

# SPECT-DM API Functions

---

*Version v1.1*

The Kromek logo is a registered trademark of Kromek Limited.

© Kromek Limited. All rights reserved.

# Contents

---

Purpose .....	4
Functions.....	5
System Connection Functions.....	5
System Functions .....	6
Gamma Module Functions.....	12
Gamma Module Setup Functions .....	12
Gamma Module ASIC Setup Functions .....	16
Gamma Module Pulser Setup Functions.....	18
Gamma Module ASIC Global Functions .....	20
Gamma Module ASIC Channel Functions .....	29
Collected data functions .....	33
Additional functions.....	34
Appendices.....	40
A1. Register layout.....	40

## Purpose

This document lists all of the functions in the SPECT-DM API and provides additional information where necessary.

## Functions

### System Connection Functions

No.	1
Function name	<code>void SetHostIPAddress(const char* hostIPAddress)</code>
Additional notes	Informs the API of the IP address of the computer connected to the SPECT-DM device.

No.	2
Function name	<code>void SetCameraIPAddress(const char* cameraIPAddress)</code>
Additional notes	Informs the API of the IP address of the SPECT-DM device.

No.	3
Function name	<code>std::string GetCameraIPAddress()</code>
Additional notes	Retrieves the IP address of the SPECT-DM device that was set using <code>SetCameraIPAddress()</code> .

## System Functions

No.	4
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.

No.	5
Function name	bool IsSystemInitialized()
Additional notes	Returns true if Initialize() call succeeded.

No.	6
Function name	bool StartPhotonCollection()
Additional notes	Prompts the device to start collecting photons.

No.	7
Function name	bool StopPhotonCollection()
Additional notes	Prompts the device to stop collecting photons.

No.	8
Function name	bool IsSystemCollectingPhotons()
Additional notes	Returns the photon collection state.

No.	9
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.

No.	10
Function name	bool IsDebugModeActive()
Additional notes	Returns debug mode state.

No.	11
Function name	bool SetMBMultiplexerAddressType(MBMultiplexerAddrType addressType)
Additional notes	<p>Selects MB analog MUX channel for Gamma Module 1 or Gamma Module 4 ADC1 Channel 2 conversion.</p> <p>Valid MBMultiplexerAddrType values are:</p> <p>           MBMultiplexerAddrType_GM1_AUX            MBMultiplexerAddrType_GM2_AUX            MBMultiplexerAddrType_GM3_AUX            MBMultiplexerAddrType_GM4_AUX            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         </p>

No.	12
Function name	MBMultiplexerAddrType GetMBMultiplexerAddressType()
Additional notes	Returns MB multiplexer address type

No.	13
Function name	bool SetPacketTransferRate(int transferRate)
Additional notes	<p>Sets the maximum data transfer rate per second from SPECT-DM device to host computer.</p> <p>transferRate argument must be between 1000 and 63000 and must be divisible by 1000.</p>

No.	14
Function name	int GetPacketTransferRate()
Additional notes	Returns packet transfer rate

No.	15
Function name	bool SetSysTokenType(SysTokenType sysTokenType)
Additional notes	<p>Valid SysTokenType values are:</p> <p>SysTokenType_DaisyChain SysTokenType_GM3Only</p> <p>SysTokenType_DaisyChain is the default. SysTokenType_GM3Only is only used for SPECT-DM devices containing one Gamma Module plugged into GM3 slot.</p>

No.	16
Function name	SysTokenType GetSysTokenType()
Additional notes	Returns sys token type

No.	17
Function name	bool SaveConfiguration(const char* filename)
Additional notes	<p>Saves your current SPECT-DM configuration and any collected packets to filename.</p> <p>filename must have ".dat" file extension.</p>

No.	18
Function name	bool LoadConfiguration(const char* filename)
Additional notes	<p>Loads data from filename and automatically sets up the SPECT-DM device for you.</p> <p>Please check the following before using this function:</p> <ul style="list-style-type: none"> <li>- filename must have ".dat" file extension.</li> <li>- 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.</li> <li>- SetHostIPAddress() has been called prior to calling this function.</li> </ul> <p>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.</p>



No.	19
Function name	bool SaveCollection(const char* filename)
Additional notes	<p>Saves data collected from the SPECT-DM device to file.</p> <p>filename must have “.csv” file extension.</p> <p>The file format is as follows:</p> <p>GM_ID, Timestamp, Coordinate, Energy, Positive Energy Event, Time Detect, Positive Time Detect Event</p> <p>There is one of these entries for each photon collected.</p> <p><b>Note:</b> This function makes use of the operation callbacks for progress, error and completion. The function is non-blocking, returning after doing a few error checks, and it is advised that you do not allow the user to interact with the API until the operation complete callback is received.</p>

No.	20
Function name	bool SaveBinaryCollection(const char* filename)
Additional notes	<p>Saves data collected from the SPECT-DM device to a binary file.</p> <p>filename must have “.dat” file extension.</p> <p>The format of the file is as follows:</p> <p>GM_ID – char Timestamp – short Coordinate – short Energy – short Positive Energy Event – char Time Detect – short Positive Time Detect Event – char</p> <p>There is one of these entries for each photon collected.</p>

No.	21
Function name	bool StartSys()
Additional notes	<p>Initiates SPECT-DM system power-up.</p> <p>If you call SetSysStatusFunction() prior to calling this function then you will see the output from the SPECT-DM device.</p>

No.	22
Function name	bool StopSys()
Additional notes	Powers off the SPECT-DM system.

