GTDIO™

GT-DIO Product Family

GTDIO Software

Programmer's Reference Guide

Last Updated September 29, 2015

MARVIN TEST SOLUTIONS

Safety and Handling

Each product shipped by Marvin Test Solutions is carefully inspected and tested prior to shipping. The shipping box provides protection during shipment, and can be used for storage of both the hardware and the software when they are not in use.

The circuit boards are extremely delicate and require care in handling and installation. Do not remove the boards from their protective plastic coverings or from the shipping box until you are ready to install the boards into your computer.

If a board is removed from the computer for any reason, be sure to store it in its original shipping box. Do not store boards on top of workbenches or other areas where they might be susceptible to damage or exposure to strong electromagnetic or electrostatic fields. Store circuit boards in protective anti-electrostatic wrapping and away from electromagnetic fields.

Be sure to make a single copy of the software CD for installation. Store the original CD in a safe place away from electromagnetic or electrostatic fields. Return compact disks (CD) to their protective case or sleeve and store in the original shipping box or other suitable location.

Warranty

Marvin Test Solutions products are warranted against defects in materials and workmanship for a period of 12 months. Marvin Test Solutions shall repair or replace (at its discretion) any defective product during the stated warranty period. The software warranty includes any revisions or new versions released during the warranty period. Revisions and new versions may be covered by a software support agreement. If you need to return a board, please contact Marvin Test Solutions Customer Technical Services Department via http://www.marvintest.com/magic/ - the Marvin Test Solutions on-line support system.

If You Need Help

Visit our web site at http://www.marvintest.com more information about Marvin Test Solutions products, services and support options. Our web site contains sections describing support options and application notes, as well as a download area for downloading patches, example, patches and new or revised instrument drivers. To submit a support issue including suggestion, bug report or questions please use the following link: http://www.marvintest.com/magic/

You can also use Marvin Test Solutions technical support phone line (949) 263-2222. This service is available between 8:30 AM and 5:30 PM Pacific Standard Time.

Disclaimer

In no event shall Marvin Test Solutions or any of its representatives be liable for any consequential damages whatsoever (including unlimited damages for loss of business profits, business interruption, loss of business information, or any other losses) arising out of the use of or inability to use this product, even if Marvin Test Solutions has been advised of the possibility for such damages.

Copyright

Trademarks

ATEasy®, CalEasy, DIOEasy®, DtifEasy, WaveEasy	Marvin Test Solutions, Inc., Geotest – Marvin Test Systems, Inc (prior company name)
C++ Builder, Delphi	Embarcadero Technologies Inc.
LabView, LabWindowstm/CVI	National Instruments
Microsoft Developer Studio, Microsoft Visual C++, Microsoft Visual Basic, .NET, Windows 95, 98, NT, ME, 2000, XP, VISTA, Windows 7 and 8	Microsoft Corporation

Table of Contents

Chapter 1 - Driver Function Reference

This chapter presents information on the GT-DIO driver functions for the GT-DIO system. The functions are organized in alphabetical order.

These general notes apply to the GT-DIO driver functions:

- All pointers are far pointers.
- Strings are ASCIIZ (null terminated).
- Returned value (*pnStatus*) is < 0 for failure and 0 for success unless an exception was specified in the function documentation. Use the DioGetErrorString function to retrieve the error string if *pnStatus* is < 0.
- All functions are declared void _far Pascal xxx(...).

Variable Naming Conventions

Parameter name prefixes and data types are defined as follows:

Type	Prefix	Type Description	Example
ARRAY	*a*	Prefix this before the simple type.	*anArray*
BOOL	*b*	Signed 8-bit integer as Boolean. TRUE if 0, FALSE otherwise.	*bSelected*
BYTE	*uc*	Unsigned 8-bit integer (unsigned CHAR type).	*ucSection*
CHAR	*c*	Signed 8-bit integer.	*cKeyPress*
DOUBLE	*d*	64-bit floating point.	*dReading*
DWORD	*dw*	Unsigned 32-bit integer (non-handle).	*dwTimeout*
DWORD	*hwnd*	Window handle - unsigned 32-bit integer used as handle. Only the lower 16-bits are used under Windows 3.1.	*hwndPanel*
LONG	*l*	Signed 32-bit integer.	*lBits*
LP*type*	*p*	32-bit pointer. Far pointer for Windows 3.1. Usually returns a value. Prefix this before the simple type.	*pnStatus*
LPSTR	*sz*	Zero-terminated ASCIIZ text string.	*szMsg*
SHORT	*n*	Signed 16-bit integer.	*nMode*
WORD	*w*	Unsigned 16-bit integer.	*wCount*

Table 1-1: Marvin Test Solutions's Variable Naming Conventions

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e and GX5295 Functions List

The table below is an alphabetical listing, by function that identifies functions with applicable boards. Pages that follow contain detailed descriptions of these functions in the order listed below.

Function	Applies To
DioArm	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioCompareData	General information or utility function
DioCompareDataWithMaskArray	General information or utility function
DioCompareFiles	File
DioConfigureInterfaceStandardLevel	GX5280, GX5290, GX5290e, GX5295, File
DioDataPack	General information or utility function
DioDataUnpack	General information or utility function
DioFileClose	File
DioFileConvert	General utility function
DioFileDeleteBoard	File
DioFileFindLabel	File
DioFileGetBoardHandle	File
DioFileGetChannelName	File
DioFileGetHandle	File
DioFileGetLabel	File
DioFileGetNumberOfBoards	File
DioFileGetNumberOfSteps	File
DioFileImport	File
DioFileInsertBoard	File
DioFileOpen	File
DioFileReadIgnoreData	File
DioFileSetChannelName	File
DioFileSetLabel	File
DioFileSetNumberOfSteps	File
DioFileWriteIgnoreData	File
DioFreqDoublerGetClkSource	GX5150
DioFreqDoublerSetupClkSource	GX5150
DioGetBClkFrequency	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Function	Applies To
DioGetBoardSummary	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioGetBoardType	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioGetCalibrationInfo	GX5055, GX5295
DioGetChannelsOutputStates	GX5280, GX5290, GX5290e, GX5295, File
DioGetChannelsVoltageLevel	GX5280, GX5290, GX5290e, File
DioGetClkStrobeDelay	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioGetClkStrobeExternalGateMode	GX5280, GX5290, GX5290e, GX5295, File
DioGetClkStrobeSource	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioGetControlToPxiTriggerBusLineMode	GX5290e, File
DioGetCounterOverflowMode	GX5150, GX5280, GX5290, GX5290e, GX5295
DioGetDriverSummary	General information function.
DioGetErrorString	General information function.
DioGetExternalInputsStates	GX5150, File
DioGetExternalJumpATriggerMode	GX5150, File
DioGetExternalRefClkFrequency	GX5280, GX5290, GX5290e, GX5295, File
DioGetFrequency	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioGetInputLoadCurrent	GX5055, GX5295, File
DioGetInputLoadCommutatingVoltage	GX5055, GX5295, File
DioGetInputLoadResistance	GX5055, File
DioGetInputLoadState	GX5055, GX5295, File
DioGetInputThresholdVoltages	GX5055, GX5295, File
DioGetInputInterface	GX5280, GX5290, GX5290e, File
DioGetInterfaceStandardLevel	GX5280, GX5290, GX5290e, File
DioGetIOConfiguration	GX5150, GX5280, File
DioGetIOModuleType	GC5050, GX5050, GX5150
DioGetIOPinsStaticDirection	GX5055, GX5295
DioGetJumpAddress	GX5150, File
DioGetMemoryBankSize	GC5050, GX5050, GX5055, GX5150
DioGetNextCtrlCommandStep	GX5280, GX5290, GX5290e, GX5295, File

Function	Applies To
DioGetNumberOfSteps	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioGetOperationMode	GC5050, GX5050
DioGetOutputClocksState	GX5280, GX5290, GX5290e, GX5295
DioGetOutputDataFormat	GX5055, File
DioGetOutputOverCurrentEnable	GX5055
DioGetOutputOverCurrentStates	GX5055
DioGetOutputSlewRate	GX5055, File
DioGetSkewDelay	GX5055, GX5295, File
DioGetOutputState	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295
DioGetOutputVoltages	GX5055, GX5295, File
DioGetPauseCount	GX5150
DioGetPxiStarTriggerMode	GX5055, GX5280, GX5290, GX5290e, GX5295, File
DioGetPxiTriggerBusLineMode	GX5280, GX5290, GX5290e, GX5295, File
DioGetPauseCounterMode	GX5150, File
DioGetSequencerMode	GX5150
DioGetSignalEdgeOrLevelMode	GX5280, GX5290, GX5290e, GX5295
DioGetSlaveHandle	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioGetStepCounter	GX5150
DioGetTermination	GX5150, GX5280, GX5290, GX5290e
DioGetTriggerDEvent	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioGetTriggerMode	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioGetTriggerPEvent	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioGetTriggerTEvent	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioGetTriggerXEventSource	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioGetTriStateTerminationMode	GX5295, File
DioGetTriStateTerminationVoltage	GX5295, File
DioHalt	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295

Function	Applies To
DioInitialize	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295
DioLevelShifterGetLoadResistance	GC5050, GX5050, GX5150
DioLevelShifterGetOutputMode	GC5050, GX5050, GX5150
DioLevelShifterGetSummary	GC5050, GX5050, GX5150
DioLevelShifterGetVoltage	GC5050, GX5050, GX5150
DioLevelShifterSetByLogicFamily	GC5050, GX5050, GX5150
DioLevelShifterSetLoadResistance	GC5050, GX5050, GX5150
DioLevelShifterSetOutputMode	GC5050, GX5050, GX5150
DioLevelShifterSetVoltage	GC5050, GX5050, GX5150
DioLoadFile	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295
DioMemoryFill	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioMeasure	GX5055, GX5295, File
DioPanel	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioPanelEx	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioPause	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioReadCtrlCommand	GX5280, GX5290, GX5290e, GX5295, File
DioReadCtrlMemory	GC5050, GX5050, GX5055, GX5150, File
DioReadDirectionMemory	GX5055, GX5290, GX5290e, GX5295, File
DioReadInMemory	GC5050, GX5050, GX5055, GX5290, GX5290e, GX5295, File
DioReadIOMemory	GX5150, GX5280, File
DioReadIOPinsValue	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioReadOutMemory	GC5050, GX5050, GX5055, GX5290, GX5290e, GX5295, File
DioReadProgramCounter	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioReadStatusRegister	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioReadValidDataMemory	GX5055, File
DioReadXRegister	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Function	Applies To
DioRealTimeCompareClearResultMemory	GX5290, GX5295, File
DioRealTimeCompareGetConditionValue	GX5290, GX5295, File
DioRealTimeCompareGetDataDelay	GX5290, GX5295, File
DioRealTimeCompareGetFailureCount	GX5290, GX5295, File
DioRealTimeCompareGetInputDataClockEdge	GX5290, GX5295, File
DioRealTimeCompareGetResultsDataType	GX5290, GX5295, File
DioRealTimeCompareGetStopCondition	GX5290, GX5295, File
DioRealTimeCompareReadExpectedMemory	GX5290, GX5295, File
DioRealTimeCompareReadMaskMemory	GX5290, GX5295, File
DioRealTimeCompareReadResults	GX5290, GX5295, File
DioRealTimeCompareSetupConditionValue	GX5290, GX5295, File
DioRealTimeCompareSetupDataDelay	GX5290, GX5295, File
DioRealTimeCompareSetupInputDataClockEdge	GX5290, GX5295, File
DioRealTimeCompareSetupResultsDataType	GX5290, GX5295, File
DioRealTimeCompareSetupStopCondition	GX5290, GX5295, File
DioRealTimeCompareWriteExpectedMemory	GX5290, GX5295, File
DioRealTimeCompareWriteMaskMemory	GX5290, GX5295, File
DioReset	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioResetChannels	GX5295
DioResetOutputOverCurrentStates	GX5055
DioRemapChannel	GX5280, GX5290, GX5295, File
DioSaveFile	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioSaveDomainConfiguration	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioSelfTest	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioSetJumpAddress	GX5150, File
DioSetPauseCount	GX5150, File
DioSetPauseCounterMode	GX5150, File
DioSetupBClkFrequency	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioSetupChannelsOutputStates	GX5280, GX5290, GX5290e, GX5295, File
DioSetupChannelsVoltageLevel	GX5280, GX5290, GX5290e, File

Function	Applies To
DioSetupClkStrobeDelay	GX5150, GC5050, GX5050, GX5055, GX5280, GX5290, GX5290e, GX5295, File
DioSetupClkStrobeExternalGateMode	GX5280, GX5290, GX5290e, GX5295, File
DioSetupClkStrobeSource	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioGetControlToPxiTriggerBusLineMode	GX5290e, File
DioSetupCounterOverflowMode	GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioSetupExternalJumpATriggerMode	GX5150, File
DioSetupExternalPauseAndTriggerMode	GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioSetupFrequency	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioSetupInitialization	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioSetupInitializationVisa	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioSetupInputInterface	GX5280, GX5290, GX5290e, File
DioSetupInputLoadCurrent	GX5055, GX5295, File
DioSetupInputLoadCommutatingVoltage	GX5055, GX5295, File
DioSetupInputLoadResistance	GX5055, File
DioSetupInputLoadState	GX5055, GX5295, File
DioSetupInputThresholdVoltages	GX5055, File
DioSetupIOConfiguration	GX5150, GX5280, GX5290, GX5290e, GX5295, File
DioSetupIOPinsStaticDirection	GX5055, GX5295
DioSetupOutputClocksState	GX5280, GX5290, GX5290e, GX5295
DioSetupOutputDataFormat	GX5055, File
DioSetupOutputOverCurrentEnable	GX5055
DioSetupOutputSlewRate	GX5055, File
DioSetupSkewDelay	GX5055, GX5295, File
DioSetupOutputState	GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e,
DioSetupOutputVoltages	GX5055, GX5295, File
DioSetupPxiStarTriggerMode	GX5055, GX5280, GX5290, GX5290e, GX5295, GX5295, File

Functions unique to the GX5280, GX5290, GX5290e and GX5295 boards

Function	Applies To
DioGetOutputClocksState	GX5280, GX5290, GX5290e, GX5295
DioGetSignalEdgeOrLevelMode	GX5280, GX5290, GX5290e, GX5295
DioSetupOutputClocksState	GX5280, GX5290, GX5290e, GX5295
DioSetupSignalEdgeOrLevelMode	GX5280, GX5290, GX5290e, GX5295
DioConfigureInterfaceStandardLevel	GX5280, GX5290, GX5290e, File
DioGetChannelsOutputStates	GX5280, GX5290, GX5290e, GX5295, File
DioGetChannelsVoltageLevel	GX5280, GX5290, GX5290e, File
DioGetClkStrobeExternalGateMode	GX5280, GX5290, GX5290e, GX5295, File
DioGetExternalRefClkFrequency	GX5280, GX5290, GX5290e, GX5295, File
DioGetInputInterface	GX5280, GX5290, GX5290e, File
DioGetInterfaceStandardLevel	GX5280, GX5290, GX5290e, File
DioGetNextCtrlCommandStep	GX5280, GX5290, GX5290e, GX5295, File
DioGetPxiTriggerBusLineMode	GX5280, GX5290, GX5290e, GX5295, File
DioGetTriStateTerminationMode	GX5295, File
DioGetTriStateTerminationVoltage	GX5295, File
DioReadCtrlCommand	GX5280, GX5290, GX5290e, GX5295, File
DioSetupChannelsOutputStates	GX5280, GX5290, GX5290e, GX5295, File
DioSetupChannelsVoltageLevel	GX5280, GX5290, GX5290e, File
DioSetupClkStrobeExternalGateMode	GX5280, GX5290, GX5290e, GX5295, File
DioSetupInputInterface	GX5280, GX5290, GX5290e, File
DioWriteCtrlCommand	GX5280, GX5290, GX5290e, GX5295, File

Functions unique to the GX5295 boards

Function	Applies To
DioGetAuxiliaryToTimingInput	GX5295, File
DioGetAuxiliaryToTimingOutput	GX5295, File
DioGetChannelMode	GX5295, File
DioGetInputDataSource	GX5295, File
DioGetInputLoadCurrent	GX5295, File
DioGetInputLoadCommutatingVoltage	GX5295, File
DioGetInputLoadState	GX5295, File
DioGetInputThresholdVoltages	GX5295, File
DioGetSkewDelay	GX5295, File
DioGetOutputVoltages	GX5295, File
DioGetOverTemperatureStatus	GX5295, File
DioGetTriStateTerminationMode	GX5295, File
DioGetTriStateTerminationVoltage	GX5295, File
DioMeasure	GX5295, File
DioPmuGetComparatorsSource	GX5295, File
DioPmuGetComparatorsValues	GX5295, File
DioPmuGetComparisonResult	GX5295, File
DioPmuGetForcedCurrent	GX5295, File
DioPmuGetForcedCurrentCommutatingVoltage	GX5295, File
DioPmuGetForcedVoltage	GX5295, File
DioPmuSetupComparatorsSource	GX5295, File
DioPmuSetupComparatorsValues	GX5295, File
DioPmuSetupForcedCurrent	GX5295, File
DioPmuSetupForcedCurrentCommutatingVoltage	GX5295, File
DioPmuSetupForcedVoltage	GX5295, File
DioRealTimeCompareClearChannelsStatus	GX5295
DioRealTimeCompareGetActiveChannels	GX5295, File
DioRealTimeCompareGetDomainFailStatus	GX5295
DioRealTimeCompareGetMode	GX5295, File
DioRealTimeCompareReadChannelsStatus	GX5295, File
DioRealTimeCompareSetupActiveChannels	GX5295, File
DioRealTimeCompareSetupMode	GX5295, File

Function	Applies To
DioSaveDomainConfiguration	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioSetupAuxiliaryToTimingInput	GX5295, File
DioSetupAuxiliaryToTimingOutput	GX5295, File
DioSetupChannelMode	GX5295, File
DioSetupInputDataSource	GX5295, File
DioSetupInputLoadCurrent	GX5295, File
DioSetupInputLoadCommutatingVoltage	GX5295, File
DioSetupInputLoadState	GX5295, File
DioSetupInputThresholdVoltages	GX5295, File
DioSetupSkewDelay	GX5295, File
DioSetupOutputVoltages	GX5295, File

Functions unique to the GX5055 boards

Function	Applies To
DioGetInputLoadCurrent	GX5055, File
DioGetInputLoadCommutatingVoltage	GX5055, File
DioGetInputLoadResistance	GX5055, File
DioGetInputLoadState	GX5055, File
DioGetInputThresholdVoltages	GX5055, File
DioGetIOPinsStaticDirection	GX5055
DioGetOutputDataFormat	GX5055, File
DioGetOutputOverCurrentEnable	GX5055
DioGetOutputOverCurrentStates	GX5055
DioGetOutputSlewRate	GX5055, File
DioGetOutputVoltages	GX5055, File
DioMeasure	GX5055, File
DioPowerSupplyGetRailsState	GX5055
DioPowerSupplyGetRailsVoltage	GX5055
DioPowerSupplyGetStatus	GX5055
DioPowerSupplyIsSupported	GX5055
DioPowerSupplyMeasure	GX5055
DioPowerSupplyResetFault	GX5055
DioPowerSupplySetRailsState	GX5055
DioPowerSupplySetRailsVoltage	GX5055
DioReadIoPinsValidity	GX5055, GX5295
DioReadValidDataMemory	GX5055, File
DioResetOutputOverCurrentStates	GX5055
DioSaveDomainConfiguration	GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295
DioSetupInputLoadCurrent	GX5055, File
DioSetupInputLoadCommutatingVoltage	GX5055, File
DioSetupInputLoadResistance	GX5055, File
DioSetupInputLoadState	GX5055, File
DioSetupInputThresholdVoltages	GX5055, File
DioSetupIOPinsStaticDirection	GX5055
DioSetupOutputDataFormat	GX5055, File

Function	Applies To
DioSetupOutputOverCurrentEnable	GX5055
DioSetupOutputSlewRate	GX5055, File
DioSetupOutputVoltages	GX5055, File
DioWriteValidDataMemory	GX5055, File

Functions unique to the GX5290 and GX5295 Real Time Compare only boards

Function	Applies To
DioRealTimeCompareGetConditionValue	GX5290, GX5295, File
DioRealTimeCompareGetDataDelay	GX5290, GX5295, File
DioRealTimeCompareGetFailureCount	GX5290, GX5295, File
DioRealTimeCompareGetInputDataClockEdge	GX5290, GX5295, File
DioRealTimeCompareGetResultsDataType	GX5290, GX5295, File
DioRealTimeCompareGetStopCondition	GX5290, GX5295, File
DioRealTimeCompareReadExpectedMemory	GX5290, GX5295, File
DioRealTimeCompareReadMaskMemory	GX5290, GX5295, File
DioRealTimeCompareReadResults	GX5290, GX5295, File
DioRealTimeCompareSetupConditionValue	GX5290, GX5295, File
DioRealTimeCompareSetupDataDelay	GX5290, GX5295, File
DioRealTimeCompareSetupInputDataClockEdge	GX5290, GX5295, File
DioRealTimeCompareSetupResultsDataType	GX5290, GX5295, File
DioRealTimeCompareSetupStopCondition	GX5290, GX5295, File
DioRealTimeCompareWriteExpectedMemory	GX5290, GX5295, File
DioRealTimeCompareWriteMaskMemory	GX5290, GX5295, File

DioArm

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Switches the state from HALT to PAUSE of the specified Master board and enables trigger events.

Syntax

DioArm (*nMasterHandle, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

DioArm must be called in order to switch state from HALT to PAUSE.

The board must be in the HALT mode prior calling this function; otherwise the function returns an error.

GC5050, GX5050, GX5050, GX5150, GX5280, GX5290, GX5290e and GX5295:

All can be armed and trigger from any step (boundaries of 32-bit), in order to do so the user just needs to set the program counter before issuing the ARM command. The default starting step is zero.

Example

In the following example, the board will switch state from HALT to PAUSE, and then RUN immediately when **DioArm** is called:

```
DioWriteXRegister (nMasterHandle, 0x000A, &nStatus);
DioSetupTriggerXEventSource (nMasterHandle, 1, &nStatus);
DioSetupTriggerMode (nMasterHandle, 1, &nStatus);
DioSetupTriggerDEvent (nMasterHandle, 0xFFFF, 0x000A, &nStatus);
DioArm (nMasterHandle, &nStatus);
```

See Also

DioTrig, **DioHalt, DioReadStatusRegister, DioSetupTriggerMode, DioGetErrorString**

DioCompareData

Applies To

General utility function, not board related.

Purpose

Compares two arrays with a mask value, returns the first value for both arrays that differ.

Syntax

DioCompareData (*pvBuffer1*, *pvBuffer2*, *pdwStep1*, *pdwStep2*, *pvData1*, *pvData2*, *dwSize*, *dwMask*, *nDataWidth*, *pnStatus*)

Parameters

Name	Type	Comments
pvBuffer1	PVOID	First buffer
pvBuffer2	PVOID	Second buffer
pdwStep1	PDWORD	Sets the starting step for first buffer. Returns the failed step number on comparison fail, or –1 on success.
pdwStep2	PDWORD	Starting step for second buffer. Returns the failed step number on comparison fail, or –1 on success.
pvData1	PVOID	Returns the failed step data from *pvBuffer1* on comparison fail, or –1 on success.
pvData2	PVOID	Returns the failed step data from *pvBuffer2* on comparison fail, or –1 on success.
dwSize	DWORD	Number of steps to compare.
dwMask	DWORD	Mask value when comparing values.
nDataWidth	SHORT	Array data type: 0 COMPARE_DATA_BYTE: data type BYTE. 1 COMPARE_DATA_WORD: data type WORD. 2 COMPARE_DATA_DWORD: data type DWORD.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function is for general use in order to compare two arrays using a mask value.

Example

The following example compares the first 1000 cells while using a mask:

```
DWORD dwStep1=0, dwStep2=0, dwData1, dwData2;
SHORT nStatus;
DioCompareData (pdwBuffer1, pdwBuffer2, &dwStep1, &dwStep2, &dwData1, &dwData2, 1000, 0xAA55AA55,
2, &nStatus)
```

See Also

DioCompareDataWithMaskArray, DioGetErrorString

DioCompareDataWithMaskArray

Applies To

General utility function, not board related.

Purpose

Compares two arrays with a mask array values, returns the first value for both arrays that differ.

Syntax

DioCompareDataWithMaskArray (*pvBuffer1*, *pvBuffer2*, *pdwStep1*, *pdwStep2*, *pvData1*, *pvData2*, *dwSize*, *pvMaskBuffer*, *pdwMaskStep*, *nDataWidth*, *pnStatus)*

Parameters

Name	Type	Comments
pvBuffer1	PVOID	First buffer
pvBuffer2	PVOID	Second buffer
pdwStep1	PDWORD	Starting step for first buffer. Returns the failed step number on comparison fail, or –1 on success.
pdwStep2	PDWORD	Starting step for second buffer. Returns the failed step number on comparison fail, or –1 on success.
pvData1	PVOID	Returns the failed step data from *pvBuffer1* on comparison fail, or –1 on success.
pvData2	PVOID	Returns the failed step data from *pvBuffer2* on comparison fail, or –1 on success.
dwSize	DWORD	Number of steps to compare.
pvMaskBuffer	PVOID	Mask buffer values when comparing values.
pdwMaskStep	PDWORD	Starting step for the mask buffer. Returns the failed step number on comparison fail, or –1 on success.
nDataWidth	SHORT	Array data type: 0 COMPARE_DATA_BYTE: data type BYTE. 1 COMPARE_DATA_WORD: data type WORD. 2 COMPARE_DATA_DWORD: data type DWORD.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function is for general use in order to compare two arrays using a mask value.

Example

The following example compares the first 1000 cells of two buffers while using a mask array:

```
DWORD dwStep1=0, dwStep2=0, dwMaskStep=0, dwData1, dwData2;
SHORT nStatus;
DioCompareData (pdwBuffer1, pdwBuffer2, &dwStep1, &dwStep2, &dwData1, &dwData2, 1000,
pvMaskBuffer, dwMaskStep, 2, &nStatus)
```

See Also

DioCompareData, DioGetErrorString

DioCompareFiles

Applies To

File.

Purpose

Compares two DIO files.

