kromek
detect image identify

SPECT-DM System API

Version v2.5

Contents

Purpose

This document explains how to setup your software to use the API and lists all functions in the SPECT-DM system API.

API usage

Integration

To integrate the SPECT-DM system API into your own software follow the two steps below.

1. Add the library libSpectDMSharedLib42.a to your project.

   Different IDE's will have different ways of doing this, consult your IDE's help for information on how to do this.

2. Include the spectdmdll.h file found in the SpectDM_includes directory in the install.

After completing these two steps your software is ready to use the SPECT-DM system API.

General usage

The API contains many functions returning a bool, when the return value is false you can find out why using GetLastError().

e.g.

```
if(SpectDMDll::Initialize())
{
  // Actions on successful Initialize() call
}
else
{
  // Report error using SpectDMDll::GetLastError()
}
```

Some of the functions in the API make use of enums, when using Get functions returning an enum you typically have to initialise the variable as some Get functions return a bool. Where this is the case, enums in the API have an undefined option to initialise a variable of that type prior to calling onto the Get function.

e.g.

```
GMCathodeMode l_GMCathMode = GMCathMode_Undefined;

if(SpectDMDll::GetGMCathodeMode(&l_GMCathMode))
{
  // Actions on successful GetGMCathodeMode() call
}
else
{
  // Report error using SpectDMDll::GetLastError()
}
```

Functions

System Connection Functions

Function name	void SetHostIPAddress(const char* hostIPAddress)
Additional notes	Informs the API of the IP address of the computer connected to the SPECT-DM device.

Function name	void SetCameraIPAddress(const char* cameraIPAddress)
Additional notes	Informs the API of the IP address of the SPECT-DM device.

Function name	std::string GetCameraIPAddress()
Additional notes	Retrieves the IP address of the SPECT-DM device that was set using SetCameraIPAddress().

System Functions

Function name	bool Initialize()
Additional notes	This function connects your machine to the SPECT-DM device. The Host IP and Camera IP must have been set prior to calling this function.

Function name	bool IsSystemInitialized()
Additional notes	Returns true if Initialize() call succeeded.

Function name	bool StartPhotonCollection(int collectionTimeInSeconds = -1)
Additional notes	Prompts the device to start collecting photons. A collection time limit can be provided in seconds which will cause the collection to end once the time limit has been reached. A time limit of <= 0 will result in a continuous collection that will only end when StopPhotonCollection() is called.

Function name	bool StopPhotonCollection()
Additional notes	Prompts the device to stop collecting photons.

Function name	bool IsSystemCollectingPhotons()
Additional notes	Returns the photon collection state.

Function name	bool SetDebugMode(bool enable)
Additional notes	Enables debugging signals in GBE FPGA. In GBE debug mode the data bus and all MB resources are disabled.

Function name	bool IsDebugModeActive()
Additional notes	Returns debug mode state.

Function name	bool SetMBMultiplexerAddressType(MBMultiplexerAddrType addressType)
Additional notes	Selects MB analog MUX channel for Gamma Module 1 or Gamma Module 4 ADC1 Channel 2 conversion. Valid MBMultiplexerAddrType values are: MBMultiplexerAddrType_AUX0 MBMultiplexerAddrType_AUX1 MBMultiplexerAddrType_AUX2 MBMultiplexerAddrType_AUX3 MBMultiplexerAddrType_GM0AXPK62 MBMultiplexerAddrType_GM0AXPK63 MBMultiplexerAddrType_EMKO_IN MBMultiplexerAddrType_MB_TEMP MBMultiplexerAddrType_REF2V MBMultiplexerAddrType_S3_3V MBMultiplexerAddrType_P2_5V MBMultiplexerAddrType_A2_5V MBMultiplexerAddrType_D2_5V MBMultiplexerAddrType_F2_5V MBMultiplexerAddrType_F1_5V MBMultiplexerAddrType_S5V

Function name	MBMultiplexerAddrType GetMBMultiplexerAddressType()
Additional notes	Returns MB multiplexer address type

Function name	bool SetPacketTransferRate(int transferRate)
Additional notes	Sets the maximum data transfer rate per second from SPECT-DM device to host computer. transferRate argument must be between 1000 and 63000 and must be divisible by 1000.

Function name	int GetPacketTransferRate()
Additional notes	Returns packet transfer rate

Function name	bool SetSysTokenType(SysTokenType sysTokenType)
Additional notes	Valid SysTokenType values are: SysTokenType_DaisyChain SysTokenType_GM3Only

	SysTokenType_DaisyChain is the default. SysTokenType_GM3Only is only used for SPECT-DM devices containing one Gamma Module plugged into GM3 slot.

Function name	SysTokenType GetSysTokenType()
Additional notes	Returns sys token type

Function name	bool SaveConfiguration(const char* filename)
Additional notes	Saves your current SPECT-DM configuration and any collected packets to filename. filename must have ".dat" file extension.

Function name	bool LoadConfiguration(const char* filename)
Additional notes	Loads data from filename and automatically sets up the SPECT-DM device for you. Please check the following before using this function: - filename must have ".dat" file extension. - The IP address of the SPECT-DM device you are connecting to is the same as that which the file was saved under. The SaveConfiguration() function saves the SPECT-DM IP address to allow automatic connection and setup on calling this function. - SetHostIPAddress() has been called prior to calling this function. If the number of GMs in the configuration file match that in the SPECT-DM device then the configuration of the loaded GMs will automatically be copied to the GMs in the SPECT-DM device. If there is a mismatch in GM counts here then you will have to use the CopyLoadedGMToGM() function to setup GMs in the SPECT-DM system from GMs loaded in. If you have called SetSysStatusFunction(), this will report number of GMs in SPECT-DM system and number of GMs loaded in.

Function name	bool SaveCollection(const char* filename)
Additional notes	Saves data collected from the SPECT-DM device to file. filename must have ".csv" file extension.

Function name	bool StartSys()
Additional notes	Initiates SPECT-DM system power-up. If you call SetSysStatusFunction() prior to calling this function then you will see the output from the SPECT-DM device.