No.	23
Function name	bool Disconnect()
Additional notes	Disconnects the host computer from the SPECT-DM device.

No.	24
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.

No.	25
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.

No.	26
Function name	int GetNoOfLoadedGMs()
Additional notes	Returns the number of GMs loaded after a call to LoadConfiguration().

No.	27
Function name	bool CopyLoadedGMTToGM(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.

No.	28
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.

No.	29
Function name	bool SetPixelMappingMode(PixelMappingMode pixelMappingMode)
Additional notes	Valid PixelMappingMode values are:  PixelMappingMode_Global PixelMappingMode_GMBased  PixelMappingMode_GMBased is the default. GM-based pixel mapping means that each GM will return a pixel number between 1 and 121 during photon collection. Global pixel mapping treats all four GMs as one and returns a pixel number between 1 and 484 (121 x 4).

No.	30
Function name	PixelMappingMode GetPixelMappingMode()
Additional notes	Returns the pixel mapping mode.

## 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

No.	31
Function name	bool SetGMUpdateType(GMUpdateType GMUpdateType)
Additional notes	<p>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.</p> <p>GMUpdateType valid values are:</p> <p>GMUpdateType_SingleGM GMUpdateType_Broadcast</p> <p>Calling SetActiveGM() automatically sets GMUpdateType_SingleGM.</p>

No.	32
Function name	GMUpdateType GetGMUpdateType()
Additional notes	Returns the GM update type

No.	33
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.

No.	34
Function name	bool GetDefaultGM(int* GM_ID)
Additional notes	<p>Function returns the default GM of the active GM.</p> <p>The function populates an int passed to the function via a pointer.</p>

No.	35
Function name	bool SetActiveGM(int GM_ID)
Additional notes	<p>Sets the GM that you want to configure by Set functions below.</p> <p>Calling this function automatically sets the GM update type to GMUpdateType_SingleGM.</p>

No.	36
Function name	int GetActiveGM()
Additional notes	Returns the ID of the active GM that was set using SetActiveGM().

No.	37
Function name	bool IsActiveGMSet()
Additional notes	-

No.	38
Function name	bool GetGMStatus(GMStatus* status)
Additional notes	<p>GM Statuses returned:</p> <p>GMStatus_Idle                      - Indicates that the GM is not collecting photons.</p> <p>GMStatus_ASICLoadError       - Indicates CRC error during ASIC calibration data and settings reload from EEPROM.</p> <p>GMStatus_FIFOFull               - Indicates that packets could get lost or be returned incomplete.</p> <p>The function populates a GMStatus object passed to the function via a pointer.</p>

No.	39
Function name	bool GetGMFirmwareVersion(int * version)
Additional notes	<p>Returns the firmware version of the Active GM.</p> <p>The function populates an int passed to the function via a pointer.</p>

No.	40
Function name	bool SetGMOptions(GMOption options)
Additional notes	<p>GMOption values are:</p> <p>GMOption_None            GMOption_DisablePhotonCollect (DIS_PACKETS)            GMOption_DebugMode (GM_DEBUG)            GMOption_Channel1TestMode (ASIC E1)            GMOption_EnablePixelMap</p> <p>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), enable channel 1 test mode (GMOption_Channel1TestMode) and toggle pixel mapping (GMOption_EnablePixelMap).</p> <p><b>Note:</b> GMOption_EnablePixelMap only has an effect if the pixel mapping mode is set to PixelMappingMode_GMBased (see SetPixelMappingMode())</p> <p>GMOptions can be masked together like below:</p> <p>SetGMOptions(GMOption_DisablePhotonCollect   GMOption_DebugMode);</p>

No.	41
Function name	bool GetGMOptions(GMOption* options)
Additional notes	<p>Returns GM options for active GM.</p> <p>The function populates a GMOption variable passed to the function via a pointer.</p>

No.	42
Function name	bool SetDelayTime(int delayTimeInNanoSeconds)
Additional notes	<p>Defines delay time for ASIC readout.</p> <p>delayTimeInNanoSeconds argument must be between 20 and 5100 and must be divisible by 20.</p>

No.	43
Function name	bool GetDelayTime(int* delayTimeInNanoSeconds)
Additional notes	<p>The function returns the delay time for the active GM.</p> <p>The function populates an int passed to the function via a pointer.</p>

No.	44
Function name	bool SetTimestampResolution(int resolutionInMicroSeconds)
Additional notes	<p>Defines timestamp clock.</p> <p>resolutionInMicroSeconds argument must be between 100 and 25500 and must be divisible by 100.</p>

No.	45
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.