Syntax

DioCompareFiles (*szFileName1, szFileName2, pdwStep1, pdwStep2, pdwIn1, pdwIn2, nBoards, pnStatus)*

Parameters

Name	Type	Comments
szFileName1 *szFileName2*	STRING	DIO files to be compared. Both files must be in the **DIOEasy 2.0** format.
pdwStep1 *pdwStep2*	PDWORD	When calling this function, these parameters contain the (0 based) starting step number for comparing szFileName1 and szFileName2. The function returns step numbers (if any) where contents differ. Negative one (-1) is returned if the end of file is reached and no difference is encountered.
pdwIn1 *pdwIn2*	PDWORD	Returns array containing the first step number with difference for the Master and all Slaves. The function fills the array with the failed step from the two files to be compared. Each element contains the step-input value for each board. The size of the array is the number of configured Slaves plus 1 (for the Master).
nBoards	SHORT	The number of elements in either *pdwIn1* or *pdwIn2* arrays. This number represents the number of boards associated with these files (Master and Slaves).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the files do not contain the same number of steps, the function returns -1 in the last step of *pdwStepX* (where X represents the shorter file). Consecutive calls to the function returns the next step values for *pdwStepY* (where Y is the longer file). The function treats steps without corresponding match step as empty or null not zero.

The function compares only *.DIO files.

Example

The following example prints all differences between the two files. The first file is a ***DIOEasy2.0*** file and the second file is a file created by the **DioSaveFile** function.

```
DWORD dwStep1=0, dwStep2=0, dwIn1, dwIn2;
SHORT nStatus;
do{ DioCompareFiles ("a.dio", "b.di", &dwStep1, &dwStep2,    &dwIn1, &dwIn2, 1, &nStatus);
    if (dwStep1 == -1 && dwStep2 == -1) break;
    printf ("failed step=%1X, vector1=%1X, vector2=%1X",     dwStep1, dwIn1, dwIn2);
} while(1);
```

See Also

DioSaveFile, DioLoadFile, DioGetErrorString

DioConfigureInterfaceStandardLevel

Applies To

GX5280, GX5290, File

Purpose

Configure the board TTL interface standards to operate according to the specified level standard.

Syntax

DioConfigureInterfaceStandardLevel (*nHandle, nStandardLevel, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
nStandardLevel	SHORT	TTL interface standards: 0 DIO_INTERFACE_STANDARD_TWO_POINT_FIVE_VOLTS: Default interface level standard of 2.5V. 1 DIO_INTERFACE_STANDARD_THREE_POINT_THREE_VOLTS: Interface level standard of 3.3V. 2 DIO_INTERFACE_STANDARD_ONE_POINT_EIGHT_VOLTS: Interface level standard of 1.8V. 3 DIO_INTERFACE_STANDARD_ONE_POINT_FIVE_VOLTS: Interface level standard of 1.5V.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function configures the board to operate according to the specified level standard in order to interface with different TTL standards. All I/O data lines as well external input line such as external trigger, clock pause will operate according to the configured standards level.

NOTE: Power must be recycled in order for the changes to take effect.

Example

The following example configures the board interface TTL standards to the 3.3V:

```
DioConfigureInterfaceStandardLevel (nHandle, DIO_INTERFACE_STANDARD_THREE_POINT_THREE_VOLTS,
      &nStatus);
```

See Also

DioGetInterfaceStandardLevel, DioGerErrorString

DioDataPack

Applies To

GX5290, File

Purpose

Pack an array of data to a single DWord.

Syntax

DioDataPack (*nNumPins, nDataSize, dwOffset, nUnpackedDataType, pdwPackedData, pvUnpackedData, pnStatus)*

Parameters

Name	Type	Comments
nNumPins	SHORT	Number of pins: 1, 2, 4, 8, 16, 32
nDataSize	SHORT	Number of elements to pack from the unpacked data buffer.
dwOffset	SHORT	Offset to start packing the data from, e.g. if offset=1 and *nNumPins=4* then the packed data will start from bit 4.
nUnpackedDataType	SHORT	The array data type in bytes: 1. Byte 2. Word 4. DWord
pdwPackedData	PDWORD	Pointer to a DWORD returning the packed data as single DWord.
pvUnpackedData	PVOID	Pointer to the unpacked data buffer.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function helps packing data to 32-bit when the number of active channels is set programmatically to be less then 32-bits.

Note: By default the DIO needs data to be on 32-bits boundaries.

Example

The following example packs an array of data to a single DWord:

```
SHORT   nStatus;
BYTE    aucUnpackedData[8];
DWORD   dwPackedData
for (int i=0; i<8; i++)
    aucUnpackedData[i]=i;
DioDataPack (4, 8, 0, 1, &dwPackedData, ucUnpackedData, &nStatus);
```

See Also

DioDataUnpack

DioDataUnpack

Applies To

GX5290, File

Purpose

Unpack a single DWord to an array of data.

Syntax

DioDataUnpack (*nNumPins, nDataSize, dwOffset, nUnpackedDataType, dwPackedData, pvUnpackedData, pnStatus)*

Parameters

Name	Type	Comments
nNumPins	SHORT	Number of pins: 1, 2, 4, 8, 16, 32
nDataSize	SHORT	Number of elements to pack from the unpacked data buffer.
dwOffset	SHORT	Offset to start unpacking the data from, e.g. if offset=1 and *nNumPins=4* then the unpacked data will start from bit 4 at the packed data.
nUnpackedDataType	SHORT	The array data type in bytes: 1. Byte 2. Word 4. DWord
dwPackedData	DWORD	Packed data as single DWORD.
pvUnpackedData	PVOID	Pointer to the unpacked data buffer.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function helps unpacking data to form a single 32-bit to the specified unpacked size when the number of active channels is set programmatically to be less then 32-bits.

Note: By default the DIO needs data to be on 32-bits boundaries.

Example

The following example packs an array of data to a single DWord:

```
SHORT   nStatus;
BYTE    aucUnpackedData[8];
DWORD   dwPackedData
DioDataPack (4, 8, 0, 1, dwPackedData, ucUnpackedData, &nStatus);
```

See Also

DioDataPack

DioDomainGetOperatingMode

Applies To

GX5290, File

Purpose

Returns the DIO domain operating mode.

Syntax

DioDomainGetOperatingMode (*nMasterHandle, pnMode, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board or File board handle.
pnMode	PSHORT	DIO domain operating mode: 0 DIO_OPERATING_MODE_DEFAULT: Default operating mode. The DIO boards in the domain operate in their default mode. 1 DIO_OPERATING_MODE_REAL_TIME_COMPARE: DIO domain is running in real tome compare mode (GX5290 only).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The Gx5290 can change the way it is operating to either be a high speed digital I/O or to real tome compare mode. The settings will change all the Gx529x boards in the domain to operate in the same mode. The operating mode is saved to the INI file and is reapplied each time the DIO is initialized until it is changed.

In real time compare mode the output memory contains the expected vector values while the input memory contains the vector masked values. See the "Gx5290UG.pdf" for more details on the real tome compare mode operation.

Example

The following example returns the DIO domain operating mode:

```
SHORT   nStatus, nMode;
DioDomainGetOperatingMode (nMasterHandle, &nMode, &nStatus);
```

See Also

DioDomainSetupOperatingMode, DioGetErrorString

DioDomainSetupOperatingMode

Applies To

GX5290, File

Purpose

Sets the DIO domain operating mode.

Syntax

DioDomainSetupOperatingMode (*nMasterHandle, nMode, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board or File board handle.
nMode	SHORT	DIO domain operating mode: 0 DIO_OPERATING_MODE_DEFAULT: Default operating mode. The DIO boards in the domain operate in their default mode. 1 DIO_OPERATING_MODE_REAL_TIME_COMPARE: DIO domain is running in real tome compare mode (GX5290 only).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The Gx5290 can change the way it is operating to either be a high speed digital I/O or to real tome compare mode. The settings will change all the Gx529x boards in the domain to operate in the same mode. The operating mode is saved to the INI file and is reapplied each time the DIO is initialized until it is changed.

In real time compare mode the output memory contains the expected vector values while the input memory contains the vector masked values. See the “Gx5290UG.pdf” for more details on the real tome compare mode operation.

Example

The following example sets the DIO domain operating mode to real tome compare mode:

```
SHORT  nStatus;
DioDomainSetupOperatingMode (nMasterHandle, DIO_OPERATING_MODE_REAL_TIME_COMPARE, &nStatus);
```

See Also

DioDomainGetOperatingMode, DioGetErrorString

DioFileClose

Applies To

File.

Purpose

Closes a DIO file specified by the File board handle.

Syntax

DioFileClose *(hFile, pnStatus)*

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the **DioFileOpen** function.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Closes the file specified by the File board handle returned by the function **DioFileOpen**.

Example

The following example opens and closes a DIO file:

```
SHORT   nStatus, hFile;
CHAR    szFilename[]="newFile.dio";
DioFileOpen (szFilename, 1, &hFile, &nStatus);
DioFileClose (hFile, &nStatus)
```

See Also

DioFileOpen, DioFileDeleteBoard, DioGetErrorString

DioFileConvert

Applies To

General utility function.

Purpose

Converts a DIO file from one format to another.

Syntax

DioFileConvert *(szSourceFile, szDestFile, nDestFileType, pnStatus)*

Parameters

Name	Type	Comments
szSourceFile	LPSTR	The source file name to convert from can have any of the following formats: *.DIO: **DIOEasy** version 2.x or version 1.x. *.ASC: raw ASCII or ASCII with commands **DIOEasy** version 2.x or version 1.x.
szDestFile	LPSTR	The file name converted to. Can have any of the following formats: *.DIO: **DIOEasy** version 2.x. *.ASC: raw ASCII or ASCII with commands, **DIOEasy** version 2.x.
nDestFileType	SHORT	The converted file type can be as follow: 0 ***DIOEasy*** version 2.x 1 Raw ASCII 2 ***DIOEasy*** version 2.x ASCII with commands.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function identifies the source file type by its extension. The function will open ***DIOEasy*** 1.x files.

The function relies on the target file name having the appropriate extension according to the *nDestFileType*.

Example

The following example converts a ***DIOEasy*** 1.x file to ***DIOEasy*** 2.x format:

```
SHORT  nStatus;
DioFileConvert ("OldDio.dio", "NewDio.dio", 0, &nStatus);
```

See Also

DioFileOpen, DioFileDeleteBoard, DioGetErrorString

DioFileDeleteBoard

Applies To

File.

Purpose

Deletes existing board entry from a DIO file previously opened using **DioFileOpen** command.

Syntax

DioFileDeleteBoard *(hFile, nBoard, pnStatus)*

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen**.
nBoard	SHORT	The board number to delete. Board number can be 1 to 15, while 0 is for the Master. Only the last board in the file can be deleted at a time. The Master board cannot be deleted.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

As an example, if we have three boards, 0 is the Master, 1 and 2 are Slaves. If we delete board number 2, the file will have only two boards 0 (Master) and one Slave.

Example

The following example creates a DIO file, adds two boards to the file, sets the number of steps in the boards to 100,000 and then deletes board number two:

```
SHORT  nStatus, hFile, hFileBoard1, hFileBoard2;
DioFileOpen ("newfile.dio", 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
{   DioFileInsertBoard (hFile, 1, &hFileBoard1, &nStatus);
    DioFileInsertBoard (hFile, 2, &hFileBoard2, &nStatus);
    DioFileSetNumberOfSteps (hFile, 100000, &nStatus);
    DioFileDeleteBoard(hFile, 2, &nStatus);
}
```

See Also

DioFileOpen, DioFileSetNumberOfSteps, DioFileClose, DioFileGetHandle, DioGetErrorString

DioFileFindLabel

Applies To

File.

Purpose

Returns the specified label's step number following the specified starting step.

Syntax

DioFileFindLabel (*hFile, dwStartStep, pdwStepLabel, pszLabel, pnStatus*)

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen**.
dwStartStep	DWORD	Specified starting step to state looking for labels in the file.
pdwStepLabel	PDWORD	Returns the specified label's step number following the specified starting step, if label was not found *pdwStepLabel* returns -1.
pszLabel	PSTR	Pointer to the string holding the label's name.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns the specified label's step number following the specified starting step:

```
SHORT   nStatus, hFile;
DWORD   dwStepLabel;
DioFileOpen (szfilename, 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
    DioFileFindLabel (hFile, 0, &dwStepLabel, "Clock", &nStatus);
```

See Also

DioFileSetLabel, DioFileGetLabel, DioFileGetChannelName, DioGetErrorString

DioFileGetBoardHandle

Applies To

File.

Purpose

Returns the specified boards handle from a DIO file previously opened using **DioFileOpen** command.

Syntax

DioFileGetBoardHandle *(hFile, nBoard, phFileBoard, pnStatus)*

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen**.
nBoard	SHORT	The board numbers can be: 0: Master 1 to 15: Slaves.
phFileBoard	PSHORT	The specified boards handle.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Using the file board handle while calling different functions will store/retrieve that setting to/from the specific file board. Any function that the **Applies To** applies also to **File** can be accessed using the file handle.

Example

The following example creates a DIO file, adds two boards to the file, sets the number of steps in the boards to 100,000 and then gets both file boards handles. It is then sets trigger mode T and sets the D and P event values.

File number 1 will have trigger settings.

```
SHORT  nStatus, hFile, hFileBoard1, hFileBoard2;
DioFileOpen (szFilename, 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
{   DioFileInsertBoard (hFile, 1, &hFileBoard1, &nStatus);
    DioFileInsertBoard (hFile, 2, &hFileBoard2, &nStatus);
    DioFileSetNumberOfSteps (hFile, 100000, &nStatus);
    DioFileGetBoardHandle(hFile, 1, &hFileBoard1, &nStatus);
    DioFileGetBoardHandle(hFile, 2, &hFileBoard2, &nStatus);
    DioSetupTriggerMode(hFile, 2, &nStatus);
    DioSetupTriggerDEvent(hFile, 0x1234, 0xAA55, &nStatus);
    DioSetupTriggerPEvent(hFile, 0, 0xFFFF, &nStatus);
}
```

See Also

DioFileGetNumberOfSteps, DioFileOpen, DioFileClose, DioFileGetHandle, DioGetErrorString

DioFileGetChannelName

Applies To

File.

Purpose

Returns the specified file board channel's name.

Syntax

DioFileGetChannelName (*hFile, nChannel, pszChannelName, nMaxLength, pnStatus*)

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen**.
nChannel	SHORT	Board's channel number ranges from 0 to 31.
pszChannelName	PSTR	Pointer to the string holding the Board's channel name.
nMaxLength	SHORT	Maximum length of *pszChannelName*.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns channel 0 name:

```
SHORT   nStatus, hFile;
CHAR    szChName[256];
DioFileOpen (szfilename, 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
    DioFileGetChannelName (hFile, 0, szChName, 256,&nStatus);
```

See Also

DioFileSetChannelName, DioGetErrorString

DioFileGetHandle

Applies To

File.

Purpose

Returns the handle to the specified file name previously opened using **DioFileOpen** command.

Syntax

DioFileGetHandle *(sFullFilename, phFile, pnStatus)*

Parameters

Name	Type	Comments
sFullFilenam	LPCSTR	The file full name including the complete path.
phFile	PSHORT	Returns the specified File board handle.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function is useful whenever editing more than one file at a time. The return File board handle gives the user the ability to flip between files while editing.

Example

The following example creates two DIO files, "File1.dio" and "File2.dio", retrieves the File board handle for the first file, adds two boards to the file and closes the first file. It then retrieves the File board handle for the second file, adds two boards to the file and closes the second file.

```
SHORT nStatus, hFile, hFileBoard;
DioFileOpen ("File1.dio", 1, &hFile, &nStatus);
DioFileOpen ("File2.dio", 1, &hFile, &nStatus);
if (hFile1!=0 && nStatus==0)
{   /* retrieve the File board handle for the first file */
    DioFileGetHandle ("File1.dio", &hFile, &nStatus)
    DioFileInsertBoard (hFile, 1, &hFileBoard, pnStatus)
    DioFileInsertBoard (hFile, 2, &hFileBoard, pnStatus)
    DioFileClose (hFile, &nStatus);
    /* retrieve the File board handle for the second file */
    DioFileGetHandle ("File2.dio", &hFile, &nStatus)
    DioFileInsertBoard (hFile, 1, &hFileBoard, pnStatus)
    DioFileInsertBoard (hFile, 2, &hFileBoard, pnStatus)
    DioFileClose (hFile, &nStatus);
}
```

See Also

DioFileGetNumberOfSteps, DioFileOpen, DioFileClose, DioGetErrorString, DioFileGetBoardHandle

DioFileGetLabel

Applies To

File.

Purpose

Returns the specified step label.

Syntax

DioFileGetLabel (*hFile, dwStep, pszLabel, nMaxLength, pnStatus*)

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen**.
dwStep	DWORD	Label's step
pszLabel	PSTR	Pointer to the string holding the label.
nMaxLength	SHORT	Maximum length of *pszLabel*.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns the label at step 100:

```
SHORT   nStatus, hFile;
CHAR    szLabel[256];
DioFileOpen (szfilename, 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
    DioFileGetLabel (hFile, 100, szLabel, 256, &nStatus);
```

See Also

DioFileSetLabel, DioFileFindLabel, DioGetErrorString

DioFileGetLoadOptions

Applies To

File.

Purpose

Returns the specified file load options.

Syntax

DioFileGetLoadOptions (*hFile, pdwFileLoadOptions, pnStatus*)

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen**.
pdwFileLoadOptions	PDWORD	File load options: 0. DIO_FILE_LOAD_ALL_DOMAIN_RESET_ON_FILE_LOAD: default settings, in this mode all the data and settings will be loaded and applied to the hardware. As well as reset the domain. 1. DIO_FILE_NO_DOMAIN_RESET_ON_FILE_LOAD: setting this flag will not reset all the boards in the domain before loading the new settings from the DIO file (default is to always reset the domain). This may be used to prevent I/O levels to change when loading consecutive vector files. 2. DIO_FILE_LOAD_DATA_ONLY: settings this flag will only load the file data to the boards' in the domain. 4. DIO_FILE_LOAD_SETTINGS_ONLY: settings this flag will only load the file settings to the boards' in the domain.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns the load data only flag:

```
SHORT   nStatus, hFile;
DWORD   dwFileLoadOptions;
DioFileOpen (szfilename, 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
    DioFileGetLoadOptions (hFile, &dwFileLoadOptions, &nStatus);
```

See Also

DioFileSetLoadOptions, DioGetErrorString

DioFileGetNumberOfBoards

Applies To

File.

Purpose

Returns number of file boards in DIO file previously opened using **DioFileOpen** command.

Syntax

DioFileGetNumberOfBoards *(hFile, pnNumBoards, pnStatus)*

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen.**
pnNumBoards	PSHORT	Returns the number of file boards in the vector file.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Example

The following example creates a DIO file, adds a board and then returns the number of boards in the file:

```
SHORT  nStatus, hFile, hFileBoard, nNumBoards;
DioFileOpen (szfilename, 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
{ DioFileInsertBoard (hFile, 1, &hFileBoard, &nStatus);
  DioFileGetNumberOfBoards(hFile, &nNumBoards, &nStatus);
  if (nNumBoards!=2)
  prints("Error: mismatch number of boards in file");
  return;}
}
```

See Also

DioFileSetNumberOfSteps, DioFileOpen, DioFileClose, DioFileInsertBoard

DioFileGetNumberOfSteps

Applies To

File.

Purpose

Returns number of steps in DIO file previously opened using **DioFileOpen** command.

Syntax

DioFileGetNumberOfSteps *(hFile, pdwSteps, pnStatus)*

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen.**
pdwSteps	PDWORD	Step number can be between 1K to the maximum number of steps allowed for the specific board.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The number of steps is equal for all the file boards in the DIO file.

Note: The board with maximum number of steps limits the maximum numbers of steps that can be added to other boards. That is, if we have a Master with width set to 16 and a Slave with width set to 32, then the maximum steps can be 32M.

The file handle (*hFile*) that is associated with this file is return by **DioFielOpen** or by calling **DioFileGetHandle.**

Example

The following example creates a DIO file, adds a board and sets the number of steps in the board to 100K and then returns the number of steps in the file:

```
SHORT   nStatus, hFile, hFileBoard;
DWORD   dwSteps;
DioFileOpen (szfilename, 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
{ DioFileInsertBoard (hFile, 1, &hFileBoard, &nStatus);
 DioFileSetNumberOfSteps (hFile, 100000, &nStatus);
 DioFileGetNumberOfSteps(hFile, &dwSteps, &nStatus);
 if (dwSteps!=100000)
  prints("Error: mismatch number of steps in file");
  return;}
}
```

See Also

DioFileSetNumberOfSteps, DioFileOpen, DioFileClose, DioFileInsertBoard

DioFileImport

Applies To

General utility function.

Purpose

Convert WGL, STIL, VCD or eVCD file type to a DIO file.

Syntax

DioFileImport (*pszInputFile, pszOutputDioFile, nBoardType, bRealtimeCompare, pszImportOptions, pnImportOptionsMaxLen, bWait, phHandle, pszError, nErrMaxLen, pnStatus*)

Parameters

Name	Type	Comments
pszInputFile	PCSTR	Specified input WGL, STIL, VCD or eVCD file name.
pszOutputDioFile	PCSTR	Specified output DIO file name. Pass NULL or empty string to this parameter to request the default import options string as explained in the *pszImportOptions* parameter.
nBoardType	SHORT	Sets the target supported DIO board type: 0x55 DIO_BOARD_TYPE_GX5055 – Gx5055 DIO board type. 0x70 DIO_BOARD_TYPE_GX5290 - Gx5290 DIO board type. 0x75 DIO_BOARD_TYPE_GX5290E - Gx5290 Express DIO board type. 0x7A DIO_BOARD_TYPE_GX5295 - Gx5295 board type.
bRealtimeCompare	BOOL	If TRUE then the generated output DIO file will be Real Time compare type.
pszImportOptions	PSTR	Optional buffer to specify or receive import options string (null terminated). When *pszOutputDioFile* is not NULL, this parameter specifies the import options such as unused signals, signal direction and cycle-based timing (VCD/eVCD only), this parameter can be NULL. See the "GTDIO/DIOEasy Software User's Guide" chapter 2: import file for more details info and examples of import options string. When requesting default import options string and *pszImportOptions* is NULL or the buffer is not large enough to hold the return string then no data is copied and the required buffer size is stored in *pnImportOptionsMaxLen*. See the "GTDIO/DIOEasy Software User's Guide" chapter 2: import file section for more details.
pnImportOptionsMaxLen	PSHORT	The size of the import options buffer size. see the "GTDIO/DIOEasy Software User's Guide" chapter 2: import file.
bWait	BOOL	Determine if the function returns immediately or wait until the file import is complete: 0 The function returns immediately and the import file will be done at the background. In this case the user can check the import file progress using the **DioFileImportGetProgress** API. 1 The function returns only when file import in complete or in case of an error.

		NOTE: If *pszOutputDioFile* is set to NULL, i.e. the user wants to get back the *pszImportOptions* string , then bWait is ignored (internally set to TRUE).
phHandle	PHANDLE	If *bWait* was set to FALSE, the function returns a HANDLE to the file import object. The user can then use that handle to monitor the progress by polling the **DioFileImportGetProgress** API. The returned handle can be used with **WaitForSingleObject** Windows API to wait until import completed or timed out. If *bWait* was set to TRUE, this parameter will be ignored and can be passes as NULL.
pszError	PSTR	Buffer to contain the returned error or warning string (null terminated).
nErrMaxLen	SHORT	Size of the buffer to contain the error or warning string.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function converts the specified file to DIO file format.

Using the *pszImportOptions* string:

The *pszImportOptions* string specifed the diffent options for each signal. The options are direction, dpendency on other signals for direction and to be use or not in the convetered file.

In the following *pszImportOptions* example string, the MCLK signal is bidirectional and its direction is set to input whenever ACKHW signal is high. Also MRB is set to be unused in this

Signals]

MCLK : Bidir In When [ACKHW 1];

MRB : Unused Output;

ACKHW Input;

AUTOEN;

[End]

The user can get the *pszImportOptions* string form the import file API by calling **DioFileImport** with the *pszOutputDioFile* parameter set to NULL. In that case the function will not convert the input file but instead will fill the *pszImportOptions* string and *pnImportOptionsMaxLen* returns the size of the string.

If the user wants to know the eaxct size of the returned *pszImportOptions* string, and have buffer ready for it with the corrct size, then call **DioFileImport** with the *pszOutputDioFile* parameter and *pszImportOptions* both set to NULL.

Examples

Importing a file with bWait=TRUE and using default szImportOptions string:

The following example convert a STIL file to a Gx5295 DIO with Real time compare file

```
SHORT nStatus;
CHAR szError[512];
Convert the STIL file to a DIO file.
DioFileImport ("InputStilFile.stil", "OutputDioFile.dio", DIO_BOARD_TYPE_GX5295, TRUE, NULL,
                NULL, TRUE, NULL, szError, sizeof (szError), &nStatus);
```

Importing a file with bWait=TRUE and using szImportOptions string:

The following example convert a STIL file to a Gx5295 DIO with Real time compare file using the following steps:

1. Get the input STIL file *szImportOptions* string size.
2. Allocate a buffer with the correct *szImportOptions* string size.
3. Convert the STIL file to a DIO file.

```
SHORT nStatus;
SHORT nImportOptionsMaxLen;
CHAR  szError[512];
PCHAR pszImportOptions=NULL;


// request the buffer size to receive the default import options string  by passing NULL to
pszOutputDioFile and pszImportOptions
DioFileImport (“InputStilFile. stil”, NULL, DIO_BOARD_TYPE_GX5295, TRUE, NULL,
               &nImportOptionsMaxLen, TRUE, NULL, szError, sizeof (szError), &nStatus);
// Allocate a buffer with the correct szImportOptions string size
pszImportOptions = new char[nImportOptionsMaxLen];
// request the default import options string by passing NULL to pszOutputDioFile
DioFileImport (“InputStilFile. stil”, NULL, DIO_BOARD_TYPE_GX5295, TRUE, NULL,
               &nImportOptionsMaxLen, TRUE, NULL, szError, sizeof (szError), &nStatus);


// modify the default import options string, for example, set SignalA to unused …
[Signals]
    SignalA : unused Output;
[End]
// Convert the STIL file to a DIO file using modified import options.
DioFileImport (“InputStilFile.stil”, “OutputDioFile.dio”, DIO_BOARD_TYPE_GX5295, TRUE,
               pszImportOptions, &nImportOptionsMaxLen, TRUE, NULL, szError, sizeof (szError),
               &nStatus);
```

Importing a file using default szImportOptions string and monitoring the progress (bWait=FALSE):

The following example convert a STIL file to a Gx5295 DIO with Real time compare file

```
SHORT   nStatus;
CHAR    szStatus[512];
HANDLE  hHandle=NULL;

DioFileImport ("InputStilFile.stil", "OutputDioFile.dio", DIO_BOARD_TYPE_GX5295, TRUE, NULL, 0,
     FALSE, &hHandle, szStatus, sizeof (szStatus), &nStatus);


// Monitor the import file process
// Monitor the import file process
while (nStatus>=0)
{
    DioFileImportGetProgress(hHandle, szStatus, sizeof (szStatus), &nStatus);
    if (nStatus>0)
        printf("Progress = %i", (int) nStatus);   // continue
    else
       break;
     // wait for import to finish or time out every 250ms without consuming the CPU
     WaitForSingleObject(hHandle, 250);
}
printf(szStatus);   // print complete (nStatus=0), warning, errrors
```

See Also

DioFileImportPanel, DioFileImportGetProgress, DioGetErrorString

DioFileImportGetProgress

Applies To

General utility function.