Function name	bool StopSys()
Additional notes	Powers off the SPECT-DM system.

Function name	bool Disconnect()
Additional notes	Disconnects the host computer from the SPECT-DM device.

Function name	bool GetGBEFirmwareVersion(int * version)
Additional notes	Returns the firmware version of the GBE. The function populates an int passed to the function via a pointer.

Function name	int GetNoOfGMs()
Additional notes	Returns the number of GMs in the device that have powered up successfully during SPECT-DM system power-up.

Function name	int GetNoOfLoadedGMs()
Additional notes	Returns the number of GMs loaded after a call to LoadConfiguration().

Function name	bool CopyLoadedGMToGM(int loadedGM, int GM)
Additional notes	This function is to be used when the number of GMs in the SPECT-DM system doesn't match the number of GMs loaded in from a configuration file. This function allows the user to copy the configuration of loaded GMs to GMs in the SPECT-DM system.

Function name	void Close()
Additional notes	This function should be called when you are finished using the SPECT-DM system, after StopSys() and Disconnect(). The function ensures that resources used by the API are released safely.

Gamma Module Functions

If GMUpdateType is GMUpdateType_SingleGM, set functions will set data on the active GM.

If GMUpdateType is GMUpdateType_Broadcast, set functions will set data on every GM in the device.

The active GM must be set when calling Get functions below.

Gamma Module Setup Functions

Function name	bool SetGMUpdateType(GMUpdateType GMUpdateType)
Additional notes	Sets the update type to be used when calling GM Set functions. You can configure one GM at a time or broadcast settings to all GMs in the SPECT-DM system. GMUpdateType valid values are: GMUpdateType_SingleGM GMUpdateType_Broadcast Calling SetActiveGM() automatically sets GMUpdateType_SingleGM.

Function name	GMUpdateType GetGMUpdateType()
Additional notes	Returns the GM update type

Function name	bool SetDefaultGM(int GM_ID)
Additional notes	The default GM is set on a GM during the SPECT-DM system power-up. This function is provided in case GMs were not installed in an array.

Function name	bool GetDefaultGM(int* GM_ID)
Additional notes	Function returns the default GM of the active GM. The function populates an int passed to the function via a pointer.

Function name	bool SetActiveGM(int GM_ID)
Additional notes	Sets the GM that you want to configure by Set functions below. Calling this function automatically sets the GM update type to GMUpdateType_SingleGM.

Function name	int GetActiveGM()
Additional notes	Returns the ID of the active GM that was set using SetActiveGM().

Function name	bool IsActiveGMSet()
Additional notes	-

Function name	bool GetGMStatus(GMStatus* status)
Additional notes	GM Statuses returned: GMStatus_Idle - Indicates that the GM is not collecting photons. GMStatus_ASICLoadError - Indicates CRC error during ASIC calibration data and settings reload from EEPROM. GMStatus_FIFOFull - Indicates that packets could get lost or be returned incomplete. The function populates a GMStatus object passed to the function via a pointer.

Function name	bool GetGMFirmwareVersion(int * version)
Additional notes	Returns the firmware version of the Active GM. The function populates an int passed to the function via a pointer.

Function name	bool SetGMOptions(GMOption options)
Additional notes	GMOption values are: GMOption_None GMOption_DisablePhotonCollect (DIS_PACKETS) GMOption_DebugMode (GM_DEBUG) GMOption_Channel1TestMode (ASIC E1) This function allows the user to disable data collection and packet transfer from a GM (GMOption_DisablePhotonCollect), enable GM FPGA debugging circuits and simulate the ASIC (GMOption_DebugMode) and enable channel 1 test mode (GMOption_Channel1TestMode) GMOptions can be masked together like below: SetGMOptions(GMOption_DisablePhotonCollect | GMOption_DebugMode);

Function name	bool GetGMOptions(GMOption* options)
Additional notes	Returns GM options for active GM. The function populates a GMOption variable passed to the function via a pointer.

Function name	bool SetDelayTime(int delayTimeInNanoSeconds)
Additional notes	Defines delay time for ASIC readout. delayTimeInNanoSeconds argument must be between 20 and 5100 and must be divisible by 20.

Function name	bool GetDelayTime(int* delayTimeInNanoSeconds)
Additional notes	The function returns the delay time for the active GM. The function populates an int passed to the function via a pointer.

Function name	bool SetTimestampResolution(int resolutionInMicroSeconds)
Additional notes	Defines timestamp clock. resolutionInMicroSeconds argument must be between 100 and 25500 and must be divisible by 100.

Function name	bool GetTimestampResolution(int* resolutionInMicroSeconds)
Additional notes	The function returns the timestamp resolution for the active GM. The function populates an int passed to the function via a pointer.

Function name	bool SetGMADC1Channel(int channel)
Additional notes	Defines ADC1 input. channel argument must be 1 or 2.

Function name	bool GetGMADC1Channel(int* channel)
Additional notes	Returns ADC1 channel input for active GM. The function populates an int passed to the function via a pointer.

Function name	bool SetGMADC2Channel(int channel)
Additional notes	Defines ADC2 input. channel must be 1 or 2.

Function name	bool GetGMADC2Channel(int* channel)
Additional notes	Returns ADC2 channel input for active GM. The function populates an int passed to the function via a pointer.

Function name	bool ReadGMADC1(int* DAC, int* volts)
Additional notes	Returns ADC1 data. The function populates two ints passed to the function via pointers.

Function name	bool ReadGMADC2(int* DAC, int* volts)
Additional notes	Returns ADC2 data. The function populates two ints passed to the function via pointers.