No.	46
Function name	bool SetGMADC1Channel(int channel)
Additional notes	Defines ADC1 input.  channel argument must be 1 or 2.

No.	47
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.

No.	48
Function name	bool SetGMADC2Channel(int channel)
Additional notes	Defines ADC2 input.  channel must be 1 or 2.

No.	49
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.

No.	50
Function name	bool ReadGMADC1(int* DAC, int* volts)
Additional notes	Returns ADC1 data.  The function populates two ints passed to the function via pointers.

No.	51
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

No.	52
Function name	bool SetGMReadoutOptions(GMReadoutOption options)
Additional notes	<p>This function allows the user to specify additional data to be included in a packet during photon collection.</p> <p>GMReadoutOption values are:</p> <p>GMReadoutOpt_None          GMReadoutOpt_NegativeEnergy (NEGATIVE_EN)          GMReadoutOpt_TimeDetect (TD_EN)</p> <p>GMReadoutOptions can be masked together like below:</p> <pre>SetGMReadoutOptions (   GMReadoutOpt_NegativeEnergy   GMReadoutOpt_TimeDetect );</pre>

  

No.	53
Function name	bool GetGMReadoutOptions(GMReadoutOption* options)
Additional notes	<p>Returns GM readout options for active GM.</p> <p>The function populates a GMReadoutOption variable passed to the function via a pointer.</p>

  

No.	54
Function name	bool SetGMReadoutMode(GMReadoutMode mode)
Additional notes	<p>Sets ASIC readout mode.</p> <p>Valid GMReadoutMode values are:</p> <p>GMReadout_ReadAll          GMReadout_SparsifiedMode</p>

  

No.	55
Function name	bool GetGMReadoutMode(GMReadoutMode* mode)
Additional notes	<p>Returns readout mode for active GM.</p> <p>The function populates a GMReadoutMode variable passed to the function via a pointer.</p>



No.	56
Function name	bool SetGMCathodeMode(GMCathodeMode mode)
Additional notes	<p>Defines cathode mode. Depending on the selected mode, the GM FPGA drives a different amount of CK clocks for the ASIC.</p> <p>Valid GMCathodeMode values are:</p> <p>GMCathMode_Unipolar GMCathMode_MultiThreshold GMCathMode_Bipolar</p>

No.	57
Function name	bool GetGMCathodeMode(GMCathodeMode* mode)
Additional notes	<p>Returns the cathode mode for the active GM.</p> <p>The function populates a GMCathodeMode variable passed to the function via a pointer.</p>

## Gamma Module Pulser Setup Functions

No.	58
Function name	bool SetGMPulserFrequency(GMPulserFrequency freq)
Additional notes	<p>This function sets the ASIC signals timing which trigger the pulser inside the ASIC.</p> <p>Valid GMPulserFrequency values are:</p> <p>GMPulserFreq_100Hz          GMPulserFreq_1kHz          GMPulserFreq_10kHz          GMPulserFreq_100kHz</p>

No.	59
Function name	bool GetGMPulserFrequency(GMPulserFrequency* freq)
Additional notes	<p>This function returns the pulser frequency for the active GM.</p> <p>The function populates a GMPulserFrequency variable passed to the function via a pointer.</p>

No.	60
Function name	bool SetGMPulserOptions(GMPulserOption options)
Additional notes	<p>This function allows the user to set the test pulse generating signals for the ASIC anodes and/or cathodes.</p> <p>GMPulserOption values are:</p> <p>GMPulserOpt_None          GMPulserOpt_Anode (A_PLSR_EN)          GMPulserOpt_Cathode (C_PLSR_EN)</p> <p>GMPulserOptions can be masked together like below:</p> <p>SetGMPulserOptions(GMPulserOpt_Anode   GMPulserOpt_Cathode);</p>

No.	61
Function name	bool GetGMPulserOptions(GMPulserOption* a_options)
Additional notes	<p>Returns the pulser options for the active GM.</p> <p>The function populates a GMPulserOption variable passed to the function via a pointer.</p>

No.	62
Function name	bool SetNoOfPulses(int noOfPulses)
Additional notes	<p>Defines number of pulses generated when starting a photon collection.</p> <p>noOfPulses argument must be between 0 and 15. 0 = continual pulsing.</p>

No.	63
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

No.	64
Function name	bool SetASICGlobalOptions(ASICGlobalOptions options)
Additional notes	<p>ASICGlobalOptions values are:</p> <p>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)</p> <p>ASICGlobalOptions can be masked together like below:</p> <p>SetASICGlobalOptions(ASICGlobal_SingleEventMode   ASICGlobal_Validation);</p>