Purpose

Returns the file import progress.

Syntax

DioFileImportGetProgress (*phHandle, pszStatus, nStatusMaxLen, pnStatus*)

Parameters

Name	Type	Comments
phHandle	PHANDLE	A HANDLE to the file import object that was returned by **DioFileImport** API, in order to monitor the progress of the import file process.
pszStatus	PSTR	Buffer to contain the returned status string (null terminated). This string will be empty while the file import is in progress (based on *pnStatus* parameter). When the file import is done, if the file import finish successfully this string will be empty or in case of a warning it will contain the warning message. In case of an error (*pnStatus* is negative), it will contain the error message.
nStatusMaxLen	SHORT	Size of the buffer to contain the status string.
pnStatus	PSHORT	When the file import is in progress *pnStatus* returns the percentage that was completed, from 1% to 100%. When file import is complete it returns 0 on success or <0 on failure.

Comments

The function can only be used after the user called **DioFileImport** and specified *bWait* =FALSE.

Example

The following example converts a STIL file to a Gx5295 DIO with Real time compare file and monitoring the progress (bWait=FALSE):

```
SHORT   nStatus;
CHAR    szStatus[512];
HANDLE  hHandle;

DioFileImport ("InputStilFile.stil", "OutputDioFile.dio", DIO_BOARD_TYPE_GX5295, TRUE, NULL, 0,
    FALSE, &hHandle, szStatus, sizeof (szStatus), &nStatus);


// Monitor the import file process
while (nStatus>=0)
{
    DioFileImportGetProgress(hHandle, szStatus, sizeof (szStatus), &nStatus);
    if (nStatus>0)
        printf("Progress = %i", (int) nStatus);   // continue
    else
       break;
     // wait for import to finish or time out every 250ms without consuming the CPU
     WaitForSingleObject(hHandle, 250);
}
printf(szStatus);   // print complete (nStatus=0), warning, errrors
```

See Also

DioFileImportPanel, DioFileImport, DioGetErrorString

DioFileImportPanel

Applies To

GX5055, GX5290, GX5290e, GX5295

Purpose

Opens the file import Panel window.

Syntax

DioFileImportPanel (*dwHwndParentl*)

Parameters

Name	Type	Comments
dwHwndParent	DWORD	Parent window handle. Used when creating the panel window.

Comments

The function is used to create the file import panel window. The file import panel window is opened as a modal, i.e. the panel will disable the parent window (*hwndParent*) and the function will return only after the window was closed by the user.

For more details on how to use the file import Panel dialog see the “GTDIO/DIOEasy Software User’s Guide” chapter 2: import file.

Example

The following example calls the file import Panel:

```
HWND    hwnd;
DioPanel (hwnd);
```

See Also

DioFileImport, **DioGetErrorString**

DioFileInsertBoard

Applies To

File.

Purpose

Inserts a File Board to a DIO file previously opened using **DioFileOpen** command.

Syntax

DioFileInsertBoard *(hFile, nBoard, phFileBoard, pnStatus)*

Parameters

Name	Type	Comments
hFile	SHORT	File handle associated with this File.
nBoard	SHORT	The board number to insert. Board number can be between 1 and 8. 0 is for the Master. The inserted board will become the last board.
phFileBoard	PSHORT	Returns a handle for later reference with this File Board (0 for error).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The file handle (*hFile*) that is associated with this file is return by **DioFielOpen** or by calling **DioFileGetHandle.**

The inserted board will have the same number of steps and channels as the Master. For example, if the Master was set to 16-bit wide and 32K steps, then the added Slave board will have the same parameters.

Example

The following example creates a DIO file, adds a board and sets the number of steps in the board to 100K:

```
SHORT  nStatus, hFile, hFileBoard;
DioFileOpen (szfilename, 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
{ DioFileInsertBoard (hFile, 1, &hFileBoard, &nStatus);
  DioFileSetNumberOfSteps (hFile, 100000, &nStatus);
}
```

See Also

DioFileSetNumberOfSteps, DioFileGetNumberOfSteps, DioFileOpen, DioFileClose, DioFileGetBoardHandle, DioGetErrorString

DioFileOpen

Applies To

File.

Purpose

Creates or opens a DIO file and returns a handle to access the file.

Syntax

DioFileOpen *(szFilename, nMode, pnBoardType, phFile, pnStatus)*

Parameters

Name	Type	Comments
szFilename	LPCSTR	Pointer to a null-terminated string that specifies the name of the DIO file to create or open.
nMode	SHORT	Specifies the type of access requested for the file, as follows: 0 Read only. If the file does not exist or cannot be found the function call fails. 1 Read and Write. The file must exist. 2 Create, opens an empty file for both reading and writing (see Comments).
pnBoardType	PSHORT	Returns the board type if nMode was set to 0 or 1. If nMode was set to 2, sets the new file to one of the following board types: GX5150 0x20 GC5050 0x30 GX5152 0x40 GX5050 0x50 GX5055 0x55 GX5280 0x60 GX5290 0x70 GX5290e 0x75 GX5295 0x7A
phFile	PSHORT	Returns a handle for later reference with this file (0 for error). The handle is the Master file board.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the type of access (*nMode)* is 2 and the given file exists, a "_" will be added in order not to overwrite it. For example, if the file name was "NewFile.dio" its name will be changed to "NewFile_.dio".

Creating a new file creates a DIO file using the ***DIOEasy2.0*** format, on success (returned status is 0) the file will have the default header information as follows:

- Board Type: e.g. GX5150
- Number of Steps: Number of steps. All files have the same number of steps. Default is 16384.
- Number of Channels: Total number of channels for all the boards. Default is 32.

The file will also have a subdirectory with one Master Board (board number 0) and all the default information that comes with it as follows:

Width:	32 channels	TriggerMask:	0xFFFF
Frequency:	5e6	DEvent:	0
B Clcck:	5e6	DMask:	0xFFFF
ExtFrequency:	2e6	PauseEvent:	0
Clock:	Internal	PauseMask:	0xFFFF
Strobe source:	Internal	StrobeTiming:	10 nSec
TriggerMode:	Disabled	XEvent:	0xFFFF
TriggerEvent:	0	XSource:	External

GX5150:

Direction:	Input
RegisterA:	0
RegisterB:	0

Example

The following example creates a DIO file, adds a board and sets the number of steps in the board to 100,000:

```
SHORT   nStatus, hFile, hFileBoard;
DioFileOpen (szfilename, 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
{ DioFileInsertBoard (hFile, 1, &hFileBoard, &nStatus);
  DioFileSetNumberOfSteps (hFile, 100000, &nStatus);
}
```

See Also

DioFileSetNumberOfSteps, DioFileGetNumberOfSteps, DioGetErrorString

DioFileReadIgnoreData

Applies To

File

Purpose

Reads Ignore data from a DIO file.

Syntax

DioFileReadIgnoreData *(hFile, pvIgnore, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen**.
pvIgnore	PVOID	Pointer to an array, array data type needs to comply with the file width settings and board type, see comments for details.
dwStart	DWORD	Starting step to read from.
dwSize	DWORD	Number of steps to read.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The ignore data sets the specified range of channels steps to either be ignored or not. A bit low means do not ignore, bit high means ignore. The ignore data has dual use:

- Using any Gx5290 family member the ignore data is being used specified channels steps to Hi-Z. When a channel step direction is set to output and ignore bit is set to high, upon loading the vector that channel step direction will be set as input.
- Comparing two DIO files channels steps with ignore bits set to high will not be compared.

Example

The following example reads 64 steps from the file:

```
DWORD adwIgnoreData [64];
DioFileReadIgnoreData (hFile, adwIgnoreData, 0, 64, &nStatus);
```

See Also

DioFileWriteIgnoreData, DioWriteOutMemory, DioWriteDirectionMemory, DioGetErrorString

DioFileSetChannelName

Applies To

File.

Purpose

Sets the specified file board channel's name.

Syntax

DioFileSetChannelName (*hFile, nChannel, pszChannelName, nMaxLength, pnStatus*)

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen**.
nChannel	SHORT	Board's channel number ranges from 0 to 31.
pszChannelName	PSTR	Pointer to the string holding the Board's channel name.
nMaxLength	SHORT	Maximum length of *pszChannelName*, the size is limited to maximum of 256 characters.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example sets channel 0 name to "Clock":

```
SHORT  nStatus, hFile;
DioFileOpen (szfilename, 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
    DioFileSetChannelName (hFile, 0, "Clock", sizeof "Clock", &nStatus);
```

See Also

DioFileGetChannelName, DioGetErrorString

DioFileSetLabel

Applies To

File.

Purpose

Sets the specified step label.

Syntax

DioFileSetLabel (*hFile, dwStep, pszLabel, pnStatus*)

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen**.
dwStep	DWORD	Sets label step.
pszLabel	PSTR	Pointer to the string holding the label's name, limited to 512 characters.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example sets a label at step 100:

```
SHORT  nStatus, hFile;
DioFileOpen (szfilename, 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
    DioFileSetLabel (hFile, 100, "Clock", &nStatus);
```

See Also

DioFileGetLabel, DioFileFindLabel, DioGetErrorString

DioFileSetLoadOptions

Applies To

File.

Purpose

Sets the specified file load options.

Syntax

DioFileSetLoadOptions (*hFile, dwFileLoadOptions, pnStatus*)

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen**.
dwFileLoadOptions	DWORD	File load options: 0. DIO_FILE_LOAD_ALL_DOMAIN_RESET_ON_FILE_LOAD: default settings, in this mode all the data and settings will be loaded and applied to the hardware. As well as reset the domain. 1. DIO_FILE_NO_DOMAIN_RESET_ON_FILE_LOAD: setting this flag will not reset all the boards in the domain before loading the new settings from the DIO file (default is to always reset the domain). This may be used to prevent I/O levels to change when loading consecutive vector files. 2. DIO_FILE_LOAD_DATA_ONLY: settings this flag will only load the file data to the boards' in the domain. 4. DIO_FILE_LOAD_SETTINGS_ONLY: settings this flag will only load the file settings to the boards' in the domain.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example sets the load data only flag:

```
SHORT  nStatus, hFile;
DioFileOpen (szfilename, 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
    DioFileSetLoadOptions (hFile, DIO_FILE_LOAD_DATA_ONLY, &nStatus);
```

See Also

DioFileGetLoadOptions, DioGetErrorString

DioFileSetNumberOfSteps

Applies To

File.

Purpose

Sets the number of steps for all boards in a file previously opened using DioFileOpen command.

Syntax

DioFileSetNumberOfSteps *(hFile, dwSteps, pnStatus)*

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen**.
dwSteps	DWORD	Number of steps. Can have the following values: Min. number: 1024 (all boards) Max. number GC5050/GX5050: 1M Max. number GX5055: 512K Max. number GX5150: 32M width set to 32, 64M width set to 16, 128M width set to 8 Max. number GX5280:32M (GX5281), 64M (GX5282), 128M (GX5283) Max. number GX5290: 64M (GX5292/GX5293)
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the new number of steps is less than the previous value, the extra steps will be truncated. If the new number of step is bigger than the previous value then the extra steps will be added at the end of the file, not changing the previous step's data.

Note: The board with maximum number of steps limits the maximum numbers of steps that can be added. That is, if we have a Master with width set to 16 and a Slave with width set to 32, then the maximum number of steps can be 32M.

Example

The following example creates a DIO file, adds two boards to the file, sets the number of steps in the boards to 100K and then gets both file boards handles:

```
SHORT  nStatus, hFile, hFileBoard1, hFileBoard2;
DioFileOpen (szfilename, 1, &hFile, &nStatus);
if (hFile!=0 && nStatus==0)
{   DioFileInsertBoard (hFile, 1, &hFileBoard1, &nStatus);
    DioFileInsertBoard (hFile, 2, &hFileBoard2, &nStatus);
    DioFileSetNumberOfSteps (hFile, 100000, &nStatus);
    DioFileGetBoardHandle(hFile, 1, &hFileBoard1, &nStatus);
    DioFileGetBoardHandle(hFile, 2, &hFileBoard2, &nStatus);
}
```

See Also

DioFileGetNumberOfSteps, DioGetErrorString

DioFileWriteIgnoreData

Applies To

File

Purpose

Writes Ignore data to a DIO file.

Syntax

DioFileWriteIgnoreData *(hFile, pvIgnore, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	Comments
hFile	SHORT	File board handle returned by the function **DioFileOpen**.
pvIgnore	PVOID	Pointer to an array, array data type needs to comply with the file width settings and board type, see comments for details.
dwStart	DWORD	Starting step to read from.
dwSize	DWORD	Number of steps to read.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The ignore data sets the specified range of channels steps to either be ignored or not. A bit low means do not ignore, bit high means ignore. The ignore data has dual use:

- Using any Gx5290 family member the ignore data is being used specified channels steps to Hi-Z. When a channel step direction is set to output and ignore bit is set to high, upon loading the vector that channel step direction will be set as input.
- Comparing two DIO files channels steps with ignore bits set to high will not be compared.

Example

The following example writes 64 steps to the file:

```
DWORD adwIgnoreData [64];
DioFileWriteIgnoreData (hFile, adwIgnoreData, 0, 64, &nStatus);
```

See Also

DioFileReadIgnoreData, DioWriteOutMemory, DioWriteDirectionMemory, DioGetErrorString

DioFreqDoublerGetClkSource

Applies To

GX5150

Purpose

Returns the clock source for the Frequency Doubler I/O module.

Syntax

DioFreqDoublerGetClkSource *(nHandle, pnSource, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
pnSource	PSHORT	Returned clock source values are: 0 Internal 1 External
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the Frequency Doubler I/O module is installed on GX5150 board, the module can be set by calling either **DioSetupClkStrobeSource** or **DioFreqDoublerSetupClkSource** functions.

Example

The following example sets the clock source to external and then verifies the settings:

```
SHORT  nStatus, nSource;
DioFreqDoublerSetupClkSource (nHandle, 1, &nStatus);
DioFreqDoublerGetClkSource (nHandle, &nSource, &nStatus);
```

See Also

DioFreqDoublerSetupClkSource, DioGetClkStrobeSource, DioSetupClkStrobeSource

DioFreqDoublerSetupClkSource

Applies To

GX5150

Purpose

Sets up the clock source for the Frequency Doubler I/O module.

Syntax

DioFreqDoublerSetupClkSource *(nHandle, nClkSource, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
nClkSource	SHORT	Clock source can have the following values: 0 Internal. 1 External.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the Frequency Doubler I/O module is installed on the GX5150 board, the module can be set by either calling **DioSetupClkStrobeSource** or **DioFreqDoublerSetupClkSource** functions.

Example

The following example sets the clock source to external and then verifies the settings:

```
SHORT  nStatus, nSource;
DioFreqDoublerSetupClkSource (nHandle, 1, &nStatus);
DioFreqDoublerGetClkSource (nHandle, &nSource, &nStatus);
```

See Also

DioFreqDoublerGetClkSource, DioGetClkStrobeSource, DioSetupClkStrobeSource

DioGetAuxiliaryToTimingInput

Applies To

GX5295, File.

Purpose

Returns the specified Auxiliary channel connections to the timing inputs signals.

Syntax

DioGetAuxiliaryToTimingInput (*nHandle, nAuxChannel, pdwInputComparator, pdwInputSignal, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nAuxChannel	SHORT	Specified Auxiliary Pin Electronics channel in the DIO board, channel can be as follows: 1000 DIO_AUX_CHANNEL_0: Auxiliary Pin Electronics channel 0. 1001 DIO_AUX_CHANNEL_1: Auxiliary Pin Electronics channel 1. 1002 DIO_AUX_CHANNEL_2: Auxiliary Pin Electronics channel 2. 1003 DIO_AUX_CHANNEL_3: Auxiliary Pin Electronics channel 3.
pdwInputComparator	PDWORD	The Auxiliary channel inputs signal can be programmatically connected to either the low voltage level comparator or the high voltage level comparator. Each bit represents one of the possible inputs as defined in *pdwInputSignal*. See Comments for details.
pdwInputSignal	PDWORD	The Auxiliary channel can be connected to any of the following signals (see comments for details): • DIO_AUX_TO_EXTERNAL_DISABLED = 0x0 • DIO_AUX_TO_EXTERNAL_CLOCK = 0x1 • DIO_AUX_TO_EXTERNAL_STROBE = 0x2 • DIO_AUX_TO_EXTERNAL_TRIGGER = 0x4 • DIO_AUX_TO_EXTERNAL_PAUSE = 0x8 • DIO_AUX_TO_EXTERNAL_CLOCK_ENABLE = 0x10. • DIO_AUX_TO_EXTERNAL_STROBE_ENABLE = 0x20 • DIO_AUX_TO_EXTERNAL_EVENT_LINE_0 = 0x100 • DIO_AUX_TO_EXTERNAL_EVENT_LINE_1 = 0x200 • DIO_AUX_TO_EXTERNAL_EVENT_LINE_2 = 0x400 • DIO_AUX_TO_EXTERNAL_EVENT_LINE_3 = 0x800
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The Auxiliary channel can be connected to configure to any of the following timing signals:

- DIO_AUX_TO_EXTERNAL_INPUT_DISABLED = 0x0: Disconnect all external input signals to the specified Auxiliary channel, default state after calling **DioReset** API.
- DIO_AUX_TO_EXTERNAL_CLOCK = 0x1: Auxiliary channel is connected to External Clock Input (reference to the internal clock PLL or sample clock source) pin number 31 on the Timing connector (J3). If *pdwInputComparator*=0x1 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_STROBE = 0x2: Auxiliary channel is connected to External Strobe Input pin number 32 on the Timing connector (J3). If *pdwInputComparator*=0x2 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_TRIGGER = 0x4: Auxiliary channel is connected to External Trigger Input (Overrides Run Command) pin number 26 on the Timing connector (J3). If *pdwInputComparator*=0x4 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_PAUSE = 0x8: Auxiliary channel is connected to External Pause Input (Overrides Pause Command) pin number 27 on the Timing connector (J3). If *pdwInputComparator*=0x8 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_CLOCK_ENABLE = 0x10: Auxiliary channel is connected to External Clock Enable Input (active low) pin number 28 on the Timing connector (J3). If *pdwInputComparator*=0x10 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_STROBE_ENABLE = 0x20: Auxiliary channel is connected to External Strobe. If *pdwInputComparator*=0x20 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_EVENT_LINE_0 = 0x100: Auxiliary channel is connected to Input External event 0 pin number 1 on the Timing connector (J3). If *pdwInputComparator*=0x100 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_EVENT_LINE_1 = 0x200: Auxiliary channel is connected to Input External event 1 pin number 2 on the Timing connector (J3) If *pdwInputComparator*=0x200 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_EVENT_LINE_2 = 0x400: Auxiliary channel is connected to Input External event 2 pin number 3 on the Timing connector (J3). If *pdwInputComparator*=0x400 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_EVENT_LINE_3 = 0x800: Auxiliary channel is connected to Input External event 3 pin number 4 on the Timing connector (J3) If *pdwInputComparator*=0x800 high voltage level comparator, otherwise uses the low voltage level comparator.

Example

The following example returns Auxiliary channel 0 connections to the timing inputs signals:

```
DWORD   dwInputSignal, dwInputComparator;
DioGetAuxiliaryToTimingInput (nHandle, DIO_AUX_CHANNEL_0, &dwInputComparator, &dwInputSignal,
                              &nStatus);
```

See Also

DioSetupAuxiliaryToTimingInput, DioSetupAuxiliaryToTimingOutput, DioGetErrorString

DioGetAuxiliaryToTimingOutput

Applies To

GX5295, File.

Purpose

Returns the specified Auxiliary channel connections to the timing output signals.

Syntax

DioGetAuxiliaryToTimingOutput (*nHandle, nAuxChannel, pdwOutputSignal, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nAuxChannel	SHORT	Specified Auxiliary Pin Electornics channel in the DIO board, channel can be as follows: 1000 DIO_AUX_CHANNEL_0: Auxiliary Pin Electronics channel 0. 1001 DIO_AUX_CHANNEL_1: Auxiliary Pin Electronics channel 1. 1002 DIO_AUX_CHANNEL_2: Auxiliary Pin Electronics channel 2. 1003 DIO_AUX_CHANNEL_3: Auxiliary Pin Electronics channel 3.
pdwOutputSignal	PDWORD	The Auxiliary channel can be connected to configured to any of the following timing signals: • DIO_AUX_TO_EXTERNAL_OUT_DISABLED = 0x0: Disconnect all specified Auxiliary channel to external output signals, default state after calling **DioReset** API. • DIO_AUX_TO_CLOCK_OUT = 0x1: Auxiliary channel is connected to User Output Clock (OClk) pin number 22 on the Timing connector (J3). • DIO_AUX_TO_STROBE_OUT = 0x2: Auxiliary channel is connected to Strobe Output (OStb) pin number 24 on the Timing connector (J3). • DIO_AUX_TO_RUN_OUT = 0x4: Auxiliary channel is connected to Output Run Indication (ORun) pin number 18 on the Timing connector (J3). • DIO_AUX_TO_ARM_OUT = 0x8: Auxiliary channel is connected to Output Arm Indication (OArm) pin number 17 on the Timing connector (J3).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Using the auxiliary channels capabilities of programmed output levels and skew delays the user can better conditioning those signals to meet his UUT requirements.

Example

The following example returns Auxiliary channel 0 connections to the timing output signal:

```
DWORD   dwOutputSignal;
DioGetAuxiliaryToTimingOutput (nHandle, DIO_AUX_CHANNEL_0, &dwOutputSignal, &nStatus);
```

See Also

DioSetupAuxiliaryToTimingInput, DioSetupAuxiliaryToTimingOutput, DioGetErrorString

DioGetBClkFrequency

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File.

Purpose

Returns the specified Master B clock frequency.

Syntax

DioGetBClkFrequency *(nMasterHandle, pdwFrequency, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
pdwFrequency	PDWORD	B Clock frequency value.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The B clock is a general-purpose, external, programmable clock.

GX5150: The B clock uses the internal reference clock.

Example

The following example sets the Master B clock frequency to 1MHz and returns its value:

```
SHORT  nStatus;
DWORD dwFrequency;
DioSetupBClkFrequency (nMasterHandle, 1e6, &nStatus);
DioGetBClkFrequency (nMasterHandle, &dwFrequency, &nStatus);
```

See Also

DioSetupBClkFrequency, DioSetupClkStrobeSource, DioGetErrorString

DioGetBoardSummary

Purpose

Returns the board information.

Syntax

DioGetBoardSummary(*nHandle*, *szSummary*, *nSumMaxLen*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Handle to a counter board.
szSummary	PSTR	Buffer to contain the returned board info (null terminated) string.
nSumMaxLen	SHORT	Size of the buffer to contain the error string.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The DIO board summary string provides information about the board. The following example is the returned board summery for a Gx5293 master board:

Gx5055:

"Gx5055 Master, FPGA-Version: 0xC028, I/O FPGA-Version: 0xFFFFC021, S/N: 50550009-AB-CD-19, Calibrated on: Fri Oct 21 16:08:48 2011"

Gx5290:

"Gx5293 Master, S/N 52930014, FPGA-Version: 0x7B15, I/O Data Interface Level: 2.5V, Installed Memory size: 256MB, Calibrated on: Wed Oct 26 09:03:36 2011"

Gx5295:

"Gx5295 Master, S/N 52950011-BC-DE-67, FPGA-Version: 0xB930, Installed Memory size: 256MB, Calibrated on: Wed Oct 26 09:03:36 2011"

Example

The following example returns the board summary:

```
SHORT nHandle, nStatus;
CHAR szSummary [256];

DioGetBoardSummary(nHandle, szSummary, sizeof(szSummary), &nStatus);
```

See Also

DioGetDriverSummary, **DioGetErrorString**

DioGetBoardType

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the board type code.

Syntax

DioGetBoardType *(nHandle, pnBoardType, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
pnBoardType	PSHORT	The board type values are: GX5150 (Master): 0x20 GX5151 (Slave): 0x21 GC5050 (Master): 0x30 GC5051 (Slave): 0x31 GX5152(Master): 0x40 GX5050: 0x50 GX5055: 0x55 GX5280: 0x60 GX5290: 0x70 GX5290e: 0x75 GX5295: 0x7A
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board type is read from the board associated with the handle.

Example

The following example returns the board type specified by the handle:

```
SHORT  nStatus, nBoardType;
DioGetBoardType (nHandle, &nBoardType, &nStatus);
switch(nBoardType)
{   case 0x20:
        printf ("Board type: GX5150");
        break;
    default:
        printf ("Board type: Invalid");
}
```

See Also

DioSetupInitialization, DioInitialize, DioGetErrorString

DioGetCalibrationInfo

Applies To

GX5055, GX5295

Purpose

Returns the calibration information.

Syntax

DioGetCalibrationInfo (*nHandle, pszCalibrationInfo, nInfoMaxLen, pnDaysUntilExpire, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
pszSummary	PSTR	Buffer to contain the returned board's calibration information (null terminated) string.
nSumMaxLen	SHORT	Size of the buffer to contain the error string.
pnDaysUntilExpire	PSHORT	Returns the number of days until or from expiration, if number is > 0 then calibration is current otherwise past due.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The returned board's calibration information has the following fields:

Model: model number, e.g. "Gx5295 Master"

Serial Number: serial number, e.g. 216

Control Number: Marvin Test Solutions control number, e.g. "*-CH-CB-0"

Production Calibration Date: Wed Oct 24 12:30:25 2010

Calibration Date: Wed Oct 24 12:31:58 2010

Recommended Interval: 1 year

Next Calibration Date: Fri Oct 24 12:31:58 2011

Status: calibration status can be either "Expired" followed by the number of days past expiration or "Current" followed by number of days until expire.