Gamma Module ASIC Setup Functions

Function name	bool SetGMReadoutOptions(GMReadoutOption options)
Additional notes	This function allows the user to specify additional data to be included in a packet during photon collection. GMReadoutOption values are: GMReadoutOpt_None GMReadoutOpt_NegativeEnergy (NEGATIVE_EN) GMReadoutOpt_TimeDetect (TD_EN) GMReadoutOptions can be masked together like below: SetGMReadoutOptions (GMReadoutOpt_NegativeEnergy | GMReadoutOpt_TimeDetect);

Function name	bool GetGMReadoutOptions(GMReadoutOption* options)
Additional notes	Returns GM readout options for active GM. The function populates a GMReadoutOption variable passed to the function via a pointer.

Function name	bool SetGMReadoutMode(GMReadoutMode mode)
Additional notes	Sets ASIC readout mode. Valid GMReadoutMode values are: GMReadout_ReadAll GMReadout_SparsifiedMode

Function name	bool GetGMReadoutMode(GMReadoutMode* mode)
Additional notes	Returns readout mode for active GM. The function populates a GMReadoutMode variable passed to the function via a pointer.

Function name	bool SetGMCathodeMode(GMCathodeMode mode)
Additional notes	Defines cathode mode. Depending on the selected mode, the GM FPGA drives a different amount of CK clocks for the ASIC. Valid GMCathodeMode values are: GMCathMode_Unipolar GMCathMode_MultiThreshold GMCathMode_Bipolar

Function name	bool GetGMCathodeMode(GMCathodeMode* mode)
Additional notes	Returns the cathode mode for the active GM. The function populates a GMCathodeMode variable passed to the function via a pointer.

Gamma Module Pulser Setup Functions

Function name	bool SetGMPulserFrequency(GMPulserFrequency freq)
Additional notes	This function sets the ASIC signals timing which trigger the pulser inside the ASIC. Valid GMPulserFrequency values are: GMPulserFreq_100Hz GMPulserFreq_1kHZ GMPulserFreq_10kHZ GMPulserFreq_100kHZ

Function name	bool GetGMPulserFrequency(GMPulserFrequency* freq)
Additional notes	This function returns the pulser frequency for the active GM. The function populates a GMPulserFrequency variable passed to the function via a pointer.

Function name	bool SetGMPulserOptions(GMPulserOption options)
Additional notes	This function allows the user to set the test pulse generating signals for the ASIC anodes and/or cathodes. GMPulserOption values are: GMPulserOpt_None GMPulserOpt_Anode (A_PLSR_EN) GMPulserOpt_Cathode (C_PLSR_EN) GMPulserOptions can be masked together like below: SetGMPulserOptions(GMPulserOpt_Anode | GMPulserOpt_Cathode);

Function name	bool GetGMPulserOptions(GMPulserOption* a_options)
Additional notes	Returns the pulser options for the active GM. The function populates a GMPulserOption variable passed to the function via a pointer.

Function name	bool SetNoOfPulses(int noOfPulses)
Additional notes	Defines number of pulses generated when starting a photon collection. noOfPulses argument must be between 0 and 15. 0 = continual pulsing.

Function name	bool GetNoOfPulses(int* noOfPulses)
Additional notes	Returns number of pulses for active GM. The function returns an int passed to the function via a pointer.

Gamma Module ASIC Global Functions

Function name	bool SetASICGlobalOptions(ASICGlobalOptions options)
Additional notes	ASICGlobalOptions values are: ASICGlobal_None ASICGlobal_SingleEventMode (SSP) ASICGlobal_EnergyMultipleFiringSuppressor (SSE) ASICGlobal_Validation (EVALID) ASICGlobal_MonitorOutputs (SAUXi) ASICGlobal_RouteTempMonitorToAXPK62 (STMPM) ASICGlobal_TimingMultipleFiringSuppressor (SSET) ASICGlobal_DisableMultipleResetAcquisitionMode (DISMRS) ASICGlobal_RouteMonitorToPinTDO (SAUXTD) ASICGlobal_BufferChnl62PreAmplifierMonitorOutput (SBPAX_CHK1) ASICGlobal_BufferChnl63PreAmplifierMonitorOutput (SBPAX_CHK2) ASICGlobal_BufferPeakAndTimeDetectorOutputs (SB) ASICGlobal_BufferAuxMonitorOutput (SBA) ASICGlobal_HighGain (HG) ASICGlobalOptions can be masked together like below: SetASICGlobalOptions(ASICGlobal_SingleEventMode | ASICGlobal_Validation);

Function name	bool GetASICGlobalOptions(ASICGlobalOptions* options)
Additional notes	Returns the ASIC Global options for the active GM. The function populates an ASICGlobalOptions variable passed to the function via a pointer.

Function name	bool SetTimingChannelUnipolarGain(TimingChannelUnipolarGain gain)
Additional notes	Valid TimingChannelUnipolarGain values are: TimingChannelUnipolarGain_27mV TimingChannelUnipolarGain_81mV

Function name	bool GetTimingChannelUnipolarGain(TimingChannelUnipolarGain* gain)
Additional notes	Returns the timing channel unipolar gain for the active GM. The function populates a TimingChannelUnipolarGain variable passed to the function via a pointer.

Function name	bool SetASICReadoutMode(GMASICReadoutMode mode)
Additional notes	The data acquisition system: (i) Uses a fast clock to identify channels with events above threshold. (ii) Readout above threshold channels and their selected neighbours. (iii) Initiates a third sparse read cycle to extract data for negative peaks. Valid GMASICReadoutMode values are: GMASICReadout_NormalSparsified GMASICReadout_EnhancedSparsified

Function name	bool GetASICReadoutMode(GMASICReadoutMode* mode)
Additional notes	Returns the ASIC readout mode for the active GM. The function populates a GMASICReadoutMode variable passed to the function via a pointer.

Function name	bool SetChannelGain(ChannelGain gain)
Additional notes	Valid ChannelGain values are: ChannelGain_20mV ChannelGain_60mV

Function name	bool GetChannelGain(ChannelGain* gain)
Additional notes	Returns the channel gain for the active GM. The function populates a ChannelGain variable passed to the function via a pointer.