No.	65
Function name	bool GetASICGlobalOptions(ASICGlobalOptions* options)
Additional notes	<p>Returns the ASIC Global options for the active GM.</p> <p>The function populates an ASICGlobalOptions variable passed to the function via a pointer.</p>

No.	66
Function name	bool SetTimingChannelUnipolarGain(TimingChannelUnipolarGain gain)
Additional notes	<p>Valid TimingChannelUnipolarGain values are:</p> <p>TimingChannelUnipolarGain_27mV            TimingChannelUnipolarGain_81mV</p>

No.	67
Function name	bool GetTimingChannelUnipolarGain(TimingChannelUnipolarGain* gain)
Additional notes	<p>Returns the timing channel unipolar gain for the active GM.</p> <p>The function populates a TimingChannelUnipolarGain variable passed to the function via a pointer.</p>

No.	68
Function name	bool SetASICReadoutMode(GMASICReadoutMode mode)
Additional notes	<p>The data acquisition system:</p> <ul style="list-style-type: none"> <li>(i) Uses a fast clock to identify channels with events above threshold.</li> <li>(ii) Readout above threshold channels and their selected neighbours.</li> <li>(iii) Initiates a third sparse read cycle to extract data for negative peaks.</li> </ul> <p>Valid GMASICReadoutMode values are:</p> <p>GMASICReadout_NormalSparsified GMASICReadout_EnhancedSparsified</p>

No.	69
Function name	bool GetASICReadoutMode(GMASICReadoutMode* mode)
Additional notes	<p>Returns the ASIC readout mode for the active GM.</p> <p>The function populates a GMASICReadoutMode variable passed to the function via a pointer.</p>

No.	70
Function name	bool SetChannelGain(ChannelGain gain)
Additional notes	<p>Valid ChannelGain values are:</p> <p>ChannelGain_20mV ChannelGain_60mV</p>

No.	71
Function name	bool GetChannelGain(ChannelGain* gain)
Additional notes	<p>Returns the channel gain for the active GM.</p> <p>The function populates a ChannelGain variable passed to the function via a pointer.</p>

No.	72
Function name	bool SetCathodeTestModeInput(TestModeInput input)
Additional notes	<p>Cathode Channel Input for Test Mode.</p> <p>Valid TestModeInput values are:</p> <p>TestModeInput_Step TestModeInput_Ramp</p>

No.	73
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.

No.	74
Function name	bool SetInternalLeakageCurrentGenerator(InternalLeakageCurrentGenerator currGen)
Additional notes	Valid InternalLeakageCurrentGenerator values are:  InternalLeakageCurrentGenerator_60pA InternalLeakageCurrentGenerator_0A

No.	75
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.

No.	76
Function name	bool SetAnodeTestPulseEdge(TestPulseEdge edge)
Additional notes	Valid TestPulseEdge values are:  TestPulseEdge_InjectNegCharge TestPulseEdge_InjectPosAndNegCharge

No.	77
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.

No.	78
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

No.	79
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.

No.	80
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

No.	81
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.

No.	82
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

No.	83
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.

No.	84
Function name	bool SetCathodeTimingChannelsShaperPeakingTime(TimingChannelsShaperPeakingTime peakingTime)
Additional notes	Valid TimingChannelsShaperPeakingTime values are:  TimingChannelShaperPeakingTime_100nS TimingChannelShaperPeakingTime_200nS TimingChannelShaperPeakingTime_400nS TimingChannelShaperPeakingTime_800nS

No.	85
Function name	bool GetCathodeTimingChannelsShaperPeakingTime(TimingChannelsShaperPeakingTime* peakingTime)
Additional notes	The function populates a TimingChannelsShaperPeakingTime variable passed to the function via a pointer.

No.	86
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.

No.	87
Function name	bool GetCathodeTimingChannelsSecondaryMultiThresholdsDisplacementStep(int* displacementStep)
Additional notes	The function populates an int passed to the function via a pointer.

No.	88
Function name	bool SetMultipleFiringSuppressTime(MultipleFiringSuppressionTime suppressTime)
Additional notes	Valid MultipleFiringSuppressionTime values are:  MultipleFiringSuppressionTime_62_5nS MultipleFiringSuppressionTime_125nS MultipleFiringSuppressionTime_250nS MultipleFiringSuppressionTime_600nS

No.	89
Function name	bool GetMultipleFiringSuppressTime(MultipleFiringSuppressionTime* suppressTime)
Additional notes	The function populates a MultipleFiringSuppressionTime variable passed to the function via a pointer.