Example

The following example returns the board's calibration information string:

```
SHORT   nStatus;
char    szCalibrationInfo[1024];
BOOL    bExpired;
DioGetCalibrationInfo(nHandle, szCalibrationInfo, sizeof szCalibrationInfo, &bExpired,&nStatus);

szCalibrationInfo string printout:

Model: Gx5295 Master
Serial Number: 216
Control Number: *-CH-CB-0
Production Calibration Date: Wed Oct 24 12:30:25 2007
Calibration Date: Wed Oct 24 12:31:58 2007
Recommended Interval: 1 year
Next Calibration Date: Fri Oct 24 12:31:58 2008
Status: Current (350 days until expire)
```

See Also

DioInitialize, DioGetBoardSummary, **DioGetErrorString**

DioGetChannelMode

Applies To

GX5295

Purpose

Returns the specified channel operating mode.

Syntax

DioGetChannelMode (*nHandle, nChannel, pnMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
nChannel	SHORT	Specified channel in the DIO board to apply the settings: • I/O channels: 0-31. • Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
pnMode	PSHORT	Channel operating mode are 0. DIO_CHANNEL_MODE_DYNAMIC_IO: default operating mode, channel is in dynamic I/O mode and output is enabled. 1. DIO_CHANNEL_MODE_DISABLED: channel's output is disabled (Tri-State). 2. DIO_CHANNEL_MODE_OUTPUT_LOW: channel's output is set to low; the output low voltage corresponds to the output low voltage settings. 3. DIO_CHANNEL_MODE_OUTPUT_HIGH: channel's output is set to high; the output high voltage corresponds to the output high voltage settings. 4. DIO_CHANNEL_MODE_PMU_FORCE_CURRENT: channel's output is set to PMU (Parametric Measurement Unit) forced current mode. The channel's forced current is set to zero 5. DIO_CHANNEL_MODE_PMU_FORCE_VOLTAGE: channel's output is set to PMU (Parametric Measurement Unit) forced voltage mode. The channel's forced voltage is set to zero volts.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For detail regarding additional setting when set to PMU (Parametric Measurement Unit) forced current and forced voltage mode see **DioPmuSetupForcedCurrent** and **DioPmuSetupForcedVoltage** functions.

Example

The following example returns channel 0 operating mode:

```
SHORT   nMode, nStatus;
DioGetChannelMode (nHandle, 0, &nMode, &nStatus);
```

See Also

DioSetupChannelMode, DioPmuSetupForcedCurrent, DioPmuSetupForcedVoltage, DioGetErrorString

DioGetChannelsOutputStates

Applies To

GX5280, GX5290, GX5290e, GX5295, File.

Purpose

Returns the state of each output channel.

Syntax

DioGetChannelsOutputStates (*nHandle, pdwStates, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
pdwStates	PDWORD	Each of the 32-bit represents channel's outputs state. Bit low channel's outputs disabled, bit high channel's outputs enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function enabled or disabled each channel output. Disabled channels in output mode are in Tri-State. This is useful for connecting to a user bus

Example

The following example returns the state of each output channel:

```
SHORT   nStatus;
DioGetChannelsOutputStates (nMasterHandle, &dwStates, &nStatus);
```

See Also

DioSetupChannelsOutputStates, DioGetErrorString

DioGetChannelsVoltageLevel

Applies To

GX5280, GX5290, GX5290e, File.

Purpose

Returns channels voltage level.

Syntax

DioGetChannelsVoltageLevel (*nHandle, pdVoltage, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
pdVoltage	PDOUBLE	Channels voltage level can be in the range of 1.4V to 3.6V with 10mV resolution.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function program the voltage level, applied for all 32 channels, in the TTL Standard connector.

Example

The following example returns the channels voltage level:

```
SHORT   nStatus;
DioGetChannelsVoltageLevel (nHandle, &dVoltage, &nStatus);
```

See Also

DioSetupChannelsVoltageLevel, DioGetErrorString

DioGetClkStrobeDelay

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the specified clock delay value.

Syntax

DioGetClkStrobeDelay (*nHandle, nClock, pdDelay, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX5150, GC5050, GX5050, GX5055: Master or File board handle. GX5280, GX5290, GX5290e, GX5295: Master, Slave or File board handle.
nClock	SHORT	Clock/Strobe can be as follow: 0 CLK_DELAY: Main Clock. 1 USER_CLK_DELAY: Out Clock 2 STROBE_DELAY: Strobe 3 BOARD_OFFSET_DELAY (Gx528X /Gx529X only): Additional offset that advance or delay the specifed DIO board's Clock and Strobe. The offset values vacan be set between -3.0 nSec to +3.0 nSec with resolution of 0.25nSec. The main purpose of this delaty is to ease timing aligment between DIO boards in a domain.
pdDelay	LPDOUBLE	Delay values are in nSec GX5150, GC5050, GX5050 GX5055 Out Clock and Strobe: 0.0 to 64 nSec with increments of 0.25. GX5280/GX5290/GX5290e Main Clock: and Strobe, any of the following values ranges: 0-3 ns with increments of 0.25 ns. 4-7 ns with increments of 0.25 ns. 8-11 ns with increments of 0.25 ns. 12-15 ns with increments of 0.25 ns. 16-19 ns with increments of 0.25 ns. 20-23 ns with increments of 0.25 ns. 24-27 ns with increments of 0.25 ns. If *nClock* = BOARD_OFFSET_DELAY (Gx528X /Gx529X only): -3.0 to +3.0 ns with increments of 0.25 ns.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function adds a delay between the timing board clock source (i.e. programmable clock) and the following clocks:

NOTE: For the GX5280/GX5290/GX5290e family when running at frequencies above 100MHz only 0-3nS delay range is available (with resolution of 0.25nSec).

Example

The following example returns the main clock delay value:

```
SHORT   nStatus;
DOUBLE  dDelay;
DioGetClkStrobeDelay (nMasterHandle, 0, &dDelay, &nStatus);
```

See Also

DioSetupClkStrobeDelay, DioSetupClkStrobeSource, DioGetClkStrobeSource, DioGetErrorString

DioGetClkStrobeExternalGateMode

Applies To

GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the master DIO board Clock or Strobe external gate mode.

Syntax

DioGetClkStrobeExternalGateMode (*nHandle, nClkStrobe, pnMode, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	DIO board or File board handle.
nClkStrobe	SHORT	0. DIO_INTERNAL_CLOCK: Internal Clock. 1. DIO_INTERNAL_STROBE: Internal Strobe
pnMode	PSHORT	External gate mode can be: 0. Disabled 1. Enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function enables full control on the timing of when data is captured or clocked out for each DIO board that resides in the same domain using external input signal.

DIO_INTERNAL_CLOCK:

Enables or disables gating the internal Clock of the specified DIO board. For this mode to be active the direction of all the board's channels must be set to output. When the mode is enabled the External Clock Enable input (J3 pin 28) will gate the clock of the specified DIO board. When the External Clock Enable input is set to low the specified DIO board's internal Clock will be active and the DIO will run, when the External Clock Enable input will be high (default state since the input has IS pulled up to VCC). The DIO will be in Pause state.

DIO_INTERNAL_STROBE:

Enables or disables gating the internal Strobe of the specified DIO board. For this mode to be active the direction of all the board's channels must be set to input. When the mode is enabled the External Strobe Enable input (J3 pin 29) will gate the strobe of the specified DIO board. When the External Strobe Enable input is set to low the specified DIO board's internal Strobe will be active and the DIO will run, when the External Strobe Enable input will be high (default state since the input has IS pulled up to VCC). The DIO will be in Pause state

Example

The following example returns the internal clock external gate mode:

```
SHORT   nMode, nStatus;
DioGetClkStrobeExternalGateMode (nHandle, DIO_INTERNAL_CLOCK, &nMode, &nStatus);
```

See Also

DioSetupClkStrobeExternalGateMode, **DioSetupClkStrobeDelay, DioSetupClkStrobeSource, DioGetClkStrobeSource, DioGetErrorString**

DioGetClkStrobeSource

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the specified Master board's clock and strobe source.

Syntax

DioGetClkStrobeSource *(nMasterHandle, pnSource, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board or File board handle.
pnSource	PSHORT	Clock and strobe source values are as follow: 0. Internal Clock and Strobe (default). Clock is programmable using an internal reference. Strobe is internal in accordance with strobe timing (default). 1. External Clock, internal Strobe. External clock. Strobe is internal in accordance with strobe timing. This setting enables the External Clock input without the need to pull down the External Clock Enable line. 2. Clock programmed by external, internal Strobe. Clock is Programmable using external clock reference. Strobe is internal in accordance with strobe timing. **Note**: not supported by Gx5055. 3. External Clock and Strobe. Both clock and strobe are external. This setting enables the External Clock and External Strobe inputs without the need to pull down the External Clock Enable and External Strobe Enable lines. 0. Internal Clock, external Strobe. Clock is programmable using an internal reference Strobe is external, setting enables the External Strobe input without the need to pull down the External Strobe Enable line (GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e and GX5295).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The **DioSetupFrequency** function can be used to set the frequency when the selected clock source (nSource) is set to 0. **DioSetupClkStrobeDelay** may be called to set the time interval between the strobe and clock signals.

Example

The following example returns the clock and strobe source from a board specified by the board handle:

```
SHORT  nStatus, nMode;
DioGetClkStrobeSource (nMasterHandle, &nMode, &nStatus);
```

See Also

DioSetupClkStrobeSource, DioSetupClkStrobeExternalGateMode, DioGetErrorString

DioGetControlToPxiTriggerBusLineMode

Applies To

GX5290e, File

Purpose

Returns the specified control signal to PXI Trigger Line and its state.

Syntax

DioGetControlToPxiTriggerBusLineMode (*nHandle, nControl, pnLine, pbEnable, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board or File board handle.
nControl	SHORT	PXI Trigger Bus Control line: 0 DIO_TRIGGER_BUS_RUN_CTRL: Run signal to PXI Trigger Bus line. 1 DIO_TRIGGER_BUS_ARM_CTRL: Arm signal to PXI Trigger Bus line. 2 DIO_TRIGGER_BUS_EXT_CLK_CTRL: External Clock signal to PXI Trigger Bus line. 3 DIO_TRIGGER_BUS_EXT_STROBE_CTRL: External Strobe signal to PXI Trigger Bus line
pnLine	PSHORT	Returned control to PXI Trigger Bus Line: 0 DIO_PXI_TRIGGER_BUS_LINE0 1 DIO_PXI_TRIGGER_BUS_LINE1 2 DIO_PXI_TRIGGER_BUS_LINE2 3 DIO_PXI_TRIGGER_BUS_LINE3 4 DIO_PXI_TRIGGER_BUS_LINE4 5 DIO_PXI_TRIGGER_BUS_LINE5 6 DIO_PXI_TRIGGER_BUS_LINE6 7 DIO_PXI_TRIGGER_BUS_LINE7
pbEnable	PBOOL	Returned PXI Trigger Bus Line state: 0 The specified control to PXI Trigger Line is disabled. 1 The specified control to PXI Trigger Line is enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The DIO domain control signals are using the PXI Trigger Bus lines to distribute those signals.

When using internal clock and strobe only two PXI Trigger Bus lines have to be assigned for the Run and Arm control signals.

Additional lines will be required to be assigned for the external clock and/or external strobe.

The assigned lines will be saved to the INI file to be automatically loaded and applied whenever the DIO is initialized.

By default the DIO will use PXI Trigger Bus lines 0 and 1 for the Run and Arm signals.

NOTE: The same PXI Trigger Bus line will be used by all DIO boards (Master and Slaves) in the domain.

Example

The following example returns the specified control signal to PXI Trigger Line and its state:

```
SHORT   nStatus, nLine;
BOOL    bEnable;

DioGetControlToPxiTriggerBusLineMode(nMasterHandle, nControl, &nLine, &bEnable, &nStatus);
```

See Also

DioSetupControlToPxiTriggerBusLineMode, DioGetErrorString

DioGetCounterOverflowMode

Applies To

GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Returns the specified board program counter overflow mode.

Syntax

DioGetCounterOverflowMode (*nMasterHandle, pnOverflowMode, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board or File board handle.
pnOverflowMode	PSHORT	Program counter over flow modes are: 0 Overflow disabled. 1 Overflow enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the overflow mode is enabled, then the sequencer will not stop at the last step, but instead wrap around to step zero. If the overflow mode is disabled, then the sequencer will go to HALT state at the last step of the memory.

Example

The following example enables the counter to over flow mode and returns the overflow mode setting:

```
SHORT   nStatus, nOverflowMode;
DioSetupCounterOverflowMode (nMasterHandle, 1, &nStatus);
DioGetCounterOverflowMode (nMasterHandle, &nMode, &nStatus);
```

See Also

DioSetupCounterOverflowMode, DioSetupSequencerMode, DioGetSequencerMode, DioGetErrorString

DioGetDriverSummary

Applies To

General information function.

Purpose

Returns the driver version and information.

Syntax

DioGetDriverSummary (*pszSummary, nCount, pdwVersion*)

Parameters

Name	Type	Comments
pszSummary	LPSTR	Returns manufacturer's name and model separated by a comma.
nCount	SHORT	Maximum length of *pszSummary*.
pdwVersion	PDWORD	Returned board's version; major in high-order WORD, minor in low-order WORD.

Example

The following example returns the driver summary:

```
CHAR    szVer[128];
DWORD   dwVer;
DioGetDriverSummary (szVer, sizeof (szVer), &dwVer);
printf(%s, major=%ld, minor=%ld", szVer, dwVer >> 16, dwVer & 0xFFFFul);
```

See Also

DioGetErrorString

DioGetErrorString

Applies To

General information function

Purpose

Returns the error description for an error number.

Syntax

DioGetErrorString (*nError, pszError, nCount*)

Parameters

Name	Type	Comments
nError	SHORT	Error number, *pnStatus*, as returned from other DIO functions.
pszError	LPSTR	Error buffer
nCount	SHORT	Maximum length of error buffer. The buffer minimum length must be 128.

Comments

Appendix B contains error codes and returned strings.

The *nError* number is returned in *pnStatus* when a function is called.

Example

The following example prints the error string if the **DioSetupInitialization** function failed:

```
CHAR szError[128];
DioSetupInitialization (0, 1, 0x0200, &nDensity, &nBanks, &nMasterHandle, &nStatus);
if (nStatus<0);
{   DioGetErrorString(nStatus, szError, sizeof (szError));
    printf(szError);
    return;
}
```

DioGetExternalInputsStates

Applies To

GX5150, File

Purpose

Returns the External Inputs States.

Syntax

DioGetExternalInputsStates *(nMasterHandle, pdwInputsStates, pnStatus)*

Parameters

Name	Type	Comments
nMsterHandle	SHORT	Master or File board handle.
pdwInputsStates	PDWORD	Returns the External Inputs States. See comments for details.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function allows monitoring of the different External Inputs States when each bit represents a specified External Inputs State as follows:

Bit #	Description
0	External clock enable monitor: 0-input is not active (input logic high), 1-input is active (input logic low).
1	External strobe enable monitor: 0-input is not active (input logic high), 1-input is active (input logic low).
2	External trigger input signal: 0-input is not active (input logic high), 1-input is active (input logic low).
3	External pause input signal: 0-input is not active (input logic high), 1-input is active (input logic low).
4	Star Trigger input Line: 0-input is not active (input logic high), 1-input is active (input logic low).
5-7	Not used, read as zeros
8-11	External Output enable input signals: 0-input is not active (input logic high), 1-input is active (input logic low).
12-15	Not used, read as zeros
16-23	PXI Trigger Bus lines input signals: 0-input is not active (input logic high), 1-input is active (input logic low).
24-31	Not used, read as zeros

Example

The following example e returns the External Inputs States:

```
DWORD   dwInputsStates;
DioGetExternalInputsStates (nHandle, &dwInputsStates, &nStatus);
```

See Also

DioGetErrorString

DioGetExternalJumpATriggerMode

Applies To

GX5150, File

Purpose

Returns the trigger mode for the external Jump A line.

Syntax

DioGetExternalJumpATriggerMode *(nMasterHandle, pnTriggerMode, pnStatus)*

Parameters

Name	Type	Comments
nMsterHandle	SHORT	Master or File board handle.
pnTriggerMode	PSHORT	Trigger modes are as follow: 0. Disable external Jump A. 1. Enable external Jump A to trigger when low level is present. 2. Enable external Jump A to trigger when low to high transient occurs.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the selection is edge, then the rising edge of the external signal will create the event. If the selection is level, then a low level will enable the rising edge internal clock to create the event.

This function applies only for "External Jump A" line located on the I/O Control connector.

The board must be in either the HALT or PAUSE state.

Example

The following example enables an external Jump A for level sense mode. A constant low level will cause a continuous execution of Jump A. Calls the **DioGetJumpATriggerMode** function to verify the setting.

```
SHORT  nStatus, nMode;
DioSetupExternalJumpATriggerMode (nHandle, 2, &nStatus);
DioGetExternalJumpATriggerMode (nHandle, &nMode, &nStatus);
```

See Also

DioSetupExternalJumpATriggerMode, DioGetErrorString

DioGetExternalPauseAndTriggerMode

Applies To

GX5150, File.

Purpose

Returns the external PAUSE and TRIG lines mode.

Syntax

DioGetExternalPauseAndTriggerMode *(nMasterHandle, pnTriggerMode, pnStatus)*

Parameters

Name	Type	Comments
nMsterHandle	SHORT	Master or File board handle.
pnTriggerMode	PSHORT	Trigger modes are as follow: 0. External PAUSE and TRIG lines trigger when low level is present (default). 1. External PAUSE and TRIG lines trigger when low to high transient occur.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the selection is edge, then the rising edge of the external signal will create the event. If the selection is level, then a low level will enable the rising edge internal clock to create the event.

This function applies to the External PAUSE and External TRIG lines located on the Timing connector.

Example

The following example sets the External PAUSE and TRIG lines to trigger when low to high transient occurs, call the **DioGetExternalPauseAndTriggerMode** function to verify the setting.

```
SHORT  nStatus, nMode;
DioSetupExternalPauseAndTriggerMode (nHandle, 1, &nStatus);
DioGetExternalPauseAndTriggerMode (nHandle, &nMode, &nStatus);
```

See Also

DioSetupExternalJumpATriggerMode, DioGetErrorString

DioGetExternalRefClkFrequency

Applies To

GX5280, GX5290, GX5290e, GX5295, file

Purpose

Returns the External reference clock frequency.

Syntax

DioGetExternalRefClkFrequency (*nHandle, nClock, pdwFrequency, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Master or File board handle.
nClock	SHORT	Not used, set to 0.
pdwFrequency	PDWORD	External reference clock frequency
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The user can provide an external reference clock to the master DIO's on-board PLL. The driver then will use that value when programming the on-board PLL different frequencies.

In order to use that feature the clock source needs to be set to CLK_PRG_EXTERNAL_STROBE_INTERNAL.

The external reference clock range is 7.5MHz to 15MHz.

Example

The following example returns the External reference clock frequency value:

```
SHORT   nStatus;
DWORD   dwFreq;
DioGetExternalRefClkFrequency (nMasterHandle, 0, &dwFreq, &nStatus);
```

See Also

DioSetupExternalRefClkFrequency, DioSetupFrequency, DioSetupClkStrobeSource, DioGetClkStrobeSource, DioGetErrorString

DioGetFrequency

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the current programmed frequency.

Syntax

DioGetFrequency *(nMasterHandle, pdwFrequency, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
pdwFrequency	PDWORD	Return frequency range. GC5050: 5Hz to 60MHz. GX5050: 5Hz to 60MHz. GX5055: 5Hz to 50MHz. GX5150: 5Hz to 60MHz. GX5281: 5Hz to 50MHz. GX5282: 5Hz to 100MHz. GX5283: 5Hz to 200MHz. GX5291: 5Hz to 100MHz. GX5292: 5Hz to 100MHz. GX5293: 5Hz to 200MHz. GX5291e: 5Hz to 100MHz. GX5292e: 5Hz to 100MHz. GX5293e: 5Hz to 200MHz. GX5295: 5Hz to 100MHz.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5150:

After calling **DioReset** or power up, the internal clock is set to 10 MHz.

GC5050, GX5050, GX5055, GX5280, GX5290, GX5290e, GX5295:

After calling **DioReset** or power up, the internal clock is set to 10 MHz.

GX5293/GX5293e: for frequencies above 100MHz the user must set the number of active I/O channels (wide) to be 16 or less, see the **DioSetupIOConfiguration** function for more details.

GX5283/GX5293/GX5293e: for frequencies above 100MHz all clock and strobe delays have to be between 0 and 3nSec, see **DioSetupClkStrobeDelay** function for more details.

GX5293/GX5293e: for frequencies above 100MHz the user must set the number of active I/O channels (wide) to be 16 or lees, see the **DioSetupIOConfiguration** function for more details.

Note: using driver v4.0 (Build 48) and above frequency can be set regardless of the board's state (PAUSE/HALT/RUN).

Example

The following example returns the frequency value from a Master board specified by the board handle:

```
SHORT   nStatus;
DWORD   dwFrequency;
DioGetFrequency(nMasterHandle, &dwFrequency, &nStatus);
```

See Also

DioSetupFrequency, DioGetErrorString

DioGetGroupsStaticModeOutputState

Applies To

GX5150

Purpose

Returns all groups' outputs states when in static mode.

Syntax

DioGetGroupsStaticModeOutputState *(nHandle, pnGroupsState, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
pnGroupsState	PSHORT	Each bit represents Groups' output mode when in Static Mode: Enables or disables the outputs of all 4 I/O channels groups when in static mode, i.e. when reading and writing to the I/O pins statically. Each bit represents a group of 8 channels as follows: 0. Group 0 (channels 0-7): 0- Channels outputs are enabled, 1—Channels outputs are disabled. 1. Group 1 (channels 8-15) 0- Channels outputs are enabled, 1—Channels outputs are disabled. 2. Group 2 (channels 16-23) 0- Channels outputs are enabled, 1—Channels outputs are disabled. 3. Group 3 (channels 24-31) 0- Channels outputs are enabled, 1—Channels outputs are disabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in a HALT or PAUSE state in order for the user to read the I/O lines.

If the memory I/O configuration width was set to 16-bit or 8-bit, then only the 16 or 8 lower bits are applicable and the remaining bits return 0.

The DIO is operating in Static Mode whenever using **DioWriteIOPinsValue or DioReadIOPinsValue** functions.

Example

The following example returns all groups' outputs states when in static mode:

```
SHORT nGroupsState, nStatus;
DioGetGroupsStaticModeOutputState (nHandle, &nGroupsState, &nStatus);
```

See Also

DioSetupGroupsStaticModeOutputState, DioWriteIOPinsValue, DioReadIOPinsValue, DioSetupIOConfiguration, DioGetIOConfiguration, DioGetErrorString

DioGetInputDataSource

Applies To

GX5295, File.

Purpose

Returns the input data comparator source.

Syntax

DioGetInputDataSource (*nHandle, nChannel,pnInputDataSource, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannel	SHORT	Specified channel in the DIO board to apply the settings: • I/O channels: 0-31. • Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
pnInputDataSource	PSHORT	Input data comparator can be as follows: 0 DIO_LOW_THRESHOLD_COMPARATOR: data will be the result of the comparison done on the input data by the low threshold comparator, logic levels are as follows: Logic Low: whenever the input signal is below the low comparator threshold. Logic High: whenever the input signal is above the low comparator threshold 1 DIO_HIGH_THRESHOLD_COMPARATOR: data will be the result of the comparison done on the input data by the high threshold comparator, logic levels are as follows: Logic Low: whenever the input signal is below the high comparator threshold. Logic High: whenever the input signal is above the high comparator threshold.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Example

The following example returns the input data comparator of channel 0:

```
SHORT nInputDataSource;
DioGetInputDataSource (nHandle, 0, &nInputDataSource, &nStatus);
```

See Also

DioSetupInputDataSource, DioSetupInputLoadCurrent , DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioGetInputLoadCurrent

Applies To

GX5055, GX5295, File.

Purpose

Returns the specified channel input source and sink load currents.

Syntax

DioGetInputLoadCurrent (*nHandle, nChannel, pdISink, pdISource, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannel	SHORT	GX5055: Specified channel in the DIO board, channel can be 0-31. GX5295: Specified channel in the DIO board to apply the settings: • I/O channels: 0-31. • Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
pdISink	PDOUBLE	Input channel constant current sink value in mA. Input channel constant current sink value can be set from 0mA to 24mA with 0.3662 μA resolution. GX5055: The input channel current sink force the specified constant current to be active when the input voltage is below the low commutating voltage value. GX5295: The input channel current sink force the specified constant current to be active when the input voltage is below the commutating voltage value.
pdISource	PDOUBLE	Input channel constant current source value. Input channel constant current source value can be set from 0mA to 24mA with 0.3662 μA resolution. GX5055: The input channel current source force the specified constant current to be active when the input voltage is above the high commutating voltage value. GX5295: The input channel current source force the specified constant current to be active when the input voltage is above the commutating voltage value.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5055:

Each channel has an independent load with the following capabilities:

- Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
- Maintain high impedance over a wide voltage.
- Separate high and low commutating voltages – **DioSetupInputLoadCommutatingVoltage** function.
- Channels' Input Resistive load options - **DioSetupInputLoadResistance** function.

With independent high and low commutating voltages, the source and sink currents each have their own threshold voltage. If the voltage on the input, when the load is activated, is between the two commutating voltages, the load will remain in a high impedance state.

GX5295:

Each channel has an independent load with the following capabilities:

- Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
- Maintain high impedance over a wide voltage.
- Single commutating voltage – **DioSetupInputLoadCommutatingVoltage** function.

With a single commutating voltage, the source and sink currents have a single threshold voltage.

Channels input source and sink constant currents can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example returns the Input constant current settings for channel 0 in mA:

```
DOUBLE dISource, dISink;
DioGetInputLoadCurrent (nHandle, 0, &dISink, &dISource, &nStatus);
```

See Also

DioSetupInputLoadState, DioSetupInputLoadCurrent , DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioGetInputLoadCommutatingVoltage

Applies To

GX5055, GX5295, File.

Purpose

Returns the specified channel input constant current commutating voltage.

Syntax

DioGetInputLoadCommutatingVoltage (*nHandle, nChannel, pdVComHi, pdVComLo, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannel	SHORT	GX5055: Specified channel in the DIO board, channel can be 0-31. GX5295: Specified channel in the DIO board to apply the settings: • I/O channels: 0-31. • Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
pdVComHi	PDOUBLE	GX5055: Input active load sink and source currents high commutating voltages value, voltage settings and range depends on supply rail voltages see comments for details. GX5295: Input active load sink and source currents commutating voltages value, voltage can be set from --2V to +7V.
pdVComLo	PDOUBLE	GX5055: Input active load sink and source currents low commutating voltages value, voltage settings and range depends on supply rail voltages see comments for details. GX5295: Not used should be set to 0.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5055:

Each channel has an independent load with the following capabilities:

- Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
- Maintain high impedance over a wide voltage.
- Separate high and low commutating voltages – **DioSetupInputLoadCommutatingVoltage** function.
- Channels' Input Resistive load options - **DioSetupInputLoadResistance** function.

