialog of the program; this is assigned to myMaster. newID is a unique identifying number given to this particular control dialog, and is assigned to m_DlgID. This function also sets m_Started to false and creates a CHistoDlg for this control dialog.

void StartData();

This function simply calls OnStart(), and is available to the master dialog for the "StartAll" button.

void SetMaster(CWnd* ms)

This function sets myMaster to ms.

int GetHisto(long *putHistoHere)

This function takes a pointer to a 2048-value array of longs (long ints). If data collection for this control dialog is started, it will populate the array with the most recent completed histogram data. The number of counts in each bin is recorded with the 0th channel being in the 0th array value. Note that the labels for each energy bin are not included by this function. See GetEnergy(double *).

double GetEnergy(double *putEnergyHere)

This function places an array of 2048 double values at the location given by *putEnergyHere. These are the labels for the energy bins. See GetHisto(long *).

double GetTime()

This simply returns the beginning time, in seconds, of the most recent completed histogram. See startH.

bool GetStarted()

Returns m_Started, which is whether or not data collection is running.

bool GetIsMaster()

Returns isMaster, which is true if the detector controlled by the dialog is set to SyncMaster gating mode.

bool SetAllowed()

Sets allowCorrection to true. This allows the automatic reset to function. See GetData().

bool SetDisallowed()

Sets allowCorrection to false. This allows the automatic reset to function. See GetData().

bool IsCalValid()

Returns true if the energy calibration of the digiBASE-E is valid, i.e. if the number of factors in the energy calibration polynomial is not zero.

UINT MakeHistoTxt(long *theData, long channelNumber, CString filename, double *theLabels)

This will create a comma-delimited file of a histogram provided, with labels, using the filename supplied. theData should be a pointer to the first element in an array of longs, containing the bin counts for each channel. channelNumber is the number of channels (usually 2048). filename is a string containing the file name desired. If no path is specified, the file will appear in the working directory of the program. theLabels is a pointer to the first element in an array of doubles. These are the labels that will be attached to the energy channels, in the same order. The file is of the format “label,counts\n” with one entry for each channel.

CUMCBIDROP m_DlistCtl

This is the control for the dropdown list of detectors. Its function is supplied by the ORTEC background processes.

CUMCBICONN m_ConnCtl

This is the control for communication with the detectors. Using the address supplied by the dropdown control, this control sends commands and receives data from the digiBASE-Es. The pertinent controls are GetRawData() and Comm(LPCSTR). Send will send commands as listed in the digiBASE-E manual. If there is a response, it is given as the return value of Comm(LPCSTR).

HICON m_hIcon

This is the icon for the program window.

int m_DlgID

This is the unique ID given to each CraneWow window. It is given by the master dialog at spawn time and used to ensure unique event identification among the windows. This allows each window to be started and stopped independently.

virtual BOOL OnInitDialog()

This function is automatically run on spawn time for each dialog. It sets up the title bar and other cosmetic objects, and places default values in each text box. There is no need to call this manually.

afx_msg …

These message handlers are invoked when buttons are pressed or other messages are received. The events related to each function can be found in the section marked BEGIN_MESSAGE_MAP in the CraneWowDlg section of CraneWowDlg.cpp.

void MakeLabels(double* theLabels, LISHDR* theHeader, int channels)

This function takes a .dat file header, already made, and extracts the calibration information. It then constructs energy bin labels and uses them to populate an array. theLabels is a pointer to the beginning of an array of doubles. theHeader is a pointer to the previously-constructed .dat header. channels is the number of channels to be labeled (usually 2048). These labels are later used to create the histogram output files.

void ClearResponse(void)

This function clears the response box of all text.

void UpdateInfo(void)

This function updates the information displayed in the window, including check boxes and radio buttons. This is automatically done during data collection once per second, as well as after sending commands. This is a somewhat intensive process, as each query must be responded to one at a time by the digiBASE-E. Excessive use can exacerbate a FIFO overflow problem, causing it to occur more readily.

void MakeHeader(LISHDR *myHeader)

This function constructs a LISHDR and places it at the location specified by myHeader. This function must gather several pieces of information from the digiBASE-E itself, as well as the computer. LISHDRs include a comment section as well. In this case, the comment is taken to be data comment as stored by MAESTRO/GammaVision.