No.	90
Function name	bool SetTestPulseStep(ChannelType type, int testPulseStep)
Additional notes	<p>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.</p> <p>testPulseStep must be between 0 and 1023, each step represents ~1.85mV, test pulse runs from 0 to 1.85V.</p>

No.	91
Function name	bool GetTestPulseStep(ChannelType type, int* testPulseStep)
Additional notes	<p>ChannelType argument must be one of ChannelType_Anode or ChannelType_Cathode.</p> <p>The function populates an int passed to the function via a pointer.</p>

No.	92
Function name	bool SetChannelThresholdStep(ChannelThresholdType type, int thresholdStep)
Additional notes	<p>Valid ChannelThresholdType values are:</p> <p>ChannelThresholdType_CathodeTimingPrimaryMultiThresholdBiPolar (PD)            ChannelThresholdType_CathodeTimingUnipolar (PC)            ChannelThresholdType_CathodeEnergy (PB)            ChannelThresholdType_AnodeNegativeEnergy (PAn)            ChannelThresholdType_AnodePositiveEnergy (PA)</p> <p>thresholdStep must be between 0 and 1023, each step represents ~1.85mV, threshold runs from 0 to 1.85V.</p>

No.	93
Function name	bool GetChannelThresholdStep(ChannelThresholdType type, int* thresholdStep)
Additional notes	<p>Valid ChannelThresholdType values are:</p> <p>ChannelThresholdType_CathodeTimingPrimaryMultiThresholdBiPolar (PD)            ChannelThresholdType_CathodeTimingUnipolar (PC)            ChannelThresholdType_CathodeEnergy (PB)            ChannelThresholdType_AnodeNegativeEnergy (PAn)            ChannelThresholdType_AnodePositiveEnergy (PA)</p> <p>The function populates an int passed to the function via a pointer.</p>

No.	94
Function name	bool SetCathodeTestClockType(CathodeTestClockType clockType)
Additional notes	<p>Valid CathodeTestClockType values are:</p> <p>CathodeTestClockType_CopyAnodeTestClock            CathodeTestClockType_ArrivesOnSDI_NSDI</p>

No.	95
Function name	bool GetCathodeTestClockType(CathodeTestClockType* clockType)
Additional notes	The function populates a CathodeTestClockType variable passed to the function via a pointer.

No.	96
Function name	bool SetTimingChannelBiPolarGain(TimingChannelBiPolarGain gain)
Additional notes	<p>Used in conjunction with Timing Channel unipolar gain for additional bipolar gain.</p> <p>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.</p> <p>Valid TimingChannelBiPolarGain values are:</p> <p>TimingChannelBiPolarGain_21mV            TimingChannelBiPolarGain_55mV            TimingChannelBiPolarGain_62mV            TimingChannelBiPolarGain_164mV</p>

No.	97
Function name	bool GetTimingChannelBiPolarGain(TimingChannelBiPolarGain* gain)
Additional notes	The function populates a TimingChannelBiPolarGain variable passed to the function via a pointer.

No.	98
Function name	bool SetCathodeChannelInternalLeakageCurrentGenerator(int channelNumber , CathodeChannel::InternalLeakageCurrentGenerator currGen)
Additional notes	<p>channelNumber must be 1 or 2.</p> <p>Valid CathodeChannel:: InternalLeakageCurrentGenerator values are:</p> <p>CathodeChannel:: InternalLeakageCurrentGenerator_350pA            CathodeChannel:: InternalLeakageCurrentGenerator_2nA</p> <p>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.</p>

No.	99
Function name	bool GetCathodeChannelInternalLeakageCurrentGenerator(int channelNumber , CathodeChannel::InternalLeakageCurrentGenerator* currGen)
Additional notes	<p>channelNumber must be 1 or 2.</p> <p>The function populates a CathodeChannel::InternalLeakageCurrentGenerator variable passed to the function via a pointer.</p>

No.	100
Function name	bool SetAnalogOutputToMonitor(AnalogOutput output)
Additional notes	<p>Valid AnalogOutput values are:</p> <p>AnalogOutput_NoFunction AnalogOutput_Baseline AnalogOutput_Temperature</p> <p>Other AnalogOutput elements exist however these are used internally when setting analog outputs that require more information, see functions following GetAnalogOutputToMonitor() below.</p> <p>You can only monitor one output at a time so this function overwrites any value set via SetAnodeChannelToMonitor(), SetCathodeEnergyTimingToMonitor() or SetDACToMonitor().</p>

No.	101
Function name	bool GetAnalogOutputMonitored(AnalogOutput* output)
Additional notes	The function populates an AnalogOutput variable passed to the function via a pointer.