Min/Max commutating voltages:

	Min	Max
pdVComHi	Low Rail Supply +2V	High Rail Supply -7V
pdVComLo	Low Rail Supply +2V	High Rail Supply -7V

E.g. if high rail supply = 18V and low rail supply = -14V then min high commutating voltage >=-11V and max high commutating voltage <=12V.

With independent high and low commutating voltages, the source and sink currents each have their own threshold voltage. If the voltage on the input, when the load is activated, is between the two commutating voltages, the load will remain in a high impedance state.

GX5295:

Each channel has an independent load with the following capabilities:

- Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
- Maintain high impedance over a wide voltage.
- Single commutating voltage – **DioSetupInputLoadCommutatingVoltage** function.

With a single commutating voltage, the source and sink currents have a single threshold voltage.

Channels input constant current commutating voltage can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example returns the input constant current commutating voltage for channel 0:

```
DOUBLE dVComHi, dVComLo;
DioGetInputLoadCommutatingVoltage(nHandle, 0, &dVComHi, &dVComLo, &nStatus);
```

See Also

DioSetupInputLoadState, DioSetupInputLoadCurrent, DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioGetInputLoadResistance

Applies To

GX5055, File.

Purpose

Returns the specified channel input pull-up and pull down resistive loads.

Syntax

DioGetInputLoadResistance (*nHandle, nChannel, pdPullup, pdPulldown, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
pdPullup	PDOUBLE	Input pull-up resistive loads can be one of the following: • DIO_RESISTIVE_LOAD_240_OHMS = 240 • DIO_RESISTIVE_LOAD_290_OHMS = 290 • DIO_RESISTIVE_LOAD_1K_OHMS = 1000 • DIO_RESISTIVE_LOAD_OPEN = -1
pdPulldown	PDOUBLE	Input pull down resistive loads can be one of the following: • DIO_RESISTIVE_LOAD_240_OHMS = 240 • DIO_RESISTIVE_LOAD_290_OHMS = 290 • DIO_RESISTIVE_LOAD_1K_OHMS = 1000 • DIO_RESISTIVE_LOAD_OPEN = -1
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each channel has an independent load with the following capabilities:
0. Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
1. Maintain high impedance over a wide voltage.
2. Separate high and low commutating voltages – **DioSetupInputLoadCommutatingVoltage** function.
3. Channels' Input Resistive load options - **DioSetupInputLoadResistance** function.

With independent high and low commutating voltages, the source and sink currents each have their own threshold voltage. If the voltage on the input, when the load is activated, is between the two commutating voltages, the load will remain in a high impedance state.

The input resistive load is useful in applications with very low DUT output swings (where a traditional active load will not switch on and off completely or quickly) and also as a means of forcing the DUT to a known voltage when the DUT is in HiZ.

NOTE: The source and sink currents should be programmed to 0 during normal operation of the resistive load.

When the resistive load is placed in HiZ (DIO_RESISTIVE_LOAD_OPEN) it maintains a low leakage current when the input voltage is between the supply rails.

Channels input pull-up and pull down resistive loads can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example returns the input pull-up and pull down resistive loads for channel 0:

```
DOUBLE  dPullup, dPulldown;
SHORT nStatus;
DioGetInputLoadResistance (nHandle, 0, &dPullup, &dPulldown, &nStatus);
```

See Also

DioSetupInputLoadState, DioSetupInputLoadCurrent , DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioGetInputLoadState

Applies To

GX5055, GX5295, File.

Purpose

Returns the specified channel input load states.

Syntax

DioGetInputLoadState (*nHandle, nChannel, pnState, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
pnState	PSHORT	Returned load states: 0. DIO_LOAD_DISCONNECTED: Load is not connected. 1. DIO_LOAD_CONNECTED: Load is connected.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5055:

The function enables all the loads to be applied to the specified input channels (constant source and the resistive load). When the load is enabled there is a ~40Ohm resistor in between the input And the loads.

Each channel has an independent load with the following capabilities:
1. Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
2. Maintain high impedance over a wide voltage.
3. Separate high and low commutating voltages – **DioSetupInputLoadCommutatingVoltage** function.
4. Channels' Input Resistive load options - **DioSetupInputLoadResistance** function.

With independent high and low commutating voltages, the source and sink currents each have their own threshold voltage. If the voltage on the input, when the load is activated, is between the two commutating voltages, the load will remain in a high impedance state.

GX5295:

Each channel has an independent load with the following capabilities:
- Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
- Maintain high impedance over a wide voltage.
- Single commutating voltage – **DioSetupInputLoadCommutatingVoltage** function.

With a single commutating voltage, the source and sink currents have a single threshold voltage. Channels input source and sink constant currents can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example returns the Input load state for channel 0:

```
SHORT nState;
DioGetInputLoadState (nHandle, 0, &nState, &nStatus);
```

See Also

DioSetupInputLoadState, DioSetupInputLoadCurrent , DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioGetInputThresholdVoltages

Applies To

GX5055, GX5295, File.

Purpose

Returns the specified channel input low and high threshold voltages.

Syntax

DioGetInputThresholdVoltages (*nHandle, nChannel, pdHiLevel, pdLoLevel, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannel	SHORT	GX5055: Specified channel in the DIO board, channel can be 0-31. GX5295: Specified channel in the DIO board to apply the settings: • I/O channels: 0-31. • Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
pdHiLevel	PDOUBLE	GX5055: Input high voltage threshold, voltage settings and range depends on supply rail voltages see comments for details. GX5295: Input high voltage threshold, value can be -2.0V to +7.0V and higher then Input low voltage threshold.
pdLoLevel	PDOUBLE	GX5055: Input low voltage threshold, voltage settings and range depends on supply rail voltages see comments for details. GX5295: Input low voltage threshold, value can be -2.0V to +7.0V and lower then Input high voltage threshold.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5055:

The input voltage can be inside one of the following ranges:

0. Input voltage is higher than high voltage threshold, input is logic high. Data will be logged as logic high to the input memory and 0 to the invalid logic level input memory.
1. Input voltage is lower than low voltage threshold, input is logic low. Data will be logged as logic low to the input memory and 0 to the invalid logic level input memory.
2. Input voltage is higher than low voltage threshold and lower then high voltage threshold, input is invalid. Data will be logged as logic low to the input memory and 1 to the invalid logic level input memory.

Min/Max threshold voltages:

	Min	Max
pdHiLevel	Low Rail Supply +2V	High Rail Supply -7V
pdLoLevel	Low Rail Supply +2V	High Rail Supply -7V

E.g. if high rail supply = 18V and low rail supply = -14V then min high threshold voltage >=-11V and max high threshold voltage <=12V.

GX5295:

The board can only save the input data for all input channels of one of the two input threshold. The user can specify the desired input threshold data to be saved by calling the **DioSetupInputDataSource** function.

Input data source was set to the low input threshold:

0. Input voltage is higher than low voltage threshold, input is logic high. Data will be logged as logic high to the input memory.
1. Input voltage is lower than low voltage threshold, input is logic low. Data will be logged as logic low to the input memory

Input data source was set to the high input threshold:

0. Input voltage is higher than high voltage threshold, input is logic high. Data will be logged as logic high to the input memory.
1. Input voltage is lower than high voltage threshold, input is logic low. Data will be logged as logic low to the input memory

Channels input low and high threshold voltages can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example returns channel 0 input threshold voltages:

```
DOUBLE  , pdHiLevel, pdLoLevel;
DioGetInputThresholdVoltages (nHandle, 0, &dHiLevel, &dLoLevel, &nStatus);
```

See Also

DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioGetInputInterface

Applies To

GX5280, GX5290, GX5290e, File.

Purpose

Returns the Input Interface.

Syntax

DioGetInputInterface (*nHandle, pnInterface, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
pnInterface	PSHORT	Input Active Interface: 0 DIO_IO_INTERFACE_TTL: Input TTL connector is the active interface. 1 DIO_IO_INTERFACE_LVDS: Input LVDS connector is the active interface.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function selects the active input connector LVDS or TTL for each group of channels when in Input mode.

Example

The following example returns the Input Active Interface:

```
SHORT  nInterface;
DioGetInputInterface (nHandle, &nInterface, &nStatus);
```

See Also

DioSetupInputInterface, DioGetErrorString

DioGetInterfaceStandardLevel

Applies To

GX5280, GX5290, GX5290e, File

Purpose

Returns the TTL interface standards level.

Syntax

DioGetInterfaceStandardLevel (*nHandle, pnStandardLevel, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
pnStandardLevel	PSHORT	TTL Interface standards level: 0 DIO_INTERFACE_STANDARD_TWO_POINT_FIVE_VOLTS: Default interface level standard of 2.5V. 1 DIO_INTERFACE_STANDARD_THREE_POINT_THREE_VOLTS: Interface level standard of 3.3V. 2 DIO_INTERFACE_STANDARD_ONE_POINT_EIGHT_VOLTS: Interface level standard of 1.8V. 3 DIO_INTERFACE_STANDARD_ONE_POINT_FIVE_VOLTS: Interface level standard of 1.5V.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function configures the board to operate according to the specified level standard in order to interface with different TTL standards. All I/O data lines as well external input line such as external trigger, clock pause will operate according to the configured standards level.

Example

The following example returns the board interface TTL standards:

```
DioGetInterfaceStandardLevel (nHandle, &nStandardLevel, &nStatus);
```

See Also

DioSetupChannelsVoltageLevel, DioConfigureInterfaceStandardLevel, DioGetChannelsVoltageLevel, DioGetErrorString

DioGetIOConfiguration

Applies To

GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the I/O memory width and direction.

Syntax

DioGetIOConfiguration *(nHandle, pnWidth, pnDirection, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or File board handle.
pnWidth	PSHORT	**GX5150**: 0. 32 bits wide 1. 16 bits wide 2. 8 bits wide **GX5283:** 0 32 bits wide, maximum number of steps is 128M, channels 0-31 1 16 bits wide, maximum number of steps is 256M, channels 0-15 2 8 bits wide, maximum number of steps is 512M, channels 0-7 3 4 bits wide, maximum number of steps is 1G, channels 0-3 4 2 bits wide, maximum number of steps is 2G, channels 0-1 5 1 bit wide, maximum number of steps is 4G, channel 0 **GX5292/GX5293/GX5292e/GX5293e/GX5295:** 0 32 bits wide, maximum number of steps is 64M, channels 0-31 1 16 bits wide, maximum number of steps is 128M, channels 0-15 2 8 bits wide, maximum number of steps is 256M, channels 0-7 3 4 bits wide, maximum number of steps is 512G, channels 0-3 4 2 bits wide, maximum number of steps is 1G, channels 0-1 5 1 bit wide, maximum number of steps is 2G, channel 0

pnDirection	PSHORT	GX5150: Selects whether the I/O Memory configured as input or output. 0 Input mode. 1 Output mode. GX5280: I/O Group channel direction bits 0-3. Each bit represents the direction of Group of 8 channels. 0. Input mode. 1. Output mode. GX5290/GX5290e: Not applicable.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in HALT state before calling this function.

The I/O memory width determines the number of I/O pins used for all steps for the specified board.

All not used I/O channels pins will be set to Tri-state.

The last setting of the I/O configuration is saved to the DIO.INI file, to be used whenever calling **DioInitialize**.

GX5150: If **Frequency Doubler** I/O module is installed, then the function will check if the current setting is valid. If width was set to 2, then the function returns an error. Valid setting for the **Frequency Doubler** I/O module width are 1 (16 channels) and 2-(8 channels).

GX5283/GX5292/GX5293/GX5292e/GX5293e/GX5295:

The I/O memory width determines the number of I/O pins used for all steps for the specified board. If the I/O Memory width was set to Word or Byte, the remaining I/O pins will be set to Tri-state. Calling this function will reset the all the DIO boards in the domain and will clear all the commands.

GX5293/GX5293e: for frequencies above 100MHz the user must set the number of active I/O channels (wide) to be 16 or less, see the **DioSetupIOConfiguration** function for more details.

The last setting of the I/O configuration is saved to the GTDIO.INI file, to be used whenever calling **DioInitialize**.

Example

The following example sets the I/O memory width to 32-bit and the direction to input:

```
SHORT  pnWidth, pnDirection, pnStatus;
DioSetupIOConfiguration (nHandle, &pnWidth, &pnDirection, &pnStatus):
```

See Also

DioSetupIOConfiguration, DioGetErrorString

DioGetIOModuleType

Applies To

GC5050, GX5050, GX5150

Purpose

Returns the type of the I/O module currently connected.

Syntax

DioGetIOModuleType *(nHandle, pnType, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
pnType	PSHORT	Return types are as follow: 0x10: Standard TTL. 0x20: Frequency-Doubler. 0x30: Level Shifter. 0x40: PECL-TTL Translation. 0x50: Real Time Comparison 0x60: Low Voltage Differential Signaling (LVDS). 0xFF: No I/O module present.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Example

The following example returns the I/O module type:

```
SHORT   nStatus, nType;
DioGetIOModuleType(nHandle, 0, &nType, &nStatus)
```

See Also

DioGetErrorString

DioGetIOPinsStaticDirection

Applies To

GX5055, GX5295

Purpose

Returns the I/O pins channels direction when in static mode.

Syntax

DioGetIOPinsStaticDirection *(nHandle, pdwDirection, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
pdwDirection	PDWORD	The 32 I/O channels directions are represented by the 32-bits. Each bit represents a channel starting from 0, e.g. channel zero is represented by bit 0. A bit high means the channel is set to output while a bit low means channel is set to input.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example returns the I/O pins channels direction when in static mode:

```
SHORT   nStatus;
DWORD   dwDirection;
DioGetIOPinsStaticDirection (nHandle, &dwDirection, &nStatus);
```

See Also

DioSetupIOPinsStaticDirection, DioReadIoPinsValidity, DioReadIOPinsValue DioWriteIOPinsValue, DioGetErrorString

DioGetJumpAddress

Applies To

GX5150, File

Purpose

Returns register A or B Jump address.

Syntax

DioGetJumpAddress (*nMasterHandle, nRegister, pdwJumpAddress, pnStatus*)

Parameters

Name	Type	Comments
NMasterHandle	SHORT	Master or File board handle.
NRegister	SHORT	0 Register A. 1 Register B.
PdwJumpAddress	PDWORD	Jump address.
PnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The jump address can be any step in the physical memory range.

The jump address must reflect the current Master board width setting. If the width were set to 16-bit, then the available address space is double what it would be if it were set to 32-bit. If the width were set to 8-bit, then the available address space is quadruple what it would be if it were set to 32-bit.

Example

The following example sets the register A jump address to 123 and then verifies the address by calling **DioGetJumpAddress**:

```
SHORT   nStatus;
DWORD   dwAddress;
DioSetJumpAddress(nMasterHandle, 0, 123, &nStatus);
DioGetJumpAddress(nMasterHandle, 0, &dwAddress, &nStatus);
```

See Also

DioGetJumpAddress, DioSetupIOConfiguration, DioWriteIOPinsValue, DioGetErrorString

DioGetMemoryBankSize

Applies To

GC5050, GX5050 GX5150

Purpose

Returns the specified memory bank density.

Syntax

DioGetMemoryBankSize *(nHandle, nBank, pnDensity, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or Slave board handle.
nBank	SHORT	Bank numbers are as follow: GX5150: 0-7: Data memory SIMMs, connectors SIMM1 through SIMM8. 8: Control memory, connector SIMM9. GC5050, GX5050: 0: Data memory SIMMs
pnDensity	PSHORT	Bank density value are as follow: 0. 256K by 32 (8MB). 1. 1M by 32 (32MB). 2. 2M by 32 (64MB). 3. 4M by 32 (128MB). 0xF Memory not installed.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example retrieves bank 0 density:

```
SHORT  nStatus, nDensity;
DioGetMemoryBankSize (nHandle, 0, &nDensity, &nStatus);
```

See Also

DioGetErrorString

DioGetNextCtrlCommandStep

Applies To

GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the next control command step number starting at the specified step number.

Syntax

DioGetNextCtrlCommandStep *(nHandle, pdwStep, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or Slave board handle.
pdwSteps	PDWORD	Returned the next control command step number starting at the specified step number. If Returned value is 0xFFFFFFFF (4294967295) then no more control commands were found.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Example

The following example returns next control command step number starting from step 100:

```
SHORT   nStatus;
DWORD   dwStep=100;
DioGetNextCtrlCommandStep (nHandle, &dwStep, &nStatus);
```

See Also

DioWriteCtrlCommand, DioReadCtrlCommand, DioGetErrorString

DioGetNumberOfSteps

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the specified DIO board number of steps.

Syntax

DioGetNumberOfSteps *(nHandle, pdwSteps, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or Slave board handle.
pdwSteps	PDWORD	Returned specified DIO board number of steps.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

In case of GX5150 the number of steps adjusted according to the width settings. See for details "**DioSetupIOConfiguration**".

Example

The following example returns DIO board number of steps:

```
SHORT   nStatus;
DWORD   dwSteps;
DioGetNumberOfSteps (nHandle, &dwSteps, &nStatus);
```

See Also

DioSetupIOConfiguration, DioGetErrorString

DioGetOperationMode

Applies To

GC5050, GX5050.

Purpose

Returns the current operation mode.

Syntax

DioGetOperationMode *(nHandle, pnMode, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or Slave board handle.
pnMode	PSHORT	Operation mode are: OP_MODE_NORMAL: 0x0 (default) OP_MODE_FAST_ARM: 0x10 OP_MODE_FAST_TRIG: 0x20 OP_MODE_FAST_HALT: 0x40
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function accelerates the execution of ARM and/or TRIG commands when the running in frequencies less then 200KHz. The command is executed immediately with no delays or testing.

Users can accelerate the execution time of the ARM, TRIG and HALT commands by setting (or combing) the operation modes.

Operation modes can be combined using Boolean OR operation.

Example

The following example sets the GC5050 operation mode for fast ARM and TRIG and then returned the operation mode:

```
SHORT  nStatus, nOutputState;
DioSetOperationMode (nHandle, OP_MODE_FAST_ARM | OP_MODE_FAST_TRIG, &nStatus);
DioGetOperationMode (nHandle, &nOutputState, &nStatus);
```

See Also

DioSetOperationMode, DioGetErrorString

DioGetOutputClocksState

Applies To

GX5280, GX5290, GX5290e, GX5295

Purpose

Returns the specified output clock pin state.

Syntax

DioGetOutputClocksState *(nHandle, nClock, pnState, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or Slave board handle.
nClock	SHORT	specified output clocks are: DIO_OUT_CLOCK – output clock (J3 pin 22). DIO_OUT_STROBE – output clock (J3 pin 24). DIO_OUT_B_CLOCK – output clock (J3 pin 20). DIO_OUT_LVDS_CLOCK – output clock (J4 pins 4 and 38).
pnState	PSHORT	States values are: 0. DIO_CLOCK_OUTPUT_DISABLE – specified clock output pin is disabled. 1. DIO_CLOCK_OUTPUT_ENABLE– specified clock output pin is enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The DIO needs to be in HALT mode before calling this function.

This function sets the specified output clock signal pin state to be enabled or disabled state.

NOTE: settings will be overwritten whenever calling : reset, settings main clock frequency or settings B clock frequency.

Example

The following example returns the output strobe signal state:

```
SHORT  nStatus, nState;
DioGetOutputClocksState (nHandle, DIO_OUT_STROBE, &nState, &nStatus);
```

See Also

DioSetupOutputClocksState, DioWriteIOPinsValue, DioReadIOPinsValue, DioGetErrorString

DioGetOutputDataFormat

Applies To

GX5055, File.

Purpose

Returns the specified channel output data format.

Syntax

DioGetOutputDataFormat (*nHandle, nChannel, pnDataFormat, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
pnDataFormat	PSHORT	Output data format can be as follows: 0. DIO_OUTPUT_DATA_FORMAT_NR: No Return 1. DIO_OUTPUT_DATA_FORMAT_R0: Return to Zero 2. DIO_OUTPUT_DATA_FORMAT_R1: Return to One 3. DIO_OUTPUT_DATA_FORMAT_RZ: Return to Hi-Z 4. DIO_OUTPUT_DATA_FORMAT_RC: Return to Complement
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

DIO_OUTPUT_DATA_FORMAT_NR: No Return, the output logic level stay either high or low for the duration of the clock period.

DIO_OUTPUT_DATA_FORMAT_R0: Return to Zero. The signal returns to zero between consecutive data clocks at 50% of the specified output frequency clock duty cycle. This takes place even if a number of consecutive 0's or 1's occur in the signal. The signal is self clocking and does not require any additional settings.

DIO_OUTPUT_DATA_FORMAT_R1: Return to One. The signal returns to one between consecutive data clocks at 50% of the specified output frequency clock duty cycle. This takes place even if a number of consecutive 0's or 1's occur in the signal. The signal is self clocking and does not require any additional settings.

DIO_OUTPUT_DATA_FORMAT_RZ: Return to Hi-Z .The signal returns to hi-z between consecutive data clocks at 50% of the specified output frequency clock duty cycle. This takes place even if a number of consecutive 0's or 1's occur in the signal. The signal is self clocking and does not require any additional settings.

DIO_OUTPUT_DATA_FORMAT_RC: Return to Complement (RC). In Return to Complement (also called Manchester code) each transmitted data bit has at least one transition at 50% of the specified output frequency clock duty cycle. It is, therefore, self-clocking, which means that a clock signal can be recovered from the encoded data. Return to Complement ensures frequent line voltage transitions, directly proportional to the clock rate, this helps clock recovery. Logic low is expressed by a low-to-high transition. Logic high is expressed by high-to-low transition. The transitions which signify logic high or low occur at the midpoint of a period, the direction of the mid-bit transition indicates the data.

NOTE: the specified channel data format will be applied to all the channels' steps that are set as outputs. Input data will not be formatted.

Channels data format can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example returns channel 0 input data format:

```
SHORT   nDataFormat;
DioGetOutputDataFormat (nHandle, 0, &nDataFormat, &nStatus);
```

See Also

DioSetupOutputDataFormat, DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioGetOutputOverCurrentEnable

Applies To

GX5055

Purpose

Returns the channels over-current enable states.

Syntax

DioGetOutputOverCurrentEnable (*nHandle, pdwOverCurrentEnable, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
pdwOverCurrentEnable	PDWORD	Over-current flag state, each bit corresponds to a channel i.e. bit 0 represents channel 0 and bit 31 represents channel 31. • A bit with logic high bit will enable the over current circuit, when output over current condition is detected, sink or source, it will automatically set to its output in Hi-Z. Enabling channels outputs which were set to Hi-Z as a result of over current can only be reset by calling **ResetOutputOverCurrentStates** function. • A bit with logic low will ignore over current condition and will not set it to Hi-Z.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

NOTE over current condition can only be cleared by calling **DioResetOutputOverCurrentStates**. After the over current is cleared the channel output driver will be active again.

Channels over current high and low flags states can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example reads back the over current protection flags:

```
DWORD    dwOverCurrentEnable;
DioGetOutputOverCurrentEnable (nHandle, &dwOverCurrentEnable, &nStatus);
```

See Also

DioSetupOutputOverCurrentEnable, DioSetupOutputDataFormat, DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioGetOutputOverCurrentStates

Applies To

GX5055

Purpose

Returns all 32 channels over current high and low flags states.

Syntax

DioGetOutputOverCurrentStates (*nHandle, nChannel, pdwOverCurrentHi, pdwOverCurrentLo, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
pdwOverCurrentHi	PDWORD	Channels over current high flags states, each bit corresponds to a channel, i.e. bit 0 represents channel 0 and bit 31 represents channel 31. A bit with logic high signals that an over current condition occurs on the output high voltage. A low signals that no over current occurs. Once over current condition has been detected the corresponded channels' driver is placed in HiZ until the flag is cleared.
pdwOverCurrentLo	PDWORD	Channels over current low flags states, each bit corresponds to a channel, i.e. bit 0 represents channel 0 and bit 31 represents channel 31. A bit with logic high signals that an over current condition occurs on the output low voltage. A low signals that no over current occurs. Once over current condition has been detected the corresponded channels' driver is placed in HiZ until the flag is cleared.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

NOTE over current condition can only be cleared by calling **DioResetOutputOverCurrentStates**. After the over current is cleared the channel output driver will be active again.

Channels over current high and low flags states can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example returns all 32 channels over current high and low flags states:

```
DWORD  dwOverCurrentHi, dwOverCurrentLo;
DioGetOutputOverCurrentStates (nHandle, &dwOverCurrentHi, &dwOverCurrentLo, &nStatus);
```

See Also

DioSetupOutputDataFormat, DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioGetOutputSlewRate

Applies To

GX5055, File.

Purpose

Returns the specified channel output rising and falling slew rate.

Syntax

DioGetOutputSlewRate (*nHandle, nChannel, pdRisingEdge, pdFallingEdge, pdBias, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
pdRisingEdge	PDOUBLE	Output rising slew rate values are 0.1V ns to 1.2V/ns.
pdFallingEdge	PDOUBLE	Output falling slew rate, values are 0.1V ns to 1.2V/ns.
pdBias	PDOUBLE	Returns the driver output bias current, values are 0-7. The driver output stage has a programmable bias current to allow applications that require slower edge rates to consume less power.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each channel's input high and low voltage threshold comparators have timing capability with the following characteristics:

- Each output driver has a separate and independent adjustments for the rising and falling slew rate
- Each output driver output stage has a programmable bias current to allow applications that require slower edge rates to consume less power.
- Separate and independent delay circuitry for each channel.

Channels output rising and falling slew rates can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example returns channel output rising and falling slew rate:

```
DOUBLE dRisingEdge, dFallingEdge, dBias;
DioGetOutputSlewRate (nHandle, 0, &dRisingEdge, &dFallingEdge, &dBias, &nStatus);
```

See Also

DioSetupOutputSlewRate, **DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString**

DioGetOutputState

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e

Purpose

Returns the current output pins state.

Syntax

DioGetOutputState *(nHandle, pnOutputState, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or Slave board handle.
pnOutputState	PSHORT	States values are: 0. Enabled-Outputs retain last value. 1. Disabled-Outputs are set to Tri-state. 2. Disabled on Halt/Pause-Outputs are set to Tri-state only when in HALT or PAUSE. 3. Outputs are set to zero. 4. Outputs are set to one (GX5280 only).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The current state does not have any effect on the DIO running mode or the loaded vector. The function can be called while the DIO is in the PAUSE, HALT or RUN state.