Function name	bool SetCathodeTestModeInput(TestModeInput input)
Additional notes	Cathode Channel Input for Test Mode. Valid TestModeInput values are: TestModeInput_Step TestModeInput_Ramp

Function name	bool GetCathodeTestModeInput(TestModeInput* input)
Additional notes	Returns the cathode test mode input for the active GM. The function populates a TestModeInput variable passed to the function via a pointer.

Function name	bool SetInternalLeakageCurrentGenerator(InternalLeakageCurrentGenerator currGen)
Additional notes	Valid InternalLeakageCurrentGenerator values are: InternalLeakageCurrentGenerator_60pA InternalLeakageCurrentGenerator_0A

Function name	bool GetInternalLeakageCurrentGenerator(InternalLeakageCurrentGenerator* currGen)
Additional notes	Returns the internal leakage current generator for the active GM. The function populates an InternalLeakageCurrentGen variable passed to the function via a pointer.

Function name	bool SetAnodeTestPulseEdge(TestPulseEdge edge)
Additional notes	Valid TestPulseEdge values are: TestPulseEdge_InjectNegCharge TestPulseEdge_InjectPosAndNegCharge

Function name	bool GetAnodeTestPulseEdge(TestPulseEdge* edge)
Additional notes	Returns the anode test pulse edge for the active GM. The function populates a TestPulseEdge variable passed to the function via a pointer.

Function name	bool SetPeakDetectTimeout(ChannelType type, int timeoutInMicroSeconds)
Additional notes	ChannelType argument must be one of ChannelType_Anode or ChannelType_Cathode. Valid values for timeoutInMicroSeconds change based on whether ASICGlobal_HighGain (HG) is set or not. When HG = 0, valid values are 1, 2, 4 and 8 When HG = 1, valid values are 4, 8, 16 and 32

Function name	bool GetPeakDetectTimeout(ChannelType type, int* timeoutInMicroSeconds)
Additional notes	ChannelType argument must be one of ChannelType_Anode or ChannelType_Cathode. The function populates an int passed to the function via a pointer.

Function name	bool SetTimeDetectRampLength(ChannelType type , int rampLengthInMicroSeconds)
Additional notes	ChannelType argument must be one of ChannelType_Anode or ChannelType_Cathode. Valid values for rampLengthInMicroSeconds change based on whether ASICGlobal_HighGain (HG) is set or not. When HG = 0, valid values are 1, 2, 3 and 4 When HG = 1, valid values are 4, 8, 12 and 16

Function name	bool GetTimeDetectRampLength(ChannelType type , int* rampLengthInMicroSeconds)
Additional notes	ChannelType argument must be one of ChannelType_Anode or ChannelType_Cathode. The function populates an int passed to the function via a pointer.

Function name	bool SetPeakingTime(ChannelType type, float peakingTimeInMicroSeconds)
Additional notes	ChannelType argument must be one of ChannelType_Anode or ChannelType_Cathode. Valid values for peakingTimeInMicroSeconds change based on whether ASICGlobal_HighGain (HG) is set or not. When HG = 0, valid values are 0.25, 0.5, 1 and 2 When HG = 1, valid values are 1.5, 3, 6 and 12

Function name	bool GetPeakingTime(ChannelType type, float* peakingTimeInMicroSeconds)
Additional notes	ChannelType argument must be one of ChannelType_Anode or ChannelType_Cathode. The function populates a float passed to the function via a pointer.

Function name	bool SetCathodeTimingChannelsShaperPeakingTime(TimingChannelsShaperPeakingTime peakingTime)
Additional notes	Valid TimingChannelsShaperPeakingTime values are: TimingChannelShaperPeakingTime_100nS TimingChannelShaperPeakingTime_200nS TimingChannelShaperPeakingTime_400nS TimingChannelShaperPeakingTime_800nS

Function name	bool GetCathodeTimingChannelsShaperPeakingTime(TimingChannelsShaperPeakingTime* peakingTime)
Additional notes	The function populates a TimingChannelsShaperPeakingTime variable passed to the function via a pointer.

Function name	bool SetCathodeTimingChannelsSecondaryMultiThresholdsDisplacementStep(int displacementStep)
Additional notes	displacementStep must be between 0 and 15, each step represents 6.25mV and displacement runs from -100 to -6.25.

Function name	bool GetCathodeTimingChannelsSecondaryMultiThresholdsDisplacementStep(int* displacementStep)
Additional notes	The function populates an int passed to the function via a pointer.

Function name	bool SetMultipleFiringSuppressTime(MultipleFiringSuppressionTime suppressTime)
Additional notes	Valid MultipleFiringSuppressionTime values are: MultipleFiringSuppressionTime_62_5nS MultipleFiringSuppressionTime_125nS MultipleFiringSuppressionTime_250nS MultipleFiringSuppressionTime_600nS

Function name	bool GetMultipleFiringSuppressTime(MultipleFiringSuppressionTime* suppressTime)
Additional notes	The function populates a MultipleFiringSuppressionTime variable passed to the function via a pointer.

Function name	bool SetTestPulseStep(ChannelType type, int testPulseStep)
Additional notes	ChannelType argument must be one of ChannelType_Anode or ChannelType_Cathode. If ChannelType_Cathode, if Cathode Test Mode Input is 0, adjusts step, if 1, adjusts ramp slope. testPulseStep must be between 0 and 1023, each step represents ~1.85mV, test pulse runs from 0 to 1.85V.

Function name	bool GetTestPulseStep(ChannelType type, int* testPulseStep)
Additional notes	ChannelType argument must be one of ChannelType_Anode or ChannelType_Cathode. The function populates an int passed to the function via a pointer.

Function name	bool SetChannelThresholdStep(ChannelThresholdType type, int thresholdStep)
Additional notes	Valid ChannelThresholdType values are: ChannelThresholdType_CathodeTimingPrimaryMultiThresholdBiPolar (PD) ChannelThresholdType_CathodeTimingUnipolar (PC) ChannelThresholdType_CathodeEnergy (PB) ChannelThresholdType_AnodeNegativeEnergy (PAn) ChannelThresholdType_AnodePositiveEnergy (PA) thresholdStep must be between 0 and 1023, each step represents ~1.85mV, threshold runs from 0 to 1.85V.