No.	102
Function name	bool SetAnodeChannelToMonitor(int channel)
Additional notes	<p>channel argument must be between 1 and 128.</p> <p>You can only monitor one output at a time so this function overwrites any value set via SetAnalogOutputToMonitor(), SetCathodeEnergyTimingToMonitor() or SetDACToMonitor().</p>

No.	103
Function name	bool GetAnodeChannelMonitored(int* channel)
Additional notes	<p>Returns the anode channel monitored if SetAnodeChannelToMonitor() has been called.</p> <p>The function populates an int passed to the function via a pointer.</p>

No.	104
Function name	bool SetCathodeEnergyTimingToMonitor(CathodeEnergyTiming energyTiming)
Additional notes	<p>Valid CathodeEnergyTiming values are:</p> <p>CathodeEnergyTiming_Channel1Energy CathodeEnergyTiming_Channel1Timing CathodeEnergyTiming_Channel2Energy CathodeEnergyTiming_Channel2Timing</p> <p>You can only monitor one output at a time so this function overwrites any value set via SetAnalogOutputToMonitor(), SetAnodeChannelToMonitor() or SetDACToMonitor().</p>

No.	105
Function name	bool GetCathodeEnergyTimingMonitored(CathodeEnergyTiming* energyTiming)
Additional notes	<p>Returns the cathode energy/timing monitored if SetCathodeEnergyTimingToMonitor() has been called.</p> <p>The function populates a CathodeEnergyTiming variable passed to the function via a pointer.</p>

No.	106
Function name	bool SetDACToMonitor(DACS dac)
Additional notes	<p>This is the function dedicated to setting the DAC analog output.</p> <p>Valid DACS values are:</p> <p>DACS_AnodeEnergyThreshold  DACS_AnodeEnergyTransient  DACS_CathodeEnergyThreshold  DACS_CathodeTimingUnipolarThreshold  DACS_CathodeTimingFirstMultiThreshold  DACS_CathodeTimingSecondMultiThreshold  DACS_CathodeTimingThirdMultiThreshold  DACS_AnodeTestSignal  DACS_CathodeTestSignal</p> <p>You can only monitor one output at a time so this function overwrites any value set via SetAnalogOutputToMonitor(), SetAnodeChannelToMonitor() or SetCathodeEnergyTimingToMonitor().</p>

No.	107
Function name	bool GetDACMonitored(DACS* dac)
Additional notes	<p>Returns the DAC monitored if SetDACToMonitor() has been called.</p> <p>The function populates a DACS variable passed to the function via a pointer.</p>

## 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.

No.	108
Function name	void SetChannelUpdateType(ChannelUpdateType channelUpdateType)
Additional notes	Valid ChannelUpdateType values are:  ChannelUpdateType_SingleChannel ChannelUpdateType_Broadcast

No.	109
Function name	void SetActiveChannelType(ChannelType channelType)
Additional notes	Valid ChannelType values are:  ChannelType_Anode ChannelType_Cathode

No.	110
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 ChannelUpdateType_Broadcast.

### Channel functions

No.	111
Function name	bool MaskChannel(bool mask)
Additional notes	Channel Mask (shaper shut-down)

No.	112
Function name	bool IsChannelMasked(ChannelType type, int channelNumber, bool* mask)
Additional notes	<p>type must be one of ChannelType_Anode or ChannelType_Cathode.</p> <p>If type is ChannelType_Anode, channelNumber must be between 1 and 128, if ChannelType_Cathode, channelNumber must be 1 or 2.</p> <p>The function populates a bool passed to the function via a pointer.</p>

No.	113
Function name	bool EnableChannelTestCapacitor(bool enable)
Additional notes	Test Capacitor 200fF.

No.	114
Function name	bool IsChannelTestCapacitorEnabled(ChannelType type, int channelNumber , bool* enable)
Additional notes	<p>type must be one of ChannelType_Anode or ChannelType_Cathode.</p> <p>If type is ChannelType_Anode, channelNumber must be between 1 and 128, if ChannelType_Cathode, channelNumber must be 1 or 2.</p> <p>The function populates a bool passed to the function via a pointer.</p>

No.	115
Function name	bool MonitorAnodeSignal(Signal signal)
Additional notes	<p>Valid Signal values are:</p> <p>Signal_Positive</p> <p>Signal_Negative</p>

No.	116
Function name	bool GetAnodeSignalMonitored(int channelNumber, Signal* signal)
Additional notes	<p>channelNumber must be between 1 and 128.</p> <p>The function populates a Signal variable passed to the function via a pointer.</p>

No.	117
Function name	bool SetCathodeChannelShapedTimingSignal (CathodeChannel::ShapedTimingSignal signal)
Additional notes	<p>Valid CathodeChannel::ShapedTimingSignal values are:</p> <p>CathodeChannel::ShapedTimingSignal_Unipolar CathodeChannel::ShapedTimingSignal_Bipolar</p> <p>Providing that a cathode channel is being monitored (SetCathodeEnergyTimingToMonitor ()), this function controls what shaped timing signal is available on the monitor.</p>