The state setting does not have any affect if the board is in input mode.

Example

The following example returns the current output state from a board specified by the board handle:

```
SHORT  nStatus, nOutputState;
DioGetOutputState (nHandle, &nOutputState, &nStatus);
```

See Also

DioSetupOutputState, DioSetupIOConfiguration, DioGetIOConfiguration, DioGetErrorString

DioGetOutputVoltages

Applies To

GX5055, GX5295, File.

Purpose

Returns the specified channel output driver high and low voltages.

Syntax

DioGetOutputVoltages (*nHandle, nChannel, pdHiLevel, pdLoLevel, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannel	SHORT	GX5055: Specified channel in the DIO board, channel can be 0-31. GX5295: Specified channel in the DIO board to apply the settings: • I/O channels: 0-31. • Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
pdHiLevel	PDOUBLE	GX5055: Output driver high voltage corresponding to logic high voltage settings and range depends on supply rail voltages see comments for details. GX5295: Output driver high voltage corresponding to logic high. Voltage can be set from -2V to +7V and must be greater than the output driver low voltage.
pdLoLevel	PDOUBLE	GX5055: Output driver low voltage corresponding to a logic low voltage settings and range depends on supply rail voltages see comments for details. GX5295: Output driver low voltage corresponding to a logic low. Voltage can be set from -2V to +7V and must be lower than the output driver high voltage.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each channels' output driver has two level driver, low and high level.
Channels output driver voltages can be read back and set dynamically at any time even while the DIO is running mode.

GX5055:

The total output driver voltage swing (Output driver high voltage less Output driver low voltage) is limited to 24V per channel.

Min/Max output driver high and low voltages:

	Min	Max
pdHiLevel	Low Rail Supply +5V	High Rail Supply -3V
pdLoLevel	Low Rail Supply +4V	High Rail Supply -7V
pdHiLevel - pdLoLevel	0.5V	

E.g. if high rail supply = 18V and low rail supply = -14V then min output driver high voltage >=-11V and max output driver high voltage <=12V.

Example

The following example returns channel 0 output driver high and low voltages:

```
DOUBLE  dHiLevel, dLoLevel;
DioGetOutputVoltages (nHandle, 0, &dHiLevel, &dLoLevel, &nStatus);
```

See Also

DioSetupOutputVoltages, DioSetupOutputSlewRate, **DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString**

DioGetOverTemperatureStatus

Applies To

GX5295

Purpose

Returns the over temperature flag state.

Syntax

DioGetOverTemperatureStatus *(nHandle, pdwOverTemp, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or Slave board handle.
pdwOverTemp	PDWORD	Over temperature flag state: 0. No channel had an over temperature condition. 1. One or more channels had an over temperature condition.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

All channels over temperature flag are Ored together to generate the over temperature flag state. If one or more channels had an over temperature condition the functions returns a ‘1’. The actual temperature of each channel can be measure by calling the **DioMeasure** function.

Example

The following example returns over temperature flag:

```
DWORD   dwOverTemp;
DioGetOverTemperatureStatus (nHandle, &dwOverTemp, &nStatus);
```

See Also

DioMeasure, DioGetErrorString

DioGetPauseCount

Applies To

GX5150, File.

Purpose

Returns the Pause count value.

Syntax

DioGetPauseCount (*nMasterHandle, pdwCount, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
pdwCount	PDWORD	Count value can be 0 to 4,294,967,295 (0 to 0xFFFFFFFF).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board will pause after executing the specified number of steps if the pause on step mode was previously enabled by calling the "DioSetPauseCounterMode" function.

The board must be in a HALT or PAUSE mode.

Example

The following example returns the PAUSE count:

```
SHORT   nStatus;
DWORD   dwCount;
DioGetPauseCount (nMasterHandle, &dwCount, &nStatus);
```

See Also

DioSetPauseCount, DioSetPauseCounterMode, DioGetPauseCounterMode, DioGetStepCounter, DioGetErrorString

DioGetPauseCounterMode

Applies To

GX5150, File.

Purpose

Returns Pause counter mode.

Syntax

DioGetPauseCounterMode (*nMasterHandle, pnMode, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
pnMode	PSHORT	Pause counter modes are: 0 Disabled. 1 Enabled. 2 Zero on Pause.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in a HALT or PAUSE mode.

When in “Zero on Pause” mode, the Pause counter will be set to zero each time the board is paused by the Pause counter.

Example

The following example returns the Pause counter mode:

```
SHORT   nStatus, nMode;
DioGetPauseCounterMod (nHandle, &nMode &nStatus);
```

See Also

DioSetPauseCounterMode, DioGetPauseCount, DioSetPauseCount, DioGetStepCounter, DioGetErrorString

DioGetPowerConnect

Applies To

GX5055

Purpose

Returns the front-end board power connect relay state

Syntax

DioGetPowerConnect (*nHandle, pnState, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Returned Handle (session identifier) that can be used to call any other operations of that resource
pnState	PSHORT	Returns the front-end power relay state. 0. DIO_POWER_DISCONNECT: Disconnect power 1. DIO_POWER_FROM_FRONT: Connect to front J7 Power Connector 2. DIO_POWER_FROM_BACKPLANE: Connect to the backplane for power
pnStatus	PSHORT	Returned status: 0 on success, negative value on failure.

Comments

The pin electronics require the rail voltages, (high rail voltage (VCC) and low rail voltage (VEE) to be present in order to operate correctly. These rails are connected by way of a relay switch to either the J7 front panel connector or to the backplane when using a special high powered chassis with the J5 connecter present. When setting the connection state to DIO_POWER_FROM_FRONT or DIO_POWER_FROM_BACKPLANE, the Vmid will be calculated based on what rail voltages are detected. The rail voltages should be present before connecting the power so the Vmid can be calculated correctly.

Example

The following example returns the power to the front panel J7 Connector:

```
SHORT nState, nStatus;

DioGetPowerConnect(nHandle, &nState, &nStatus);
```

See Also

DioSetPowerConnect, GtDio6xGetErrorString

DioGetPxiStarTriggerMode

Applies To

GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the board PXI Star Trigger input mode.

Syntax

DioGetPxiStarTriggerMode *(nMasterHandle, pnMode, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board or File board handle.
pnMode	PSHORT	PXI Star Trigger mode: 0. DIO_PXI_STAR_TRIGGER_DISABLED: the DIO star trigger input is disabled. 1. DIO_PXI_STAR_TRIGGER_TO_PAUSE: the star trigger input will pause the board. 2. DIO_PXI_STAR_TRIGGER_TO_TRIGGER: the DIO star trigger input will trigger the board.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When the DIO PXI Star Trigger is set to either trigger or pause the board the input signal can be programmed to respond to low level or rising edge. See for details **DioSetupSignalEdgeOrLevelMode**

Example

The following example returns the PXI Star Trigger mode:

```
SHORT nMode;
DioGetPxiStarTriggerMode (nMasterHandle, &nMode, &nStatus);
```

See Also

DioSetupPxiStarTriggerMode, DioSetupSignalEdgeOrLevelMode, **DioSetupPxiTriggerBusLineMode, DioGetErrorString**

DioGetPxiTriggerBusLineMode

Applies To

GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the PXI trigger bus to group output state.

Syntax

DioGetPxiTriggerBusLineMode *(nMasterHandle, nLine, pnMode, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board or File board handle.
nLine	SHORT	PXI Trigger Bus Line: 0 DIO_PXI_TRIGGER_BUS_LINE0 1 DIO_PXI_TRIGGER_BUS_LINE1 2 DIO_PXI_TRIGGER_BUS_LINE2 3 DIO_PXI_TRIGGER_BUS_LINE3 4 DIO_PXI_TRIGGER_BUS_LINE4 5 DIO_PXI_TRIGGER_BUS_LINE5 6 DIO_PXI_TRIGGER_BUS_LINE6 7 DIO_PXI_TRIGGER_BUS_LINE7
pnMode	PSHORT	Returned PXI Trigger Bus Line mode: 0 DIO_PXI_TRIGGER_BUS_LINE_MODE_DISABLE: the specified PXI Trigger Line is disabled. 1 DIO_PXI_TRIGGER_BUS_LINE_MODE_PAUSE_INPUT: the specified PXI Trigger Line is set to an input pause. 2 DIO_PXI_TRIGGER_BUS_LINE_MODE_PAUSE_OUTPUT: the specified PXI Trigger Line is set to as output pause. 3 DIO_PXI_TRIGGER_BUS_LINE_MODE_TRIGGER_INPUT: the specified PXI Trigger Line is set to as input trigger. 4 DIO_PXI_TRIGGER_BUS_LINE_MODE_TRIGGER_OUTPUT: the specified PXI Trigger Line is set to as output trigger.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function allows you to route trigger and paused signals to a specific PXI Trigger Bus line. The trigger and paused signals can be outputs to other Master DIO's or instruments that share the PXI Trigger Bus and/or inputs signals. The Trigger Bus Line assignments are mutually exclusive and can be set in tandem.

Example

The following example returns the PXI Trigger Bus Line number 1 mode:

```
SHORT nMode;
DioGetPxiTriggerBusLineMode (nMasterHandle, DIO_PXI_TRIGGER_BUS_LINE1, &nMode, &nStatus);
```

See Also

DioSetupPxiTriggerBusLineMode, DioGetErrorString

DioGetSequencerMode

Applies To

GX5150.

Purpose

Returns whether the specified board Sequencer is enabled or disabled.

Syntax

DioGetSequencerMode *(nMasterHandle, pnSequencerMode, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
pnSequencerMode	PSHORT	0 Sequencer disabled. 1 Sequencer enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in a **HALT** or **PAUSE** mode.

Disabling the sequencer causes the Master to disregard any control memory command or external **JumpA**. This may be useful if the content of the control memory is unknown, or for debugging.

Example

The following example returns the Sequencer mode from a board specified by the board handle:

```
SHORT  nStatus, nSequencerMode;
DioGetSequencerMode (nHandle, &nSequencerMode &nStatus);
```

See Also

DioSetupSequencerMode, DioGetErrorString

DioGetSignalEdgeOrLevelMode

Applies To

GX5280, GX5290, GX5290e, GX5295

Purpose

Returns the specified Signal edge or level active mode.

Syntax

DioGetSignalEdgeOrLevelMode (*nMasterHandle, nSignal, pnMode, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
nSignal	SHORT	Signal can be one of the following: 0 DIO_EXTERNAL_STROBE_ENABLE 1 DIO_EXTERNAL_CLOCK_ENABLE 2 DIO_EXTERNAL_PAUSE 3 DIO_EXTERNAL_TRIGGER 4 DIO_OUTPUT_RUN 5 DIO_OUTPUT_ARM 6 DIO_PXI_STAR_TRIGGER 7 DIO_PXI_TRIGGER_BUS See Comments below for details.
pnMode	PSHORT	Signal active mode can be one of the following: 0 DIO_SIGNAL_ACTIVE_LOW - signal is active when low level is present (default). 1 DIO_SIGNAL_ACTIVE_HIGH - signal is active when high level is present. 2 DIO_SIGNAL_ACTIVE_RISING_EDGE - signal logic when low to high transient occur. 3 DIO_SIGNAL_ACTIVE_FALLING_EDGE - signal is active when high to low transient occur. See Comments below for details.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function returns the current settings of the specified active mode for input control signals and output monitoring signals residing on the Timing connector. Each signal can be set as follows:

DIO_EXTERNAL_STROBE_ENABLE:

The external strobe enable input signal Active mode (Timing connector) can be set as one of the following:

0. DIO_SIGNAL_ACTIVE_LOW – signal is active when low level is present (default).
1. DIO_SIGNAL_ACTIVE_HIGH – signal is active when high level is present.

DIO_EXTERNAL_CLOCK_ENABLE:

The external clock enable input signal Active mode (Timing connector) can be set as one of the following:

0. DIO_SIGNAL_ACTIVE_LOW – signal is active when low level is present (default).
1. DIO_SIGNAL_ACTIVE_HIGH – signal is active when high level is present.

DIO_EXTERNAL_PAUSE:

The external pause enable input signal Active mode (Timing connector) can be set as one of the following:

0. DIO_SIGNAL_ACTIVE_LOW – signal is active when low level is present (default).
1. DIO_SIGNAL_ACTIVE_HIGH – signal is active when high level is present.
2. DIO_SIGNAL_ACTIVE_RISING_EDGE – signal logic when low to high transient occur.
3. DIO_SIGNAL_ACTIVE_FALLING_EDGE – signal is active when high to low transient occur.

DIO_EXTERNAL_TRIGGER:

The external trigger input signal Active mode (Timing connector) can be set as one of the following:

0. DIO_LOGIC_ACTIVE_LOW – signal is active when low level is present (default).
1. DIO_LOGIC_ACTIVE_HIGH – signal is active when high level is present.
2. DIO_LOGIC_ACTIVE_RISING_EDGE – signal logic when low to high transient occur.
3. DIO_LOGIC_ACTIVE_FALLING_EDGE – signal is active when high to low transient occur.

DIO_OUTPUT_RUN:

The output run signal Active mode (Timing connector) can be set as one of the following:

0. DIO_LOGIC_ACTIVE_LOW – signal is active when low level is present (default).
1. DIO_LOGIC_ACTIVE_HIGH – signal is active when high level is present.

DIO_OUTPUT_ARM:

The external arm input signal Active mode (Timing connector) can be set as one of the following:

0. DIO_LOGIC_ACTIVE_LOW – signal is active when low level is present (default).
1. DIO_LOGIC_ACTIVE_HIGH – signal is active when high level is present.

DIO_PXI_STAR_TRIGGER:

The external star trigger input signal Active mode (Timing connector) can be set as one of the following:

0. DIO_LOGIC_ACTIVE_LOW – signal is active when low level is present (default).
2. DIO_LOGIC_ACTIVE_RISING_EDGE – signal logic when low to high transient occur.

DIO_PXI_TRIGGER_BUS:

The external PXI trigger bus input signal Active mode (Timing connector) can be set as one of the following:

0. DIO_LOGIC_ACTIVE_LOW – signal is active when low level is present (default).
2. DIO_LOGIC_ACTIVE_RISING_EDGE – signal logic when low to high transient occur.

Example

The following example returns the external strobe enable input signal edge or level active mode:

```
SHORT nMode;
DioGetSignalEdgeOrLevelMode (nMasterHandle, DIO_EXTERNAL_STROBE_ENABLE, &nMode, &nStatus);
```

See Also

DioSetupSignalEdgeOrLevelMode, **DioSetupPxiTriggerBuslineMode, DioGetErrorString**

DioGetSkewDelay

Applies To

GX5055, GX5295, File.

Purpose

Returns the specified channel skew delay.

Syntax

DioGetSkewDelay (*nHandle, nChannel, nSource, nEdge, pdDelay, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannel	SHORT	GX5055: Specified channel in the DIO board, channel can be 0-31. GX5295: Specified channel in the DIO board to apply the settings: • I/O channels: 0-31. • Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
nSource	SHORT	Delay source can be as follows: 0 DIO_SKEW_DELAY_OUTPUT_DATA: add delay to the output driver data in reference to the DIO clock. 1 DIO_SKEW_DELAY_OUTPUT_DATA_ENABLE: add delay to the output driver data enable in reference to the DIO clock. Delaying the output data enable adds an additional delay output driver data in reference to the DIO clock. 2 DIO_SKEW_DELAY_INPUT_LOW_THRESHOLD: apply the delay to the input low voltage threshold comparator. 3 DIO_SKEW_DELAY_INPUT_HIGH_THRESHOLD: apply the delay to the input high voltage threshold comparator.
nEdge	SHORT	Delay edge can be as follows: 0. DIO_RISING_EDGE: signal rising edge. 1. DIO_FALLING_EDGE: signal falling edge.

pdDelay	PDOUBLE	GX5055 Output Rising Edge: Output skew delay can be set from 0 ns to +9.6875 ns with 312.5 ps resolution. GX5055 Input falling edge: Skew delay can be set from -2.5 ns to +2.1875 ns with 312.5 ps resolution. GX5295 Output Rising Edge: Output skew delay can be set from 0 ns to + 4.6875 ns with 19.53 ps resolution. The propagation delay circuitry adds timing delay to the rising edge and the falling edge in equal amounts. Propagation delay adjustment is typically used for aligning the timing of multiple channels inside a tester. GX5295 Output Falling Edge: Output falling edge skew delay can be set from -2.5 ns to +2.1875 ns with 19.53 ps resolution. The falling edge delay circuitry adds or subtracts timing delay to or from the falling edge while having no effect on the rising edge. Propagation delay adjustment is typically used for removing any pulse width distortion inside a tester. GX5295 Input Rising Edge: Input skew delay can be set from 0 ns to + 4.6875 ns with 19.53 ps resolution. The propagation delay circuitry adds timing delay to the rising edge and the falling edge in equal amounts. Propagation delay adjustment is typically used for aligning the timing of multiple channels inside a tester. GX5295 Input Falling Edge: Input falling edge skew delay can be set from -156.25 ps to +146.5 ps with 9.76 ps resolution. The falling edge delay circuitry adds or subtracts timing delay to or from the falling edge while having no effect on the rising edge. Propagation delay adjustment is typically used for removing any pulse width distortion inside a tester.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Output skew delays:

Each channel output has timing capability with the following characteristics:

- Separate and independent delay circuitry for the driver output data and the driver output data enable.
- Separate and independent delay circuitry for each channel.
- Propagation delay adjust (both rising and edge Tpd are delayed equally).

Input skew delays:

Each channel's input high and low voltage threshold comparators have timing capability with the following characteristics:

- Separate and independent delay circuitry for the high and low voltage threshold comparators.
- Separate and independent delay circuitry each channel.
- Propagation delay adjust (both rising and edge Tpd are delayed equally).

Channels output rising and falling edge skew delays can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example sets channel 0 delay edge skew of the output data to 2.0ns:

```
DOUBLE dDelay;
DioGetSkewDelay (nHandle, 0, DIO_OUTPUT_SKEW_DELAY_DATA, DIO_RISING_EDGE, &dDelay, &nStatus);
```

See Also

DioSetupSkewDelay, **DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString**

DioGetSlaveHandle

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Returns the Slave board handle of the specified Master board.

Syntax

DioGetSlaveHandle *(nMasterHandle, nBrdNum, pnHandle, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
nBrdNum	SHORT	Slave board number (1-7).
pnHandle	PSHORT	Returned Slave board handle.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function can be called only after **DioInitialize** is called to retrieve the handle of Slave board. If an error occurs, *pnHandle* is 0.

Example

The following example returns the handle for Slave board number 1:

```
SHORT nHandle, nStatus;
DioGetSlaveHandle (nMasterHandle, 1, &nHandle, &nStatus);
```

See Also

DioSetupInitialization, DioInitialize

DioGetStepCounter

Applies To

GX5150.

Purpose

Returns the current step counter value.

Syntax

DioGetStepCounter (*nMasterHandle, pdwStepCounter, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
pdwStepCounter	PDWORD	Step counter value can be 0 to 4,294,967,295 (0 to 0xFFFFFFFF).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The step counter increment each time a step is executed. The counter wraps back to zero when reaching the max value (0xFFFFFFFF).

Example

The following example returns the counter value:

```
SHORT   nStatus;
DWORD   dwCount;
DioGetStepCounter (nHandle, &dwCount, &nStatus);
```

See Also

DioSetPauseCounterMode, DioGetPauseCounterMode, DioGetPauseCount, DioSetPauseCount, DioGetErrorString

DioGetTermination

Applies To

GX5150, GX5280, GX5290, GX5290e

Purpose

Returns the specified board I/O Module/Group termination mode.

Syntax

DioGetTermination (*nHandle, nIOModuleCon, pnTermination, pnStatus*).

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
pnTermination	PSHORT	Termination mode values are as follow: GX5150: 0 Disable. 1 Enable. GX5280/GX5290/GX5290e: Returns bits 0-3 I/O Group Terminators enables or disables. Each bit represents Group0 thorough 3 each Group with 8 channels. Bit Low: Terminators Off Bit High: Terminators On
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function returns an error if the current I/O module does not supports the termination option. See the specific I/O module manual to check if it supports this option.

Example

The following example returns the termination state for a board specified by the board handle:

```
SHORT  nStatus, nTrmination;
DioGetTermination (nHandle, &nTrmination, &nStatus);
```

See Also

DioSetupTermination, DioGetIOModuleType, DioGetErrorString

DioGetTriggerDEvent

DioGetTriggerPEvent

DioGetTriggerTEvent

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the specified board D, P or T trigger mask and event registers values.

Syntax

DioGetTriggerDEvent *(nMasterHandle, pwEvent, pwMask, pnStatus)*

DioGetTriggerPEvent *(nMasterHandle, pwEvent, pwMask, pnStatus)*

DioGetTriggerTEvent *(nMasterHandle, pwEvent, pwMask, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
pwEvent	PWORD	Event registers value (0 default).
pwMask	PWORD	Mask registers value (0xFFFF default).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The **DioSetupTriggerMode** must be called to enable triggering using external event lines.

Example

The following example returns the D trigger register mask and event:

```
SHORT   nStatus;
WORD    wDEvent, wDMask, wPEvent, wPMask, wTEvent wTMask;
DioGetTriggerDEvent (nMasterHandle, &wDEvent, &wDMask, &nStatus);
DioGetTriggerPEvent (nMasterHandle, &wPEvent, &wPMask, &nStatus);
DioGetTriggerTEvent (nMasterHandle, &wTEvent, &wTMask, &nStatus);
```

See Also

DioSetupTriggerDEvent, DioSetupTriggerPEvent, DioSetupTriggerTEvent, DioSetupTriggerMode, DioGetTriggerMode, DioGetErrorString

DioGetTriggerMode

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the trigger mode setting.

Syntax

DioGetTriggerMode *(nMasterHandle, pnMode, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
pnMode	PSHORT	Trigger modes: 0 Conditional Trigger and Pause Disabled (default). 1 Triggers on D events. 2 Triggers on T events. 3 Triggers first on D events, then on T events. 4 Triggers first on T events, then on D events.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The **DioTrig** function or the external trigger control line will force a trigger event overriding the conditional trigger mode set by this function.

Available Trigger modes are:

0 Conditional trigger and pause events disabled. Sequencer commands using D Register disabled.

1 Enables conditional trigger and pause events. DIO creates trigger events when external event input lines ANDed with D mask register equals the D event register.

2 Enables conditional trigger and pause events. DIO creates trigger events when external event input lines ANDed with T mask register equals the T event register.

3 Enables conditional trigger and pause events. This is a two-condition trigger. The first condition is the external event input lines ANDed with the D mask register equals the D event register. The second condition is the external event input lines ANDed with T mask register equal the T event register. These are sequential conditions; the D trigger event must occur before the T Trigger event will cause a sequencer trigger event.

4 Enables conditional trigger and pause events. This is a two-condition trigger. The first condition is the external event input lines ANDed with the T mask register equals the T event register. The second condition is the external event input lines ANDed with D mask register equal the D event register. Those are sequential conditions. The T trigger event must take place before the D Trigger event will cause a sequencer trigger event.

Pause event control:

The Pause event control is enabled whenever the trigger mode is not in disabler mode the user need to follow the following guide lines in order to prevent conflicting conditions of trigger and pause. The PEvent value needs to be different then <>X Register (or the actual External Events lines if they are used) **AND** PEvent value needs to be different then the trigger event itself to prevent conflicting conditions. E.g. if the Trigger Mode was set to D, then the DEvent value need to be different then <> PEvent value.

Note: The board must be in the PAUSE state to enable triggering.

Example

The following example returns the trigger mode setting from a Master board specified by the board handle:

```
SHORT   nStatus, nMode;
DioGetTriggerMode (nMasterHandle, &nMode, &nStatus);
```

See Also

DioSetupTriggerMode, DioSetupTriggerDEvent, DioSetupTriggerPEvent, DioGetTriggerPEvent, DioSetupTriggerTEvent, DioGetTriggerTEvent, DioGetErrorString

DioGetTriggerXEventSource

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the specified Master board external event lines source.

Syntax

DioGetTriggerXEventSource (*nMasterHandle, pnSource, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
pnSource	PSHORT	Values are: 0 The source is external event lines (default). 1 The source is the X register.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function is used to simulate external event lines. When *nSource* is 1, the *X* register value is used instead of external event lines on trigger, pause and conditional commands in the running vector. **DioWriteXRegister** can be used to set the value of the *X* register.

Example

The following example returns the X register trigger source:

```
SHORT  nStatus, nSource;
DioGetTriggerXEventSource (nMasterHandle, &nSource, &nStatus);
```

See Also

DioSetupTriggerXEventSource, DioWriteXRegister, DioSetupTriggerMode

DioGetTriStateTerminationMode

Applies To

GX5295, File.

Purpose

Return the specified channel Tri-State termination mode.

Syntax

DioGetTriStateTerminationMode (*nHandle, nChannel, pnMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
pnMode	PSHORT	Returned Tri-State termination mode: 0 DIO_TRI_STATE_TERMINATION_MODE_DEFAULT: Default termination mode, in this mode the channel will be set to Hi-Z whenever the channel direction is set to input. 1 DIO_TRI_STATE_TERMINATION_MODE_LEVEL: in this mode the channel will be set to the voltage as specified in **DioSetupTriStateTerminationVoltage** API whenever the channel direction is set to input.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5295:

The specified termination voltage is the same voltage that is used by the input active load commutating voltage. Channels termination voltage can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example returns the specified channel Tri-State termination mode for channel 0:

```
SHORT nMode;
DioGetTriStateTerminationMode (nHandle, 0, &nMode, &nStatus);
```

See Also

DioSetupTriStateTerminationMode, DioSetupTriStateTerminationVoltage, **DioSetupInputLoadCurrent , DioSetupInputLoadCommutatingVoltage, DioGetErrorString**

DioGetTriStateTerminationVoltage

Applies To

GX5295, File.

Purpose

Return the specified channel Tri-State termination voltage.

Syntax

DioGetTriStateTerminationMode (*nHandle, nChannel, pdVoltage, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
pdVoltage	PDOUBLE	Returned Tri-State termination voltage. The specified termination voltage is the same voltage that is used by the input active load commuting voltage. Voltage range is -2V to +7V.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5295:

Channels termination voltage can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example returns the specified channel Tri-State termination voltage for channel 0:

```
DOUBLE dVoltage;
DioGetTriStateTerminationVoltage (nHandle, 0, &dVoltage, &nStatus);
```

See Also

DioSetupTriStateTerminationVoltage, **DioGetTriStateTerminationMode, DioSetupInputLoadCurrent , DioSetupInputLoadCommutatingVoltage, DioGetErrorString**

DioHalt

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Sets the specified Master board to HALT state.