Function name	bool GetChannelThresholdStep(ChannelThresholdType type, int* thresholdStep)
Additional notes	Valid ChannelThresholdType values are: ChannelThresholdType_CathodeTimingPrimaryMultiThresholdBiPolar (PD) ChannelThresholdType_CathodeTimingUnipolar (PC) ChannelThresholdType_CathodeEnergy (PB) ChannelThresholdType_AnodeNegativeEnergy (PAn) ChannelThresholdType_AnodePositiveEnergy (PA) The function populates an int passed to the function via a pointer.

Function name	bool SetCathodeTestClockType(CathodeTestClockType clockType)
Additional notes	Valid CathodeTestClockType values are: CathodeTestClockType_CopyAnodeTestClock CathodeTestClockType_ArrivesOnSDI_NSDI

Function name	bool GetCathodeTestClockType(CathodeTestClockType* clockType)
Additional notes	The function populates a CathodeTestClockType variable passed to the function via a pointer.

Function name	bool SetTimingChannelBiPolarGain(TimingChannelBiPolarGain gain)
Additional notes	Used in conjunction with Timing Channel unipolar gain for additional bipolar gain. If unipolar gain is 27mV then bipolar gain must be 21mv or 55mV. If unipolar gain is 81mv then bipolar gain must be 62mV or 164mV. Valid TimingChannelBiPolarGain values are: TimingChannelBiPolarGain_21mV TimingChannelBiPolarGain_55mV TimingChannelBiPolarGain_62mV TimingChannelBiPolarGain_164mV

Function name	bool GetTimingChannelBiPolarGain(TimingChannelBiPolarGain* gain)
Additional notes	The function populates a TimingChannelBiPolarGain variable passed to the function via a pointer.

Function name	bool SetCathodeChannelInternalLeakageCurrentGenerator(int channelNumber , CathodeChannel::InternalLeakageCurrentGenerator currGen)
Additional notes	channelNumber must be 1 or 2. Valid CathodeChannel:: InternalLeakageCurrentGenerator values are: CathodeChannel:: InternalLeakageCurrentGenerator_350pA CathodeChannel:: InternalLeakageCurrentGenerator_2nA The reason you have to precede InternalLeakageCurrentGenerator with CathodeChannel:: is because this type exists in the CathodeChannel namespace due to there already being an InternalCurrentLeakage type with different values.

Function name	bool GetCathodeChannelInternalLeakageCurrentGenerator(int channelNumber , CathodeChannel::InternalLeakageCurrentGenerator* currGen)
Additional notes	channelNumber must be 1 or 2. The function populates a CathodeChannel::InternalLeakageCurrentGenerator variable passed to the function via a pointer.

Function name	bool SetAnalogOutputToMonitor(AnalogOutput output)
Additional notes	Valid AnalogOutput values are: AnalogOutput_NoFunction AnalogOutput_Baseline AnalogOutput_Temperature Other AnalogOutput elements exist however these are used internally when setting analog outputs that require more information, see functions following GetAnalogOutputToMonitor() below. You can only monitor one output at a time so this function overwrites any value set via SetAnodeChannelToMonitor(), SetCathodeEnergyTimingToMonitor() or SetDACToMonitor().

Function name	bool GetAnalogOutputMonitored(AnalogOutput* output)
Additional notes	The function populates an AnalogOutput variable passed to the function via a pointer.

Function name	bool SetAnodeChannelToMonitor(int channel)
Additional notes	channel argument must be between 1 and 128. You can only monitor one output at a time so this function overwrites any value set via SetAnalogOutputToMonitor(), SetCathodeEnergyTimingToMonitor() or SetDACToMonitor().

Function name	bool GetAnodeChannelMonitored(int* channel)
Additional notes	Returns the anode channel monitored if SetAnodeChannelToMonitor() has been called. The function populates an int passed to the function via a pointer.

Function name	bool SetCathodeEnergyTimingToMonitor(CathodeEnergyTiming energyTiming)
Additional notes	Valid CathodeEnergyTiming values are: CathodeEnergyTiming_Channel1Energy CathodeEnergyTiming_Channel1Timing CathodeEnergyTiming_Channel2Energy CathodeEnergyTiming_Channel2Timing You can only monitor one output at a time so this function overwrites any value set via SetAnalogOutputToMonitor(), SetAnodeChannelToMonitor() or SetDACToMonitor().

Function name	bool GetCathodeEnergyTimingMonitored(CathodeEnergyTiming* energyTiming)
Additional notes	Returns the cathode energy/timing monitored if SetCathodeEnergyTimingToMonitor() has been called. The function populates a CathodeEnergyTiming variable passed to the function via a pointer.

Function name	bool SetDACToMonitor(DACS dac)
Additional notes	This is the function dedicated to setting the DAC analog output. Valid DACS values are: DACS_AnodeEnergyThreshold DACS_AnodeEnergyTransient DACS_CathodeEnergyThreshold DACS_CathodeTimingUnipolarThreshold DACS_CathodeTimingFirstMultiThreshold DACS_CathodeTimingSecondMultiThreshold DACS_CathodeTimingThirdMultiThreshold DACS_AnodeTestSignal DACS_CathodeTestSignal You can only monitor one output at a time so this function overwrites any value set via SetAnalogOutputToMonitor(), SetAnodeChannelToMonitor() or SetCathodeEnergyTimingToMonitor().

Function name	bool GetDACMonitored(DACS* dac)
Additional notes	Returns the DAC monitored if SetDACToMonitor() has been called. The function populates a DACS variable passed to the function via a pointer.

Gamma Module ASIC Channel Functions

Pre-Channel setting functions

To update a channel in the ASIC the API requires the following information:

- Channel update type – Would you like to update one channel or all?
- Channel type – Anode or Cathode?
- Channel number – only needed if channel update type is single channel.

The related functions below explain more about valid value ranges.