No.	118
Function name	bool GetCathodeChannelShapedTimingSignal (int channelNumber, CathodeChannel::ShapedTimingSignal* signal)
Additional notes	<p>channelNumber must be 1 or 2.</p> <p>The function populates a CathodeChannel::ShapedTimingSignal variable passed to the function via a pointer.</p>

No.	119
Function name	bool SetCathodeChannelTimingMode(CathodeChannel::TimingMode mode)
Additional notes	<p>Valid CathodeChannel::TimingMode values are:</p> <p>CathodeChannel::TimingMode_Unipolar CathodeChannel::TimingMode_MultiThreshold_Unipolar CathodeChannel::TimingMode_Bipolar_Unipolar</p>

No.	120
Function name	bool GetCathodeChannelTimingMode(int channelNumber, CathodeChannel::TimingMode* mode)
Additional notes	<p>channelNumber must be 1 or 2.</p> <p>The function populates a CathodeChannel::TimingMode variable passed to the function via a pointer.</p>

No.	121
Function name	bool SetCathodeTimingChannelTrimStep (CathodeTimingChannelType channelType, int trimStep)
Additional notes	<p>Valid CathodeTimingChannelType values are:</p> <p>CathodeTimingChannelType_FirstMultiThresholdBiPolar (DAM) CathodeTimingChannelType_SecondMultiThreshold (DAMb) CathodeTimingChannelType_ThirdMultiThreshold (DAMc) CathodeTimingChannelType_Unipolar (DAU)</p> <p>trimStep must be between 0 and 15, each step represents 3.5mV and trim runs from -52.5 to 0.</p>

No.	122
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.

No.	123
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.

No.	124
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.

No.	125
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.

No.	126
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

No.	127
Function name	int GetNoOfCollectedPackets()
Additional notes	Returns how many packets have been collected.

No.	128
Function name	bool GetPacketData(int packetNo, PacketData data, int* returnedData)
Additional notes	<p>PacketData argument can be one of:</p> <p>PacketData_GMNo            PacketData_Timestamp            PacketData_PhotonCount            PacketData_ActualPhotonCount</p> <p>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.</p> <p>The function populates an int passed to the function via a pointer.</p>

No.	129
Function name	GetPhotonData(int packetNo, int photonNo, PhotonData data, int* returnedData)
Additional notes	<p>PhotonData can be one of:</p> <p>PhotonData_Coordinate            PhotonData_Energy            PhotonData_EnergyPosEvent            PhotonData_TimeDetect            PhotonData_TimeDetectPosEvent</p> <p>The function populates an int passed to the function via a pointer.</p>

## Additional functions

No.	130
Function name	std::string GetVersion()
Additional notes	This function returns the version of the API.

No.	131
Function name	bool RegWrite(int regNum, unsigned char MSB, unsigned char LSB)
Additional notes	<p>This function is to be used to write directly to system control registers.</p> <p>See registers 14-16 in Appendix A1 for register layout.</p>

No.	132
Function name	bool RegRead(int regNum, unsigned char* MSB, unsigned char* LSB)
Additional notes	<p>This function is to be used to read directly from system control registers.</p> <p>See registers 14-16 in Appendix A1 for register layout.</p> <p>The function populates two unsigned chars passed to the function via pointers.</p>

No.	133
Function name	bool GMRegWrite(int GMRegNum, unsigned char data)
Additional notes	<p>This function is to be used to write directly to GM registers.</p> <p>See registers 18-24 in Appendix A1 for register layout.</p>

No.	134
Function name	bool GMRegRead(int GMRegNum, unsigned char* data)
Additional notes	<p>This function is to be used to read directly from GM registers.</p> <p>See registers 18-24 in Appendix A1 for register layout.</p> <p>The function populates an unsigned char passed to the function via a pointer.</p>

No.	135
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.

No.	136
Function name	void SetConnectStatusFunction(callback_function func)
Additional notes	<p>typedef void(*callback_function)(const char*);</p> <p>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.</p> <p>The function doesn't have to be called but could be useful for diagnosing connection problems.</p> <p>e.g.</p> <pre>static void StatusCallbackFunc(const char* a_Data);  SpectDMDII::SetConnectStatusFunction(StatusCallbackFunc);</pre>

No.	137
Function name	void SetSysStatusFunction(callback_function sysStatusFunc)
Additional notes	<p>Argument type is the same as that used in SetConnectStatusFunction() above.</p> <p>This function allows the user to set a callback function which will then be called with system statuses.</p> <p>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().</p> <p>e.g.</p> <pre>static void StatusCallbackFunc(const char* a_Data);  SpectDMDII::SetSysStatusFunction(StatusCallbackFunc);</pre>

No.	138
Function name	void SetToolbarStatusFunction(callback_function toolbarStatusFunc)
Additional notes	<p>Argument type is the same as that used in SetConnectStatusFunction().</p> <p>This function allows the user to set a callback function which will then be called with a status intended for toolbars.</p> <p>The function doesn't have to be called but it does provide feedback on events such as configuration loading and packet collection.</p> <p>e.g.</p> <pre>static void ToolbarStatusCallbackFunc(const char* a_Data);  SpectDMDII::SetToolbarStatusFunction(ToolbarStatusCallbackFunc);</pre>