Syntax

DioHalt (*nMasterHandle*, *pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The **DioHalt** function sets the board program counter value to zero. Calling **DioArm** and **DioTrig** after the board is in a HALT state will cause the board to continue running from the current program counter address.

Example

The following example sets the board to the HALT state:

```
SHORT   nStatus;
DioHalt (nMasterHandle, &nStatus);
```

See Also

DioArm, DioTrig, DioReadStatusRegister

DioInitialize

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Initializes a Master board and all Slave boards associated with it.

Syntax

DioInitialize *(nMasterNum, pnMasterHandle, pnStatus)*

Parameters

Name	Type	Comments
nMasterNum	SHORT	Master board number in the system (1-8).
pnMasterHandle	PSHORT	Returned Master handle, 0 for error.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The **DioInitialize** can be used to initialize the DIO system configuration (Master and Slave boards) with one command. The required parameters (base address, density, banks etc.) are taken from the last configuration set by one of the following:

0. A previous call to DioSetupInitialization.
1. DioPanel configure push button.
2. GTDIO.INI file.

The function calls **DioSetupInitialization** for the Master and its Slaves.

The *pnMasterHandle* should be used in all subsequent calls to the driver library.

Example

The following example initializes a Master board and ARM it:

```
SHORT nStatus, nMasterHandle;
DioInitialize (1, &nMasterHandle, &nStatus);
DioArm(&nMasterHandle, &nStatus);
```

See Also

DioSetupInitialization, DioGetSlaveHandle

DioLevelShifterGetLoadResistance

Applies To

GC5050, GX5050, GX5150

Purpose

Returns the specified group load settings.

Syntax

DioLevelShifterGetLoadResistance (*nHandle, nGroup, pdwLoadVal, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
nGroup	SHORT	Group number can be as follows: 0 Group A, I/O pins 1 through 16. 1 Group B, I/O pins 17 through 32.
pdwLoadVal	PDWORD	Returned specified group load value. Value range from 50 Ohms to –1 (No-Load)
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

In order to maintain signal levels and automatically compensate for internal voltage drop the Level-Shifter is calibrated for a range of loads. The load value must be set before setting the voltage in order to take effect.

Example

The following example returns the current load settings of group 0:

```
SHORT   nStatus;
DWORD   dwLoad;
DioLevelShifterGetLoadResistance (nHandle, 0, &dwLoad, &nStatus);
```

See Also

DioLevelShifterSetLoadResistance, DioLevelShifterGetSummary, DioGetErrorString

DioLevelShifterGetOutputMode

Applies To

GC5050, GX5050, GX5150

Purpose

Returns the current output state.

Syntax

DioLevelShifterGetOutputMode *(nHandle, pnState, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
pnState	PSHORT	Returned output State can be as follows: 0 Disabled (Tri-state). 1 Enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

In disable mode, all channels have Tri-state high impedance (effectively disconnected).

Example

The following example returns the current output state of the Level Shifter:

```
SHORT   nStatus, nState;
DioLevelShifterGetOutputMode (nHandle, &nState, &nStatus);
```

See Also

DioLevelShifterSetOutputState, DioLevelShifterGetSummary, DioGetErrorString

DioLevelShifterGetSummary

Applies To

GC5050, GX5050, GX5150

Purpose

Returns the Level Shifter board summary from the onboard EEPROM.

Syntax

DioLevelShifterGetSummary (*nHandle, pszBoardSum, nSumMaxLen, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
pszBoardSum	LPSTR	Buffer to contain the returned board info string (null terminated string).
nSumMaxLen	SHORT	Size of the buffer to contain the error string.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The summary string provides the following data from the onboard EEPROM on the Level Shifter board in the order shown:

- Instrument Name (for example, GT5930)
- EEPROM format version (for example, 1.00)
- PCB revision (for example, 'A')
- Serial Number (for example, 59300100)
- Calibration time and date

For example, the returned string could look like the following:

"GT5930 A, Version 1.00, Revision A, S/N 59300100, last calibrated time Mon May 08 14:39:02 2000\n"

Example

The following example returns the Level Shifter board summary:

```
SHORT nHandle, nStatus;
CHAR szBoardSum[128]
DioLevelShifterGetSummary (nHandle, szBoardSum, 128, &nStatus)
```

See Also

DioLevelShifterSetVoltage, DioLevelShifterSetByLogicFamily, DioGetErrorString

DioLevelShifterGetVoltage

Applies To

GC5050, GX5050, GX5150

Purpose

Returns the specified group Low level, High level and Threshold voltage.

Syntax

DioLevelShifterGetVoltage *(nHandle, nGroup, pdLowVoltage, pdHighVoltage, pdTresholdVoltage, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
nGroup	SHORT	Group number can be as follows: 0. Group A, I/O pins 1 through 16. 1. Group B, I/O pins 17 through 32.
pdLowVoltage	LPDUOBLE	Returned low voltage setting.
pdHighVoltage	LPDUOBLE	Returned high voltage setting.
pdTresholdVoltage	LPDUOBLE	Returned threshold voltage setting.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

See **DioLevelShifterSetVoltage** for details on Low level, High level and Threshold voltage ranges.

Example

The following example returns the voltages settings (Low level voltage, High level voltage and Threshold voltage) for group A from a board specified by the board handle:

```
SHORT  nStatus;
DOUBLE dLowVoltage, dHighVoltage, dTresholdVoltage;
DioLevelShifterGetVoltage (nHandle, 0, &dLowVoltage, &dHighVoltage, &dTresholdVoltage, pnStatus);
```

See Also

DioLevelShifterSetVoltage, DioLevelShifterSetByLogicFamily, DioGetErrorString

DioLevelShifterSetByLogicFamily

Applies To

GC5050, GX5050, GX5150

Purpose

Sets the specified group Low, High and Threshold voltages according to a predefined logic family code.

Syntax

DioLevelShifterSetByLogicFamily *(nHandle, nGroup, nLogicFamily, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
nGroup	SHORT	Group number can be as follows: 0 Group A, channels 1 through 16. 1 Group B, channels 17 through 32.
nLogicFamily	SHORT	Logic family codes are as follow: 0 TTL. 1 ECL 10K.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Groups operate independently from one another; that is, setting of one group does not affect the other.

Threshold voltages apply only to input signals. They are used to determine whether the signal on any pin in the specified group is higher or lower than the threshold voltage.

See "I/O Modules and Interfaces User Guide" for details on the Logic families codes and parameters.

Example

The following example sets group A Low, High and Threshold voltages according to the TTL logic family code:

```
SHORT  nStatus;
DioLevelShifterSetByLogicFamily (nHandle, 1, 0, &nStatus);
```

See Also

DioLevelShifterSetVoltage, DioLevelShifterGetVoltage, DioGetErrorString

DioLevelShifterSetLoadResistance

Applies To

GC5050, GX5050, GX5150

Purpose

Sets the specified group load.

Syntax

DioLevelShifterSetLoadResistance (*nHandle, nGroup, dwLoadVal, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
nGroup	SHORT	Group number can be as follows: 0. Group A, I/O pins 1 through 16. 1. Group B, I/O pins 17 through 32.
dwLoadVal	DWORD	Sets the specified group load value. Value range from 50 Ohms to –1 (No-Load)
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

In order to maintain signal levels and automatically compensate for internal voltage drop the Level-Shifter is calibrated for a range of loads. The load value must be set before setting the voltage in order to take effect.

Example

The following example sets the load value to 125 Ohms of group 0:

```
SHORT   nStatus;
DioLevelShifterSetLoadResistance (nHandle, 0, 125, &nStatus);
```

See Also

DioLevelShifterGetLoadResistance, DioLevelShifterGetSummary, DioGetErrorString

DioLevelShifterSetOutputMode

Applies To

GC5050, GX5050, GX5150

Purpose

Enables or disables the output of group A and B.

Syntax

DioLevelShifterSetOutputMode *(nHandle, nMode, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.

nMode	SHORT	Output State can be as follow: 0 Disabled (Ttri-state). 1 Enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

In disable mode, all channels will be in Tri-state high impedance mode (disconnected).

Example

The following example enables the output State of the group 1:

```
SHORT   nStatus;
DioLevelShifterSetOutputMode (nHandle, 1, &nStatus);
```

See Also

DioLevelShifterGetOutputState, DioLevelShifterSetByLogicFamily, DioGetErrorString

DioLevelShifterSetVoltage

Applies To

GC5050, GX5050, GX5150

Purpose

Sets the specified group Low level, High level and Threshold voltages.

Syntax

DioLevelShifterSetVoltage *(nHandle, nGroup, dLowVoltage, dHighVoltage, dThresholdVoltage, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
nGroup	SHORT	Group number can be as follows: 0 Group A, I/O pins 1 through 16. 1 Group B, I/O pins 17 through 32.
dLowVoltage	DOUBLE	Low Voltage setting. See Comments for details.
dHighVoltage	DOUBLE	High voltage setting. See Comments for details.
dTresholdVoltage	DOUBLE	Threshold voltage setting. See Comments for details.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Voltage settings are limited by the voltage range setting as follows:

Level Shifter revision 'C':

Voltage range set to 0 (0 to 11):

Low Voltage: 0 to 8 volts and *dLowVoltage* must be < *dHighVoltage*.
High Voltage: 2.5 to 10.6 volts and *dHighVoltage* must be > *dLowVoltage.*
Threshold Voltage: -0.5 to 9.5 volts.
Voltage range set to 1 (-3 to 8):
Low Voltage: -3 to 5 volts and *dHighVoltage* must be > *dLowVoltage*.
High Voltage: 0 to 8 volts and *dHighVoltage* must be > *dLowVoltage*.
Threshold Voltage: - 3 to 6.5 volts

Level Shifter revision 'D' and above (single voltage range):

Low Voltage: -3 to 8 volts and *dLowVoltage* must be < *dHighVoltage*.
High Voltage: 3 to 8 volts and *dHighVoltage* must be > *dLowVoltage.*
Threshold Voltage: -3 to 8 volts.

Groups operate independently from each other, that is, the setting of one group does not affect the other. Threshold voltages apply only to input signals. They are used to determine whether the signal on any pin in the specified group is higher or lower than the threshold voltage. The level shifter support frequencies up to 50MHz for both input and output signals.

Note: If any of the voltages is out of the out of the current range setting, the function returns with error.

Example

The following example sets the **Level Shifter** voltages as follow:

```
Group = B
Low voltage level = 1.2
High voltage level = 6.8
Threshold voltage = 3.2
SHORT   nStatus;
DioLevelShifterSetVoltage (nHandle, 1, 1.2, 6.8, 3.2, &nStatus);
```

See Also

DioLevelShifterGetVoltage, DioGetErrorString

DioLoadFile

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Loads a DIO file or file segment to the specified Master and its Slaves.

Syntax

DioLoadFile *(nMasterHandle, szFileName, dwFileStart, dwStart, pdwSize, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
szFileName	LPSTR	DIO file name.
dwFileStart	DWORD	File starting step number.
dwStart	DWORD	Board starting step number.
pdwSize	PDWORD	Number of steps to load. When 0 is used, all steps to the end of the file are loaded. Upon return, the variable holds the number of steps actually loaded.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function loads steps starting at *dwFileStart* to board file(s) steps starting at *dwStart* until *dwSteps* are loaded.

The **DioLoadFile** supports the following file formats:

- DIO files created using ***DIOEasy*** 2x.
- DIO files created using ***DIOEasy*** 1.x.

The board must be in the halt state before a file can be loaded.

This function uses the default ***DIOEasy2.0*** file format.

After the function is called, the program counter is set to 0.

When loading a DIO file, the file's board setup is applied to the board (see Appendix B).

Example

The following example initializes the Master board and its Slaves and loads a DIO file:

```
DWORD dwSize=0;
DioInitialize (0, &nMasterHandle, &nStatus);
DioLoadFile (nMasterHandle, "MemoryTest.dio", 0, 0, &dwSize, &nStatus);
```

See Also

DioSaveFile, DioCompareFiles

DioMemoryFill

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Fills the specified board memory or file data with pre-defined patterns.

Syntax

DioMemoryFill (*nHandle, nPatern, nMemory, dwStart, dwSize, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO or File board handle.
nPatern	SHORT	Set the memory patterns as follow: 0 DIO_MEMFILL_ZERO: Fill with Zeros. 1 DIO_MEMFILL_RAMP: Fill with Ramp. 2 DIO_MEMFILL_CONTINUES_RAMP: Fill with continuous Ramp. 3 DIO_MEMFILL_RANDOM: Fill with Random. 4 DIO_MEMFILL_CHECKER_BOARD: Checker board. 5 DIO_MEMFILL_ONE: Fill with Ones.
nMemory	SHORT	Select the memory to fill: **GC5050, GX5050 and File:** 0 IN_ARRAY: In memory 1 OUT_ARRAY: Out memory 3 CTRL_ARRAY: Control memory **GX5280 and GX5280 File:** 2 DATA_ARRAY: I/O data memory **GX5290/GX5290e/GX5295 and GX5290/GX5290e/GX5295 File:** 0 IN_ARRAY: In memory 1 OUT_ARRAY: Out memory 4 DIRECTION_ARRAY: Direction memory **GX5055 and File:** 0 IN_ARRAY: In memory 1 OUT_ARRAY: Out memory 3 CTRL_ARRAY: Control memory 4 DIRECTION_ARRAY: Direction memory 5 VALID_DATA_ARRAY: Valid data array memory
dwStart	DWORD	Starting step number.
dwSize	PDWORD	Number of steps to fill.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function fills steps starting at *dwStart* to the specified board or file starting at *dwStart* up to *dwSize*. The patterns are filled as follows:

0 Fill with Zeros - reset those steps to zero.

1 Fill with Ramp - positive ramp starts from zero, the last step or the current width setting (GX5150) limits the last value. If the width setting limits the ramp, then the last value will be filled for the remaining steps. For example, if the width was set to 8-bits and number of steps to 512, then the steps from 255 up to 512 will have the value of 255 (0xFF).

2 Fill with continues Ramp - a positive ramp start from zero. If the ramp reaches its max value due to the width, then the next step will start another ramp. For example, if the width was set to 8-bits and number of steps to 512, then at step 256 a new ramp will start.

3 Fill with Random - random values with a random seed value.

4 Checker board - First step value is 0x55555555, next is 0xAAAAAAAA.

5 Fill with Ones - set those steps to ones.

After the function is called, the program counter is set to zero.

Example

The following example initializes a Master board and fills the first 16K steps with a ramp:

```
DWORD dwSize=0;
DioInitialize (0, &nMasterHandle, &nStatus);
DioMemoryFill (nMasterHandle, 1, 0, 0, 16384, &nStatus);
```

See Also

DioWriteInMemory, DioWriteOutMemory, DioWriteIOMemory, DioReadInMemory, DioReadOutMemory, DioReadIOMemory, DioGetErrorString

DioMeasure

Applies To

Gx5055, Gx5295

Purpose

Measure the specified signal.

Syntax

DioMeasure(*nHandle, nChannel, nSignal, pdResult, nMeasurementRate, dOp1, dOp2, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
nChannel	SHORT	GX5055: Specified channel in the DIO board, channel can be 0-31. GX5295: Specified channel in the DIO board to apply the settings: • I/O channels: 0-31. • Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
nSignal	SHORT	Signal to measure: Gx5055: 0 DIO_CH_TEMP: measures the specified channel junction temperature, units are specified by the dOp1 parameter. 1 DIO_CH_IO_VOLTAGE: measures the specified channel I/O voltage in Volts. 2 DIO_CH_OUTPUT_CURRENT: measures the specified channel output current in mA. 30 DIO_VCC: measures the board's VCC (3.3V) voltage. 31 DIO_HIGH_RAIL: measure the board's high voltage supply rail in Volts. 32 DIO_LOW_RAIL: measure the board's low voltage supply rail in Volts. Gx5295: 0 DIO_CH_TEMP: measures the specified channel junction temperature, units are specified by the dOp1 parameter. 1 DIO_CH_IO_VOLTAGE: measures the specified channel I/O voltage. 4. DIO_CH_PMU_CURRENT: measures the specified channel's PMU current in mA. 5. DIO_CH_PMU_VOLTAGE: measures the specified channel PMU voltage. 30 DIO_VCC: measures the board's VCC (3.3V) voltage.
pdResult	PDOUBLE	Returned measured signal, units are according to the signal type and optional parameters.

d*MeasurementRate*	DOUBLE	Measurement rate can be from 50000 measurements/sec (fastest least accurate measurement) down to 1 measurement/sec. The longer the measurement time the more accurate the measurement is. E.g. if the measurement rate was set to 100 then each measurement made by the DIO will be sampled for 10ms.
dOp1	DOUBLE	Optional parameter depends on the signal, by default this parameters is 0. If signal is Temperature, it will specify the Read back temperature units: 0 DIO_TEMP_CELSIUS: temperature units in Celsius. 1 DIO_TEMP_FAHRENHEIT: temperature units in Fahrenheit. If signal is other then Temperature, it will specify the number of digits to format the returned value. The number of digits can be 1 to 12.
dOp2	DOUBLE	Optional not used.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Example

The following example measures the temperature of channel 0 in Celsius and set the measurement rate to 100:

```
DOUBLE dResult;
DioMeasure (nHandle, 0, DIO_CH_TEMP, TRUE, DIO_TEMP_CELSIUS, 100, &nStatus);
```

See Also

DioWriteInMemory, DioWriteOutMemory, DioWriteIOMemory, DioReadInMemory, DioReadOutMemory, DioReadIOMemory, DioGetErrorString

DioPanel

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Opens the GT-DIO Panel window.

Syntax

DioPanel *(dwHwndParent, dMode, pdwHwndPanel)*

Parameters

Name	Type	Comments
dwHwndParent	DWORD	Parent window handle. Used when creating the panel window.
nMode	SHORT	The mode in which the panel main window is created. 0 Modeless window. 1 Modal window.
pdwHwndPanel	PDWORD	Returns panel window handle. Returns 0 on error.

Comments

The function is used to create the panel window. The panel window may be open as a modal or a modeless window depending on the *nMode* parameters.

If the mode is set to modal dialog (*nMode*=1), the panel will disable the parent window (*hwndParent*) and the function will return only after the window was closed by the user. In that case, the *pnHandle* may return the handle created by the user using the panel Initialize dialog. This handle may be used when calling other DIO functions.

If a modeless dialog was created (*nMode*=0), the function returns immediately after creating the panel window returning the window handle to the panel - *phwndPanel.* It is the responsibility of calling program to dispatch windows messages to this window so that the window can respond to messages.

Example

The following example calls the Panel:

```
HWND    hwnd, hwndPanel;
DWORD   dw;
DioPanel ((DWORD)hwnd, 0, &dw);
hwndPanel=(HWND)dw;
```

See Also

DioGetErrorString, DioPanelEx

DioPanelEx

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Opens the GTDIO Panel window. Passing a master handle will open the panel window already initialized displaying the associated master board settings skipping the Initialize step.

Syntax

DioPanel Ex*(pnHandle, dwHwndParent, dMode, pdwHwndPanel, pnStatus)*

Parameters

Name	Type	Comments
pnHandle	PSHORT	Master board handle. Passing 0 will open the panel in an initialized state.
dwHwndParent	DWORD	Parent window handle. Used when creating the panel window.
nMode	SHORT	The mode in which the panel main window is created, two modes: 0 GTDIO_PANEL_MODELESS: for modeless window 1 GTDIO_PANEL_MODAL: for modal window.
pdwHwndPanel	PDWORD	Returns panel window handle. Returns 0 on error.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function is used to create the panel window. The panel window may be open as a modal or a modeless window depending on the *nMode* parameters.

If the mode is set to modal dialog (*nMode*=1), the panel will disable the parent window (*dwHwndParent*) and the function will return only after the window was closed by the user. In that case, the *pnHandle* may return the handle created by the user using the panel Initialize dialog. This handle may be used when calling other DIO functions.

If a modeless dialog was created (*nMode*=0), the function returns immediately after creating the panel window returning the window handle to the panel - *pdwHwndPanel*. It is the responsibility of calling program to dispatch windows messages to this window so that the window can respond to messages.

Example

The following example initializes master number 1, and passing its handle to the pane to be opened initialized:

```
DWORD  hwndPanel;
SHORT nStatus, nMasterHandle;

DioInitialize (1, &nMasterHandle, &nStatus);
DioPanelEx(&nMasterHandle, NULL, GTDIO_PANEL_MODELESS, &hwndPanel, &nStatus);
```

See Also

DioGetErrorString, DioPanel

DioPause

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Change the specified Master board's state from RUN to PAUSE.

Syntax

DioPause *(nMasterHandle, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in the RUN state; otherwise this function returns an error. This function is useful for debugging.

Example

The following example set the board state to PAUSE:

```
SHORT nStatus;
DioPause (nMasterHandle, &nStatus);
```

See Also

DioHalt, DioArm, DioStep, DioReadProgramCounter, DioSetupTriggerXEventSource

DioPmuGetComparatorsSource

Applies To

GX5295

Purpose

Returns the specified channel's comparators' source.

Syntax

DioPmuGetComparatorsSource (*nHandle, nChannel, pnSourcel, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
nChannel	SHORT	Specified channel in the DIO board to apply the settings: • I/O channels: 0-31. • Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
pnSourcel	PSHORT	Returned the specified channel comparators' source, source can be: 0 DIO_PMU_COMPARATORS_SOURCE_IO_VOLTAGE: I/O voltage is the input signal to the channel low and high comparators. 1 DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT: PMU forced current is the input signal to the channel low and high comparators.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each channel's PMU (parametric measurement unit) supports two high-speed window comparators. The PMU on-chip window comparator supports a 2-bit "go / no-go" test with a low and high threshold settings. The thresholds are set using the **DioPmuSetupComparatorsValues** function call.

The PMU comparator window translates the forced voltage or the forced current being sensed into an output voltage that can be compared against an upper and lower limit.

The PMU window comparator low and high outputs can be read back directly providing direct access to the actual comparator status at any time.

Example

The following example returns channel 0 comparators' source:

```
SHORT   nSource;
DioPmuGetComparatorsValues (nHandle, 0, &nSource, &nStatus);
```

See Also

DioPmuGetComparisonResult, DioPmuSetupComparatorsValues, DioPmuSetupComparatorsSource, DioPmuSetupForcedCurrent, DioPmuSetupForcedVoltage, DioGetErrorString

DioPmuGetComparatorsValues

Applies To

GX5295

Purpose

Returns the specified channel's comparators' high and low voltage levels settings.

Syntax

DioPmuGetComparatorsValues (*nHandle, nChannel, pdHighVal, pdLowVal, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
nChannel	SHORT	Specified channel in the DIO board to apply the settings: • I/O channels: 0-31. • Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
pdHighVal	PDOUBLE	Returned Comparators high voltage level settings. • If the comparators' source was set to DIO_PMU_COMPARATORS_SOURCE_IO_VOLTAGE: I/O (0) the value is in voltages and can be between -2.V to and +7.0V and should to be less then *pdHighVal.* • If the comparators 'source was set to DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT: I/O (1) the value is in mA and should to be less then *pdHighVal.* The value should be inside the current range limits as was set by the PMU Forced Current range settings, see the **DioPmuSetupForcedCurrent** API for value limits.
pdLowVal	PDOUBLE	Returned Comparators low voltage level settings. • If the comparators' source was set to DIO_PMU_COMPARATORS_SOURCE_IO_VOLTAGE: I/O (0) the value is in voltages and can be between -2.V to and +7.0V and should to be less then *pdHighVal.* • If the comparators 'source was set to DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT: I/O (1) the value is in mA and should to be less then *pdHighVal.* The value should be inside the current range limits as was set by the PMU Forced Current range settings, see the **DioPmuSetupForcedCurrent** API for value limits.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each channel's PMU (parametric measurement unit) supports two high-speed window comparators. The PMU on-chip window comparator supports a 2-bit "go / no-go" test with a low and high threshold settings. The thresholds are set using the **DioPmuSetupComparatorsValues** function call.

The PMU comparator window translates the forced voltage or the forced current being sensed into an output voltage that can be compared against an upper and lower limit.

The PMU window comparator low and high outputs can be read back directly providing direct access to the actual comparator status at any time.

Example

The following example returns channel 0 comparators' high and low voltage levels settings:

```
DUOBLE  dHiVal, dLoLevel;
DioPmuGetComparatorsValues (nHandle, 0, &dHiVal, &dLoLevel, &nStatus);
```

See Also

DioPmuGetComparisonResult, DioPmuSetupComparatorsValues, DioPmuSetupComparatorsSource, DioPmuSetupForcedCurrent, DioPmuSetupForcedVoltage, DioGetErrorString

DioPmuGetComparisonResult

Applies To

GX5295

Purpose

Returns the PMU input comparators states and comparison validity bit.

Syntax

DioPmuGetComparisonResult (*nHandle, nChannel, pdwComparisonResults, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
pdwComparisonResults	PDWORD	Returned comparison result, the returned comparison result has the following structure: Bit 0: Low level comparison result: • Bit low: The actual input level was below the comparator low level as was set by *DioPmuSetupComparatorsValues* API. • Bit high: The actual input level was equal or above the comparator low level as was set by *DioPmuSetupComparatorsValues* API. Bit 1: High level comparison result: • Bit low: The actual input level was below the comparator high level as was set by *DioPmuSetupComparatorsValues* API. • Bit high: The actual input level was equal or above the comparator high level as was set by *DioPmuSetupComparatorsValues* API. Bit 2: Comparison validity, returns the validity of the low and high comparators result (see comments for details): • Bit low: Invalid comparison. • Bit high: valid comparison.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each channel's PMU (parametric measurement unit) supports two high-speed window comparators. The PMU on-chip window comparator supports 2-bit "go / no-go" test with a low and high threshold settings. The comparators can be used for a high speed go / no-go" test of the I/O voltage at any time or when the specified channel is set to be in Forced Voltage mode the user can either do a high speed go / no-go" test of the I/O voltage or current.

I/O Voltage comparison

The PMU comparator window compares the I/O voltage against an upper and lower voltage limits. The voltage thresholds are set using the **DioPmuSetupComparatorsValues** function call.

NOTE: The comparators source need to be set to set to voltage comparison by calling **DioPmuSetupComparatorsSource** API.

Forced Current comparison

The PMU comparator window compares the output forced current level against an upper and lower current limits. The current thresholds are set using the **DioPmuSetupComparatorsValues** function call, and the values are in mA.

In order to get higher comparison accuracy the user may set the comparators 'current range by calling **DioPmuSetupForcedCurrent** with the desired current range API.