Function name	void SetChannelUpdateType(ChannelUpdateType channelUpdateType)
Additional notes	Valid ChannelUpdateType values are: ChannelUpdateType_SingleChannel ChannelUpdateType_Broadcast

Function name	void SetActiveChannelType(ChannelType channelType)
Additional notes	Valid ChanneType values are: ChannelType_Anode ChannelType_Cathode

Function name	bool SetActiveChannel(int channel)
Additional notes	Valid channel values vary based on what the active channel type is: if ChannelType_Anode then valid channel range is 1-128 if ChannelType_Cathode then valid values are 1 or 2. This function does not need to be called if ChannelUpdateType is set to ChanneUpdateType_Broadcast.

Channel functions

Function name	bool MaskChannel(bool mask)
Additional notes	Channel Mask (shaper shut-down)

Function name	bool IsChannelMasked(ChannelType type, int channelNumber, bool* mask)
Additional notes	type must be one of ChannelType_Anode or ChannelType_Cathode. If type is ChannelType_Anode, channelNumber must be between 1 and 128, if ChannelType_Cathode, channelNumber must be 1 or 2. The function populates a bool passed to the function via a pointer.

Function name	bool EnableChannelTestCapacitor(bool enable)
Additional notes	Test Capacitor 200fF.

Function name	bool IsChannelTestCapacitorEnabled(ChannelType type, int channelNumber , bool* enable)
Additional notes	type must be one of ChannelType_Anode or ChannelType_Cathode. If type is ChannelType_Anode, channelNumber must be between 1 and 128, if ChannelType_Cathode, channelNumber must be 1 or 2. The function populates a bool passed to the function via a pointer.

Function name	bool MonitorAnodeSignal(Signal signal)
Additional notes	Valid Signal values are: Signal_Positive Signal_Negative

Function name	bool GetAnodeSignalMonitored(int channelNumber, Signal* signal)
Additional notes	channelNumber must be between 1 and 128. The function populates a Signal variable passed to the function via a pointer.

Function name	bool SetCathodeChannelShapedTimingSignal (CathodeChannel::ShapedTimingSignal signal)
Additional notes	Valid CathodeChannel::ShapedTimingSignal values are: CathodeChannel::ShapedTimingSignal_Unipolar CathodeChannel::ShapedTimingSignal_Bipolar Providing that a cathode channel is being monitored (SetCathodeEnergyTimingToMonitor ()), this function controls what shaped timing signal is available on the monitor.

Function name	bool GetCathodeChannelShapedTimingSignal (int channelNumber, CathodeChannel::ShapedTimingSignal* signal)
Additional notes	channelNumber must be 1 or 2. The function populates a CathodeChannel::ShapedTimingSignal variable passed to the function via a pointer.

Function name	bool SetCathodeChannelTimingMode(CathodeChannel::TimingMode mode)
Additional notes	Valid CathodeChannel::TimingMode values are: CathodeChannel::TimingMode_Unipolar CathodeChannel::TimingMode_MultiThreshold_Unipolar CathodeChannel::TimingMode_Bipolar_Unipolar

Function name	bool GetCathodeChannelTimingMode(int channelNumber, CathodeChannel::TimingMode* mode)
Additional notes	channelNumber must be 1 or 2. The function populates a CathodeChannel::TimingMode variable passed to the function via a pointer.

Function name	bool SetCathodeTimingChannelTrimStep (CathodeTimingChannelType channelType, int trimStep)
Additional notes	Valid CathodeTimingChannelType values are: CathodeTimingChannelType_FirstMultiThresholdBiPolar (DAM) CathodeTimingChannelType_SecondMultiThreshold (DAMb) CathodeTimingChannelType_ThirdMultiThreshold (DAMc) CathodeTimingChannelType_Unipolar (DAU) trimStep must be between 0 and 15, each step represents 3.5mV and trim runs from -52.5 to 0.

Function name	bool GetCathodeTimingChannelTrimStep (int channelNumber, CathodeTimingChannelType channelType , int* trimStep)
Additional notes	channelNumber must be 1 or 2. Valid CathodeTimingChannelType values are: CathodeTimingChannelType_FirstMultiThresholdBiPolar (DAM) CathodeTimingChannelType_SecondMultiThreshold (DAMb) CathodeTimingChannelType_ThirdMultiThreshold (DAMc) CathodeTimingChannelType_Unipolar (DAU) The function populates an int passed to the function via a pointer.

Function name	bool SetChannelPositivePulseThresholdTrimStep(int trimStep)
Additional notes	trimStep must be between 0 and 31, each step represents 2mV and positive pulse threshold trim runs from -62 to 0.

Function name	bool GetChannelPositivePulseThresholdTrimStep(ChannelType channelType, int channelNumber, int* trimStep)
Additional notes	type must be one of ChannelType_Anode or ChannelType_Cathode. If type is ChannelType_Anode, channelNumber must be between 1 and 128, if ChannelType_Cathode, channelNumber must be 1 or 2. The function populates an int passed to the function via a pointer.

Function name	bool SetAnodeChannelNegativePulseThresholdTrimStep(int trimStep)
Additional notes	trimStep must be between 0 and 31, each step represents 2mV and negative pulse threshold trim runs from -62 to 0.

Function name	bool GetAnodeChannelNegativePulseThresholdTrimStep(int channelNumber, int* trimStep)
Additional notes	channelNumber must be between 1 and 128. The function populates an int passed to the function via a pointer.

Collected data functions

Function name	int GetNoOfCollectedPackets()
Additional notes	Returns how many packets have been collected.

Function name	bool GetPacketData(int packetNo, PacketData data, int* returnedData)
Additional notes	PacketData argument can be one of: PacketData_GMNo PacketData_Timestamp PacketData_PhotonCount PacketData_ActualPhotonCount PacketData_PhotonCount is the photon count reported in the packet received from the SPECT-DM device, PacketData_ActualPhotonCount is the number of photons counted by the API. The function populates an int passed to the function via a pointer.