UINT GetData()

This function executes one loop of GetRawData(), extracting from the VARIANT, writing to the .dat file, and writing out histogram files as needed. This function is called by the OnEventTimer() message handler, which executes every TIMERINTERVAL ms (normally 3). This can sometimes take much longer than 3 ms, especially if there is a hang in the data retrieval.

CFile m_DataFile

This is a CFile object dealing with the .dat file. The .dat file is continuously updated with the data stream, and is therefore a complete record of all data coming in from the digiBASE-E. In the event of a crash or loss of power, the data file will still be valid, as there is no footer and no header information that needs to be modified.

CString m_FilePath

This is the path to the file that is manipulated by m_DataFile, minus the extension. Due to the way multiple files are created at runtime, m_FilePath is actually a beginning string to the full file path of each file. A suffix and extension are added automatically by the functions that create those files to get the actual file path.

double m_iSize

This is the size, in seconds, of each histogram created on-the-fly. Collected from user input. Care must be taken so as not to cause overlap of more than two histograms, which will cause shortened histograms as the program will only track two at once. This means that m_iSize must not be more than twice m_iEvery.

double m_iEvery

This is the frequency, in seconds, of every histogram created on-the-fly. Collected from user input. Care must be taken so as not to cause time overlap of more than two histograms, which will cause shortened histograms as the program will only track two at once. This means that m_iEvery must not be less than half of m_iSize.

double m_iMax
This is the time, in seconds, to run data collection. Collected from user input. After this time has elapsed (as determined by the computer) collection will be automatically stopped and all files will be closed. If this value is zero or less, collection will continue indefinitely.

CString m_Address
This is the address of the detector being controlled by the window. Each detector has an address, in the form of a string, associated with it in the MCBIO server master list. This address is needed to specify which detector is to receive commands or polling.

CWnd *myMaster
This is a pointer to the master dialog box of the program.

bool m_Started
This flag is set true if data collection is ongoing.

CTime m_StartTime
This is the time at which data collection was started.

CTime m_StopTime
This is the time at which data collection is set to be terminated.

long m_EventsProcessed
This is the total number of four-byte words processed by a particular CraneWow window. It is a combination of detection events and all time stamps, including real time, live time, and sync time stamps. If this number does not increase, no data is being retrieved from the digiBASE-E.

long *pHist[NUMHIST]
*pHist is an array of histograms that are built on-the-fly. It is these histograms that, when finished, are written to disk as the .csv files. 2 is the next one being built, a possible overlap, with 1 being the current focus and next to stop, and 0 being finished.

long syncTimes[NUMHIST]
syncTimes[] is an array containing the latest sync time noted at the time a histogram (in *pHist[]) was started.

double startH[NUMHIST]
startH[] is an array containing the start time for each histogram in *pHist[]. If it is past this time and before the time in stopH[], counts are added to the bins in *pHist[].

double stopH[NUMHIST]
stopH[] is an array containing the stop time for each histogram in *pHist[]. If it is before this time and after the time in startH[], counts are added to the bins in *pHist[]. When the time listed here is finished, the indices in *pHist[] and associated arrays are decremented, the histogram is written to disk if needed, the oldest histogram is deleted, and a new, blank histogram is created.

DWORD m_dwUpdateTickCount

This is used to track the number of system ticks since the last call of UpdateInfo() by the event timer.

bool useSync

This is set to true if the digiBASE-E is set to Sync or SyncMaster gate modes.

long theHistoTime

This is the latest real time stamp read by the program from the data stream, in units of .1s.

long histNumber

This is the number of histograms so far completed by the program.

long m_channelNumber

This is the number of channels available to the device (usually 2048)

LISHDR m_Header

This is the header created for the .dat file by MakeHeader(LISHDR). It contains the data comment, serial number, energy calibration, and a few other data points. See RecTypes2.h for details.

double m_Labels[2048]

These are the labels created for the histograms by MakeLabels. These are calculated by using the energy calibration data stored in the header, as pulled from the digiBASE-E.