No.	139
Function name	void SetOperationErrorFunction(callback_function opErrorFunction)
Additional notes	<p>Argument type is the same as that used in SetConnectStatusFunction().</p> <p>This function allows the user to set a callback function which will then be called when a long operation has triggered an error.</p> <p>The function should be called to ensure that the application integrating with the API can react to errors during long operations.</p> <p>Functions making use of this in the API state this in their <i>Additional notes</i>.</p> <p>e.g.</p> <pre>static void OperationErrorCallbackFunc(const char* a_Data );</pre> <pre>SpectDMDII::SetOperationErrorFunction(OperationErrorCallbackFunc);</pre>

No.	140
Function name	void SetOperationProgressFunction(callback_function_int opProgressFunction)
Additional notes	<pre>typedef void(*callback_function_int)(int);</pre> <p>This function allows the user to set a callback function which will be called as a long operation progresses. The function reports the percentage complete of an operation.</p> <p>The function doesn't necessarily have to be called but it does allow the user to display an accurate indication of the progress of an operation in their application.</p> <p>Functions making use of this in the API state this in their <i>Additional notes</i>.</p> <p>e.g.</p> <pre>static void OperationProgressCallbackFunc(int a_Progress);</pre> <pre>SpectDMDII::SetOperationProgressFunction(OperationProgressCallbackFunc);</pre>

No.	141
Function name	void SetOperationCompleteFunction(void_callback_function opCompleteFunction)
Additional notes	<p>typedef void(*void_callback_function)();</p> <p>This function allows the user to set a callback function which will be called when a long operation finishes.</p> <p>The function should be called to ensure that the application integrating with the API knows when a long operation finishes so the application can take the necessary action e.g. close a progress dialog.</p> <p>Functions making use of this in the API state this in their <i>Additional notes</i>.</p> <p>e.g.</p> <pre>static void OperationCompleteCallbackFunc(); SpectDMDII::SetOperationCompleteFunction(OperationCompleteCallbackFunc);</pre>

No.	142
Function name	bool SetPacketBufferSize(int bufferSize)
Additional notes	<p>The packet buffer is a buffer within the API that is filled with packets when the user issues StartPhotonCollection().</p> <p>By default it is set to 1MB but can be set as high as 1024MB.</p> <p>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.</p> <p>bufferSize is in MB.</p>

No.	143
Function name	int GetPacketBufferSize()
Additional notes	Returns packet buffer size in MB.

No.	144
Function name	bool SetCameraUpdateMode(CameraUpdateMode cameraUpdateMode)
Additional notes	<p>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.</p> <p>If manual mode is set then the user needs to call the following functions:</p> <p>SendGBEControlData() SendGMData() SendASICGlobalData() SendASICChannelData()</p> <p>Valid values for cameraUpdateMode are:</p> <p>CameraUpdateMode_Manual CameraUpdateMode_Automatic</p> <p>By default, the camera update mode is CameraUpdateMode_Manual.</p>

No.	145
Function name	bool SendGBEControlData()
Additional notes	<p>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 “<i>System Functions</i>” section of this document.</p> <p>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.</p>

No.	146
Function name	bool SendGMData()
Additional notes	<p>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 <i>Gamma Module Functions</i> section of this document:</p> <p><i>Gamma Module Setup Functions</i> <i>Gamma Module ASIC Setup Functions</i> <i>Gamma Module Pulser Setup Functions</i></p> <p>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.</p>

No.	147
Function name	bool SendASICGlobalData()
Additional notes	<p>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 <i>Gamma Module ASIC Global Functions</i> section in the <i>Gamma Module Functions</i> section of this document.</p> <p>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.</p>

No.	148
Function name	bool SendASICChannelData()
Additional notes	<p>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 <i>Gamma Module ASIC Channel Functions</i> section in the <i>Gamma Module Functions</i> section of this document.</p> <p>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.</p>

## Appendices

### A1. Register layout

The following tables have the following format:

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



<b>RG16</b> <b>GBE CONTROL</b>  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 AXP63; 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

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

<b>RG18</b> <b>GM Control</b> Read and write. Resides in GM FPGA	7		
	6		
	5	ADC2_CHNL	defines ADC2 input
	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.

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

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

<b>RG21</b> <b>ASIC MODE</b>  Read and write.  Resides in GM FPGA	7	CATH_MODE1	
	6	CATH_MODE0	00- Unipolar Mode, 01-Multi threshold, 10-Bi-polar.
	5	ASIC Global Reset	H-to toggle asic global reset mode
	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.
	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	

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

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

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

RG 25-reserved			
----------------	--	--	--

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

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