Function name	GetPhotonData(int packetNo, int photonNo, PhotonData data, int* returnedData)
Additional notes	PhotonData can be one of: PhotonData_Coordinate PhotonData_Energy PhotonData_EnergyPosEvent PhotonData_TimeDetect PhotonData_TimeDetectPosEvent The function populates an int passed to the function via a pointer.

Additional functions

Function name	std::string GetVersion()
Additional notes	This function returns the version of the API.

Function name	bool RegWrite(int regNum, unsigned char MSB, unsigned char LSB)
Additional notes	This function is to be used to write directly to system control registers. See registers 14-16 in Appendix A1 for register layout.

Function name	bool RegRead(int regNum, unsigned char* MSB, unsigned char* LSB)
Additional notes	This function is to be used to read directly from system control regsiters. See registers 14-16 in Appendix A1 for register layout. The function populates two unsigned chars passed to the function via pointers.

Function name	bool GMRegWrite(int GMRegNum, unsigned char data)
Additional notes	This function is to be used to write directly to GM registers. See registers 18-24 in Appendix A1 for register layout.

Function name	bool GMRegRead(int GMRegNum, unsigned char* data)
Additional notes	This function is to be used to read directly from GM registers. See registers 18-24 in Appendix A1 for register layout. The function populates an unsigned char passed to the function via a pointer.

Function name	std::string GetLastError()
Additional notes	A number of the functions in the API return a bool, if this is false then you can query what caused the return to be false through this function.

Function name	void SetConnectStatusFunction(callback_function func)

Additional notes	typedef void(*callback_function)(const char*);
	
	This function allows the user to set a callback function which will be called and passed connection messages when connecting to the SPECT-DM device.
	
	The function doesn't have to be called but could be useful for diagnosing connection problems.
	
	e.g.
	
	static void StatusCallbackFunc(const char* a_Data);
	
	SpectDMDll::SetConnectStatusFunction(StatusCallbackFunc);

Function name	void SetSysStatusFunction(callback_function sysStatusFunc)
Additional notes	Argument type is the same as that used in SetConnectStatusFunction() above. This function allows the user to set a callback function which will then be called with system statuses. The function doesn't have to be called but it does provide useful information about what is going on in the device, especially during SPECT-DM system power-up after a call to StartSys(). e.g. static void StatusCallbackFunc(const char* a_Data); SpectDMDll::SetSysStatusFunction(StatusCallbackFunc);

Function name	void SetToolbarStatusFunction(callback_function toolbarStatusFunc)
Additional notes	Argument type is the same as that used in SetConnectStatusFunction(). This function allows the user to set a callback function which will then be called with a status intended for toolbars. The function doesn't have to be called but it does provide feedback on events such as configuration loading and packet collection. e.g. static void ToolbarStatusCallbackFunc(const char* a_Data); SpectDMDll::SetToolbarStatusFunction(ToolbarStatusCallbackFunc);

Function name	bool SetPacketBufferSize(int bufferSize)
Additional notes	The packet buffer is a buffer within the API that is filled with packets when the user issues StartPhotonCollection(). By default it is set to 1MB but can be set as high as 1024MB. When the buffer reaches this limit, it's contents are copied into another buffer to be decoded by the packet decoder function, the buffer is then emptied to enable more packets to be collected. bufferSize is in MB.

Function name	int GetPacketBufferSize()
Additional notes	Returns packet buffer size in MB.

Function name	bool SetCameraUpdateMode(CameraUpdateMode cameraUpdateMode)
Additional notes	The Camera Update Mode determines how often the SPECT-DM device is updated. If manual mode is set, the user needs to call functions to update the device. If automatic mode is set then the SPECT-DM device is updated everytime a Set function is called. If manual mode is set then the user needs to call the following functions: SendGBEControlData() SendGMData() SendASICGlobalData() SendASICChannelData() Valid values for cameraUpdateMode are: CameraUpdateMode_Manual CameraUpdateMode_Automatic By default, the camera update mode is CameraUpdateMode_Manual.

Function name	bool SendGBEControlData()
Additional notes	This function forces the API to send GBE control data to the SPECT-DM device, GBE control data is what is set by functions in the "*System Functions"* section of this document. If the Camera Update Mode is set to CameraModeUpdate_Automatic then the user does not need to call this function as every Set will automatically update the SPECT-DM device.

Function name	bool SendGMData()
Additional notes	This function forces the API to send GM data to the SPECT-DM device, GM data is what is set by functions in the following sections of the *Gamma Module Functions* section of this document: *Gamma Module Setup Functions* *Gamma Module ASIC Setup Functions* *Gamma Module Pulser Setup Functions* If the Camera Update Mode is set to CameraModeUpdate_Automatic then the user does not need to call this function as every Set will automatically update the SPECT-DM device.

Function name	bool SendASICGlobalData()
Additional notes	This function forces the API to send ASIC Global Data to the SPECT-DM device, ASIC global data is what is set by functions in the *Gamma Module ASIC Global Functions* section in the *Gamma Module Functions* section of this document. If the Camera Update Mode is set to CameraModeUpdate_Automatic then the user does not need to call this function as every Set will automatically update the SPECT-DM device.

Function name	bool SendASICChannelData()
Additional notes	This function forces the API to send ASIC Channel Data to the SPECT-DM device, ASIC channel data is what is set by functions in the *Gamma Module ASIC Channel Functions* section in the *Gamma Module Functions* section of this document. If the Camera Update Mode is set to CameraModeUpdate_Automatic then the user does not need to call this function as every Set will automatically update the SPECT-DM device.

Callback Methods

Function name	bool SetOperationErrorFunction(callback_function_op_str)
Additional notes	Sets a callback method that will be called when an error occurs during an operation. The type of operation and the error message string is returned. Valid Operations: OperationType_SavePackets

Function name	bool SetOperationProgressFunction(callback_function_op_int)
Additional notes	Sets a callback method that will be called to report progress on an operation. The type of operation and percentage is returned. Valid Operations: OperationType_SavePackets

Function name	bool SetOperationCompleteFunction(callback_function_op)
Additional notes	Sets a callback method that will be called when an operation completes successfully. The type of operation is returned. Valid Operations: OperationType_SavePackets OperationType_Collection

Appendices

A1. Register layout

The following tables have the following format:

RG Name	Bit	BIT NAME	Notes
RG14 PORT NUMBER. Write only	16 bit		Port number assigned for UDP data connection.
RG15 DACs Write only	16 bit	refer to DAC MAX5533 datasheets for command format	Virtual gate passing the commands to SPI DAC on MB.

RG16 **GBE CONTROL** Read and write. Resides in GBE FPGA	15	TR_RATE5	
	14	TR_RATE4	
	13	TR_RATE3	
	12	TR_RATE2	
	11	TR_RATE1	Buffered Packet Data maximum transfer rate to GBE. 1K transfers per sec resolution.
	10	TR_RATE0	
	9	MUX_ADDR3	MB multiplexer address.
	8	MUX_ADDR2	Used for self-test.Decimal Addr:
	7	MUX_ADDR1	0-3AUX0-3; 4-GM0AXPK62; 5-GM0 AXPK63; 6-EMKO_IN; 7-MB TEMP; 8-REF2V, 9S3.3V(1/2scale); 10-P2.5V; 11-A2.5V; 12-D2.5V; 13-F2.5V; 14-F1.5V; 15-S5V(1/2 scale)
	6	MUX_ADDR0	
	5	SYS_CK1	System clock speed control in Proto
	4	SYS_CK0	00-6.25MHZ,01-10MHZ, 10-12.5MHZ,11-16.667MHZ
	3	GM3 Token	H-enables Token only trough GM3 slot.Only GM3 should be installed. L- daisy-chained token
	2	GBE_DEBUG	H-disables bus and enables GBE FPGA debugging circuits
	1	COLLECT	H-start photon collection and packet data transfer
	0	PWR_ON	H-turns GM and MB supplies ON starting GM initialization

RG17	15	FRAME-H	Frame bit set H.
GBE STATUS	14		
	13	Sys_Mode2	0- Sys OFF; 1- Power Up; 2-Idle; 3-Collecting; 4-JTAG; 5-Debug
Read only	12	Sys_Mode1	
Resides in GBE FPGA	11	Sys_Mode0	
Resets to 0 after shifting to RG14 .	10	Comm_Err2	Communication Error codes.
	9	Comm_Err1	0- no errors; 1- Packet CRC Error; 3-Incomplete Packet;
	8	Comm_Err0	4- Wrong Command; 5-EEPROM error(power up load); 6- ID setup error
	7	FRAME-L	Frame bit set L.
	6		
	5	GM_Code5	
	4	GM_Code4	GM code. Indicates GM ID which produced Comm_Err on powerup or during Collection.
	3	GM_Code3	
	2	GM_Code2	Code starts with 1.
	1	GM_Code1	
	0	GM_Code0	

RG18	7		
GM Control	6		
Read and write.	5	ADC2_CHNL	defines ADC2 input
Resides in GM FPGA	4	ADC1_CHNL	defines ADC1 input
	3	ASIC_E1	H- asic channel1 test.
	2	GM_DEBUG	H-disables bus and enables GM Debug Mode with simulated FLAG. Don't install ASIC.
	1	DIS_PACKETS	H-disables photon collection and packet data transfer in particular GM.
	0	RELOAD_ASIC	H-initiates GM FPGA to reload asic settings.

RG19	7		
GM STATUS	6		
	5		
Read only.	4		
Resets after read.	3		
Resides in GM FPGA	2	FIFO FULL	H-when FIFO full
	1	ASIC LOAD ERROR	H- wrong CRC on asic load
	0	IDLE	H- when not collecting

RG20	7		
GM ID	6		
Read and write.	5	GM_ID0	
Resides in GM FPGA	4	GM_ID0	GM ID code.
	3	GM_ID0	Assigned on powerup or overwritten by the Host
	2	GM_ID0	
	1	GM_ID0	
	0	GM_ID0	

RG21	7	CATH_MODE1	
ASIC MODE	6	CATH_MODE0	00- Unipolar Mode, 01-Multi threshold, 10-Bi-polar.
	5	ASIC Global Reset	H-to toggle asic global reset mode
Read and write.	4	TD_EN	H-Enables TD conversion and TD data placed in packet.
	3	NEGATIVE_EN	H-Enables negative data readout and placed in packet.
Resides in GM FPGA	2	READ_MODE2	ASIC readout mode binary code.
	1	READ_MODE1	000- Read All; 001-Sparsified; 010-Enhanced Sparsified, 011-FLAGs Only(for count rate output)
	0	READ_MODE0	

RG22 PULSER MODE Read and write. Resides in GM FPGA	7 6 5 4 3 2 1 0	PLS_COUNT3 PLS_COUNT2 PLS_COUNT1 PLS_COUNT0 C_PLSR_EN A_PLSR_EN PLSR_FREQ1 PLSR_FREQ0	0-continual pulsing Defines amount of pulsers plus one. (Extra pulse is generated by logic). H-Enables pulser mode for Cathodes H-Enables pulser mode for anodes Pulser frequency code: 00- 100Hz; 01-1KHz; 10-10KHz; 11-100KHz

RG23 DELAY TIME Read and write. Resides in GM FPGA	7 6 5 4 3 2 1 0	DELAY_7 DELAY_6 DELAY_5 DELAY_4 DELAY_3 DELAY_2 DELAY_1 DELAY_0	Delay time binary code. Resolution is Sys_CK dependant

RG24 TIME STAMP RESOLUTION Read and write.	8 bits	TIME_ST0-7	Resolution is Sys_CK dependant

RG 25- reserved			

RG26 GM ADC1. Read only Resides on GM FPGA	Two bytes	12 LSBs used for ADC data	RD command starts ADC1 conversion.

RG27 GM ADC2. Read only. Resides on GM FPGA	Two bytes	12 LSBs used for ADC data	RD command starts ADC2 conversion.