NOTE: The comparators source need to be set to current comparison by calling **DioPmuSetupComparatorsSource** API.

Validity bit description

The following table lists the comparators' Validity bit description:

Condition	**Comparison Validity**
Level > High Comparator's Threshold	1
Level < Low Comparator's Threshold	
Level < High Comparator's Threshold **and** Level > Low Comparator's Threshold	Invalid

NOTE: The PMU window comparator low and high outputs can be read back directly providing direct access to the actual comparator status at any time.

NOTE: Current comparison is available on any Gx5295 board that was calibrated after June 11, 2012 and using GTDIO v4.1 (Build 60) or above.

Example

I/O Voltage comparison example:

This example demonstrates the comparison of channel 1 I/O voltage levels:

```
SHORT   nStatus;
DWORD   dwComparisonResult;


// Setup comparators source to voltage comparison
DioPmuSetupComparatorsSource(nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 1, NULL, 1,
DIO_PMU_COMPARATORS_SOURCE_IO_VOLTAGE, &nStatus);
// Set channel 1 high and low comparator voltage level
DioPmuSetupComparatorsValues(nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 1, NULL, 1, 2.5, -1,
&nStatus);
// Returns the Comparison result
DioPmuGetComparisonResult(nHandle, 1 &dwComparisonResult, &nStatus);
```

Forced Voltage current comparison example:

This example demonstrates the comparison of channel 1 current low and high levels in forced voltage mode:

```
SHORT   nStatus;
DWORD   dwComparisonResult;


DioSetupChannelMode(nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 1, NULL, 1,
DIO_CHANNEL_MODE_PMU_FORCE_VOLTAGE, &nStatus) ;
// Set channel 1 forced voltage level
DioPmuSetupForcedVoltage (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 1, NULL, 1, 1.5,
GX529X_PMU_CURRENT_RANGE_N32MA_TO_P32MA, DIO_PMU_FV_SETUP_DEFAULT, 0, &nStatus);
```

```
// Set channel 1 high and low comparator current level (in mA)

DioPmuSetupComparatorsValues(nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 1, NULL, 1, 2.5, -1,
&nStatus);

// Returns the Comparison result

DioPmuGetComparisonResult(nHandle, 1, &dwComparisonResult, &nStatus);
```

See Also

DioPmuSetupComparatorsValues, DioPmuSetupComparatorsSource, DioPmuSetupForcedCurrent, DioPmuSetupForcedVoltage, DioPmuSetupForcedCurrentCommutatingVoltage, DioGetErrorString

DioPmuGetForcedCurrent

Applies To

GX5295

Purpose

Returns the specified channel's programmed PMU forced current value and the current range.

Syntax

DioPmuGetForcedCurrent (*nHandle, nChannel, pdCurrent, pnCurrentRange, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
nChannel	SHORT	Specified channel in the DIO board to apply the settings: GX5055: I/O channels: 0-31. GX5295: • I/O channels: 0-31. • Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
pdCurrent	PDOUBLE	Returned programmed PMU forced current value. PMU forced current value depends on the current range settings.

pnCurrentRange	PSHORT	Returned programmed PMU forced current range. PMU forced current range can as follows: GX5055: 0 GX5055_PMU_CURRENT_RANGE_N200MA_TO_P200MA: forced current range is between -200mA to +200mA. 1 GX5055_PMU_CURRENT_RANGE_N25MA_TO_P25MA: forced current range is between -25mA to +25mA. GX5295: 0. GX529X_PMU_CURRENT_RANGE_N32MA_TO_P32MA: PMU forced current range is between -32mA to +32mA. 1. GX529X_PMU_CURRENT_RANGE_N8MA_TO_P8MA: PMU forced current range is between -8mA to +8mA. 2. GX529X_PMU_CURRENT_RANGE_N2MA_TO_P2MA: PMU forced current range is between -2mA to +2mA. 3. GX529X_PMU_CURRENT_RANGE_N512UA_TO_P512UA: PMU forced current range is between -512uA to +512uA. 4. GX529X_PMU_CURRENT_RANGE_N128UA_TO_P128UA: PMU forced current range is between -128uA to +128uA. 5. GX529X_PMU_CURRENT_RANGE_N32UA_TO_P32UA: PMU forced current range is between -32uA to +32uA. 6. GX529X_PMU_CURRENT_RANGE_N8UA_TO_P8UA6: PMU forced current range is between -8uA to +8uA. 7. GX529X_PMU_CURRENT_RANGE_N2UA_TO_P2UA7: PMU forced current range is between -2uA to +2uA.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in the PMU Forced current mode prior calling this function otherwise the function returns an error.

Example

The following example returns channel 0 programmed PMU forced current value and the current range:

```
SHORT   nCurrentRange, nStatus;
DOUBLE  dCurrent;
DioPmuGetForcedCurrent (nHandle, 0, &dCurrent, &nCurrentRange, &nStatus);
```

See Also

DioSetupChannelMode, DioPmuSetupForcedCurrent, DioPmuSetupForcedVoltage, DioPmuSetupForcedCurrentCommutatingVoltage, DioGetErrorString

DioPmuGetForcedCurrentCommutatingVoltage

Applies To

GX5295, File.

Purpose

Returns the specified channel PMU forced current commutating voltage.

Syntax

DioPmuGetForcedCurrentCommutatingVoltage (*nHandle, nChannel, pdVComHi, pdVComLo, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannel	SHORT	Specified channel in the DIO board to apply the settings: • I/O channels: 0-31. • Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
pdVComHi	PDOUBLE	Input channel high commutating voltages value, voltage can be set from --2V to +7V.
pdVComLo	PDOUBLE	Input channel low commutating voltages value, voltage can be set from --2V to +7V.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each PMU has a set of programmable commutating voltage that limit the voltage swing when the PMU is forcing current. These commutating voltage protect the DUT when current is being forced into a high impedance node at the DUT.
If the sensed voltage exceeds the high or low commutating voltage the PMU reduces the output current in order for the output voltage to not exceed the commutating voltage. If the voltage subsequently returns back to within the commutating voltage thresholds, the PMU resumes forcing the programmed current.
The Channels PMU forced current commutating voltage can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example returns channel 0 PMU forced current commutating voltage:

```
DOUBLE  dVComHi, dVComLo;
DioPmuGetForcedCurrentCommutatingVoltage (nHandle, 0, &dVComHi, &dVComLo, &nStatus);
```

See Also

DioSetupChannelMode, DioPmuSetupForcedCurrent, DioPmuSetupForcedVoltage, DioPmuSetupForcedCurrentCommutatingVoltage, DioGetErrorString

DioPmuGetForcedVoltage

Applies To

GX5295

Purpose

Returns the specified channel's programmed PMU forced voltage value and the output current range.

Syntax

DioPmuGetForcedVoltage (*nHandle, nChannel, pdVoltage, pnCurrentRange, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
nChannel	SHORT	Specified channel in the DIO board to apply the settings: • I/O channels: 0-31. • Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
pdVoltage	PDOUBLE	Returned programmed PMU forced voltage value. PMU forced voltage value can be between -2V to and 7V.
pnCurrentRange	PSHORT	Programmed PMU I/O current range, current range can be as follows: 0 GX529X_PMU_CURRENT_RANGE_N32MA_TO_P32MA: PMU current range is between -32mA to +32mA. 1 GX529X_PMU_CURRENT_RANGE_N8MA_TO_P8MA: PMU current range is between -8mA to +8mA. 2 GX529X_PMU_CURRENT_RANGE_N2MA_TO_P2MA: PMU Programmed PMU I/O current range, current range can be as follows current range is between -2mA to +2mA. 3 GX529X_PMU_CURRENT_RANGE_N512UA_TO_P512UA: PMU Programmed PMU I/O current range, current range can be as follows current range is between -512uA to +512uA. 4 GX529X_PMU_CURRENT_RANGE_N128UA_TO_P128UA: PMU Programmed PMU I/O current range, current range can be as follows current range is between -128uA to +128uA. 5 GX529X_PMU_CURRENT_RANGE_N32UA_TO_P32UA: PMU Programmed PMU I/O current range, current range can be as follows current range is between -32uA to +32uA. 6 GX529X_PMU_CURRENT_RANGE_N8UA_TO_P8UA6: PMU Programmed PMU I/O current range, current range can be as follows current range is between -8uA to +8uA. 7 GX529X_PMU_CURRENT_RANGE_N2UA_TO_P2UA7: PMU Programmed PMU I/O current range, current range can be as follows current range is between -2uA to +2uA.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in the PMU Forced voltage mode prior calling this function otherwise the function returns an error.

In order to achieve greater accuracy when measuring PMU current while in Forced Voltage mode, the user can specify a smaller current range that better fits the expected I/O load and forced voltage condition then the default range of -32mA to +32mA. It is recommended that the estimated current will be less or equal to 80% of the specified current range when no using the default range of -32mA to +32mA., otherwise the current measurement can be inaccurate as well as the actual current might exceed the specified range causing the output voltage to be clamped.

Example

The following example returns channel 0 programmed PMU forced voltage and current range:

```
SHORT   nCurrentRange, nStatus;
DOUBLE  dVoltage;
DioPmuGetForcedVoltage (nHandle, 0, &dVoltage, & nCurrentRange, &nStatus);
```

See Also

DioSetupChannelMode, DioPmuSetupForcedCurrent, DioPmuSetupForcedVoltage, DioGetErrorString

DioPmuSetupComparatorsSource

Applies To

GX5295

Purpose

Sets the specified channels comparators' source.

Syntax

DioPmuSetupComparatorsSource (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, nSource, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nSourcel	SHORT	Specified channel comparators' source, source can be: 0 DIO_PMU_COMPARATORS_SOURCE_IO_VOLTAGE: I/O voltage is the input signal to the channel low and high comparators. 1 DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT: PMU forced current is the input signal to the channel low and high comparators.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each channel's PMU (parametric measurement unit) supports two high-speed window comparators. The PMU on-chip window comparator supports a 2-bit "go / no-go" test with a low and high threshold settings. The thresholds are set using the **DioPmuSetupComparatorsValues** function call.

The PMU comparator window translates the forced voltage or the forced current being sensed into an output voltage that can be compared against an upper and lower limit.

The PMU window comparator low and high outputs can be read back directly providing direct access to the actual comparator status at any time.

Example

The following example uses an array of channels list to their comparators' source to forced current:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioPmuSetupComparatorsSource(nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0,
DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT, &nStatus);
```

The following example sets all the board's channels comparators' source to forced current:

```
DioPmuSetupComparatorsSource(nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL, 0,
DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT, &nStatus);
```

The following example sets all the boards channels comparators' source in the domain to forced current:

```
DioPmuSetupComparatorsSource(nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL, 0,
DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT, &nStatus);
```

The following example sets channels 5 to 10 comparators' source to forced current:

```
DioPmuSetupComparatorsSource(nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10,
DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT, &nStatus);
```

See Also

DioPmuGetComparisonResult, DioPmuSetupComparatorsValues, DioPmuGetComparatorsSource, DioPmuSetupForcedCurrent, DioPmuSetupForcedVoltage, DioGetErrorString

DioPmuSetupComparatorsValues

Applies To

GX5295

Purpose

Sets the specified channel's comparators' high and low settings.

Syntax

DioPmuSetupComparatorsValues (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, dHighVal, dLowVal, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
dHighVal	DOUBLE	Comparators high voltage level settings. • If the comparators' source was set to DIO_PMU_COMPARATORS_SOURCE_IO_VOLTAGE: I/O (0) the value is in voltages and can be between -2.V to and +7.0V and should to be less then *dHighVal.* • If the comparators 'source was set to DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT: I/O (1) the value is in mA and should to be less then *dHighVal*. The value should be inside the current range limits as was set by the PMU Forced Current range settings, see the **DioPmuSetupForcedCurrent** API for value limits.
dLowVal	DOUBLE	Comparators low voltage level settings. • If the comparators' source was set to DIO_PMU_COMPARATORS_SOURCE_IO_VOLTAGE: I/O (0) the value is in voltages and can be between -2.V to and +7.0V and should to be less then *dHighVal.* • If the comparators 'source was set to DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT: I/O (1) the value is in mA and should to be less then *dHighVal*. The value should be inside the current range limits as was set by the PMU Forced Current range settings, see the **DioPmuSetupForcedCurrent** API for value limits.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each channel's PMU (parametric measurement unit) supports two high-speed window comparators. The PMU on-chip window comparator supports a 2-bit "go / no-go" test with a low and high threshold settings. The thresholds are set using the **DioPmuSetupComparatorsValues** function call.

The PMU comparator window translates the forced voltage or the forced current being sensed into an output voltage that can be compared against an upper and lower limit.

The PMU window comparator low and high outputs can be read back directly providing direct access to the actual comparator status at any time.

Example

The following example uses an array of channels list to set their comparators' high level to 3.0V and low levels to -1.0V:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioPmuGetComparatorsValues (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0, 4, -1, &nStatus);
```

The following example sets all the board's channels comparators' high level to 3.0V and low levels to -1.0V:

```
DioPmuGetComparatorsValues (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0, 4, -1, &nStatus);
```

The following example sets all the boards channels comparators' high level to 3.0V and low levels to -1.0V:

```
DioPmuGetComparatorsValues (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0, 4, -1, &nStatus);
```

The following example sets channels 5 to 10 comparators' high level to 3.0V and low levels to -1.0V:

```
DioPmuGetComparatorsValues (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10, 4, -1, &nStatus);
```

See Also

DioPmuGetComparisonResult, DioPmuGetComparatorsValues, DioPmuSetupComparatorsSource, DioPmuSetupForcedCurrent, DioPmuSetupForcedVoltage, DioGetErrorString

DioPmuSetupForcedCurrent

Applies To

GX5295

Purpose

Sets the specified channels range with the programmed PMU forced current value and the PMU forced current range.

Syntax

DioPmuSetupForcedCurrent (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, dCurrent, nCurrentRange, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
dCurrent	DOUBLE	Programmed PMU forced current value. PMU forced current value depends on the current range settings.
nCurrentRange	SHORT	Programmed PMU forced current range. PMU forced current range can as follows: <u>GX5055:</u> 0 GX5055_PMU_CURRENT_RANGE_N200MA_TO_P200MA: forced current range is between -200mA to +200mA. 1 GX5055_PMU_CURRENT_RANGE_N25MA_TO_P25MA: forced current range is between -25mA to +25mA. <u>GX5295:</u> 0. GX529X_PMU_CURRENT_RANGE_N32MA_TO_P32MA: PMU forced current range is between -32mA to +32mA. 1. GX529X_PMU_CURRENT_RANGE_N8MA_TO_P8MA: PMU forced current range is between -8mA to +8mA. 2. GX529X_PMU_CURRENT_RANGE_N2MA_TO_P2MA: PMU forced current range is between -2mA to +2mA. 3. GX529X_PMU_CURRENT_RANGE_N512UA_TO_P512UA: PMU forced current range is between -512uA to +512uA. 4. GX529X_PMU_CURRENT_RANGE_N128UA_TO_P128UA: PMU forced current range is between -128uA to +128uA. 5. GX529X_PMU_CURRENT_RANGE_N32UA_TO_P32UA: PMU forced current range is between -32uA to +32uA. 6. GX529X_PMU_CURRENT_RANGE_N8UA_TO_P8UA6: PMU forced current range is between -8uA to +8uA. 7. GX529X_PMU_CURRENT_RANGE_N2UA_TO_P2UA7: PMU forced current range is between -2uA to +2uA.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in the PMU Forced current prior calling this function otherwise the function returns an error.

Example

The following Gx5295 example uses an array of channels list to set programmed PMU forced current to 25mA and the current range to -32mA to +32mA:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioPmuSetupForcedCurrent (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0, 25.0,
GX529X_PMU_CURRENT_RANGE_N32MA_TO_P32MA, &nStatus);
```

The following example sets all the board's channels to set programmed PMU forced current to 25mA and the current range to -32mA to +32mA::

```
DioPmuSetupForcedCurrent (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL, 0, 25.0,
GX529X_PMU_CURRENT_RANGE_N32MA_TO_P32MA, &nStatus);
```

The following example sets all the boards' channels to set programmed PMU forced current to 25mA and the current range to -32mA to +32mA:

```
DioPmuSetupForcedCurrent (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL, 0, 25.0,
GX529X_PMU_CURRENT_RANGE_N32MA_TO_P32MA, &nStatus);
```

The following example sets channels 5 to 10 to set programmed PMU forced current to 25mA and the current range to -32mA to +32mA:

```
DioPmuSetupForcedCurrent (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10, 25.0,
GX529X_PMU_CURRENT_RANGE_N32MA_TO_P32MA, &nStatus);
```

See Also

DioSetupChannelMode, DioPmuGetForcedCurrent, DioPmuSetupForcedVoltage, DioGetErrorString

DioPmuSetupForcedCurrentCommutatingVoltage

Applies To

GX5295, File.

Purpose

Sets the specified channel PMU forced current commutating voltage.

Syntax

DioPmuSetupForcedCurrentCommutatingVoltage (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, pdVComHi, pdVComLo, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
pdVComHi	PDOUBLE	Input channel high commutating voltages value, voltage can be set from --2V to +7V.
pdVComLo	PDOUBLE	Input channel low commutating voltages value, voltage can be set from --2V to +7V.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each PMU has a set of programmable commutating voltage that limit the voltage swing when the PMU is forcing current. These commutating voltage protect the DUT when current is being forced into a high impedance node at the DUT.
If the sensed voltage exceeds the high or low commutating voltage the PMU reduces the output current in order for the output voltage to not exceed the commutating voltage. If the voltage subsequently returns back to within the commutating voltage thresholds, the PMU resumes forcing the programmed current.
The Channels PMU forced current commutating voltage can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example uses an array of channels list set PMU forced current commutating voltage

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioPmuSetupForcedCurrentCommutatingVoltage (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8,
anChannelList, 0, 3.5, -1, &nStatus);
```

The following example sets all the board's channels PMU forced current commutating voltage:

```
DioPmuSetupForcedCurrentCommutatingVoltage (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL
, 0, 3.5, -1, &nStatus);
```

The following example sets all the boards channels PMU forced current commutating voltage:

```
DioPmuSetupForcedCurrentCommutatingVoltage (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0,
NULL , 0, 3.5, -1, &nStatus);
```

The following example sets channels 5 to 10 PMU forced current commutating voltage:

```
DioPmuSetupForcedCurrentCommutatingVoltage (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL,
10, 3.5, -1, &nStatus);
```

See Also

DioSetupChannelMode, DioPmuSetupForcedCurrent, DioPmuSetupForcedVoltage, DioPmuGetForcedCurrentCommutatingVoltage, DioGetErrorString

DioPmuSetupForcedVoltage

Applies To

GX5295

Purpose

Sets the specified channels range programmed PMU forced voltage value.

Syntax

DioPmuSetupForcedVoltage (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, dVoltage, nCurrentRange, dwSetupMode, dMaxCurrent, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
dVoltage	DOUBLE	Programmed PMU forced voltage value. PMU forced voltage value can be between -2V to +7V.
nCurrentRange	SHORT	Programmed PMU I/O current range, current range can be as follows: 0 GX529X_PMU_CURRENT_RANGE_N32MA_TO_P32MA: PMU current range is between -32mA to +32mA. 1 GX529X_PMU_CURRENT_RANGE_N8MA_TO_P8MA: PMU current range is between -8mA to +8mA. 2 GX529X_PMU_CURRENT_RANGE_N2MA_TO_P2MA: PMU current range is between -2mA to +2mA. 3 GX529X_PMU_CURRENT_RANGE_N512UA_TO_P512UA: PMU current range is between -512uA to +512uA. 4 GX529X_PMU_CURRENT_RANGE_N128UA_TO_P128UA: PMU current range is between -128uA to +128uA. 5 GX529X_PMU_CURRENT_RANGE_N32UA_TO_P32UA: PMU current range is between -32uA to +32uA. 6 GX529X_PMU_CURRENT_RANGE_N8UA_TO_P8UA6: PMU current range is between -8uA to +8uA. 7 GX529X_PMU_CURRENT_RANGE_N2UA_TO_P2UA7: PMU current range is between -2uA to +2uA.
dwSetupMode	DWORD	PMU Forced Voltage Setup Mode can be as follows: 0 DIO_PMU_FV_SETUP_DEFAULT: the specified Forced Voltage level will be applied immediately. 1 DIO_PMU_FV_CHECK_FOR_OVER_CURRENT: before the specified Forced Voltage level is applied the driver checks the output for short condition and over current condition. If those conditions are false then the specified Forced Voltage level will be applied. Otherwise the function will return an error and the forced voltage level will be set to 0V.
dMaxCurrent	DOUBLE	Verify that the specified Forced Voltage level will not exceed the specified maximum current when the setup mode (*dwSetupMode)* parameter was set to check for over current. Otherwise this parameter is ignored.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in the PMU Forced voltage mode prior calling this function otherwise the function returns an error.

In order to achieve greater accuracy when measuring PMU current while in Forced Voltage mode, the user can specify a smaller current range that better fits the expected I/O load and forced voltage condition then the default range of -32mA to +32mA. It is recommended that the estimated current will be less or equal to 80% of the specified current range when no using the default range of -32mA to +32mA., otherwise the current measurement can be inaccurate as well as the actual current might exceed the specified range causing the output voltage to be clamped.

Note: settings those current ranges only helps in measuring the output current more accurately but does not prevent against over current as the current can exceeds the specified current range in case of a short or high forced voltage values combined with low load value.

Example

The following example uses an array of channels list set programmed PMU forced voltage to 3V and current range is between -8mA to +8mA:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioPmuSetupForcedVoltage (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0, 3.0,
GX529X_PMU_CURRENT_RANGE_N8MA_TO_P8MA, DIO_PMU_FV_SETUP_DEFAULT, 0, &nStatus);
```

The following example sets all the board's channels programmed PMU forced voltage to 3V and current range is between -8mA to +8mA:

```
DioPmuSetupForcedVoltage (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0, 3.0,
GX529X_PMU_CURRENT_RANGE_N8MA_TO_P8MA, DIO_PMU_FV_SETUP_DEFAULT, 0, &nStatus);
```

The following example sets all the boards channels programmed PMU forced voltage to 3V and current range is between -8mA to +8mA:

```
DioPmuSetupForcedVoltage (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0, 3.0,
GX529X_PMU_CURRENT_RANGE_N8MA_TO_P8MA, DIO_PMU_FV_SETUP_DEFAULT, 0, &nStatus);
```

The following example sets channels 5 to 10 programmed PMU forced voltage to 3V and current range is between -8mA to +8mA:

```
DioPmuSetupForcedVoltage (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10, 3.0,
GX529X_PMU_CURRENT_RANGE_N8MA_TO_P8MA, DIO_PMU_FV_SETUP_DEFAULT, 0, &nStatus);
```

See Also

DioPmuSetupMode, DioPmuSetupForcedCurrent, DioPmuGetForcedVoltage, DioGetErrorString

DioPowerSupplyGetRailsState

Applies To

GX5055

Purpose

Returns the high powered chassis power supply state.

Syntax

DioPowerSupplyGetRailsState (*nHandle, pnState, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Session identifier: Board handle is used when communicating with the hardware. The Board handle session identifier is returned by calling DioInitialize or DioSetupInitialization.
pnState	PSHORT	Used to get the high powered chassis power supply state 0. DIO_POWER_SUPPLY_DISABLE: high powered chassis power supply is disabled. 1. DIO_POWER_SUPPLY_ENABLE: high powered chassis power supply is enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative value on failure.

Comments

This function is used to prime the sequencer prior to an external start.

Example

```
See the DioPowerSupplySetRailsVoltage for a complete example.
```

See Also

DioPowerSupplyGetStatus, DioPowerSupplyIsSupported, DioPowerSupplyMeasure, DioPowerSupplyResetFault, DioPowerSupplySetRailsState, DioPowerSupplySetRailsVoltage, DioGetErrorString

DioPowerSupplyGetRailsVoltage

Applies To

GX5055

Purpose

Returned the programed high powered chassis power supply VCC and VEE voltage levels.

Syntax

DioPowerSupplyGetRailsVoltage (*nHandle, pdVcc, pdVee, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Session identifier: Board handle is used when communicating with the hardware. The Board handle session identifier is returned by calling DioInitialize or DioSetupInitialization.
dVcc	PDOUBLE	Returns the positive voltage rail level, The *dVcc* may be between *10V* to +28V. The maximum voltage swing between VCC and VEE may not exceed 33.5 V.
dVee	PDOUBLE	Returns the positive voltage rail level. The V*ee voltage* must may not exceed 18V and must not be less than 3V. The maximum voltage swing between VCC and VEE may not exceed 33.5 V.
pnStatus	PSHORT	Returned status: 0 on success, negative value on failure.

Comments

The Marvin Test Solutions Gx7015A and Gx7005A high powered chassis include a built in, user programmable power supply system. They use a special backplane to deliver high power to the GX5055 pin electronics via the J5 PXI connector. The power supply is made up of two VCC high rail modules, one VEE low rail module, and one VDD module.

The GX5055 DIO is capable of controlling these modules by setting their voltages, and monitoring their voltage and current output.

Example

```
See the DioPowerSupplySetRailsVoltage for a complete example.
```

See Also

DioPowerSupplyGetStatus, DioPowerSupplyIsSupported, DioPowerSupplyMeasure, DioPowerSupplyResetFault, DioPowerSupplySetRailsState, DioPowerSupplySetRailsVoltage, DioGetErrorString

DioPowerSupplyGetStatus

Applies To

GX5055

Purpose

Returns the high powered chassis power supply status

Syntax

DioPowerSupplyGetStatus (*nHandle, pdwStatus, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Session identifier: Board handle is used when communicating with the hardware. The Board handle session identifier is returned by calling DioInitialize or DioSetupInitialization.
pdwStatus	PDWORD	Used to get the high powered chassis power supply status register. Refer to the **Comments** section for the bit description of this register.
pnStatus	PSHORT	Returned status: 0 on success, negative value on failure.

Comments

The Marvin Test Solutions Gx7015A and Gx7005A high powered chassis include a built in, user programmable power supply system. They use a special backplane to deliver high power to the GX5055 pin electronics via the J5 PXI connector. The power supply is made up of two VCC high rail modules, one VEE low rail module, and one VDD module.

The GX5055 DIO is capable of controlling these modules by setting their voltages, and monitoring their voltage and current output.

The bit description of the pdwStatus parameter is as follows:

Bit #	Description
20	Latched DC Monitor, a high indicate a fault has occurred. Fault could be UVP, OVP, output or internal DC out of regulation and will inhibit the power supply.
19	Latched AC Monitor, a high indicate a fault has occurred. Fault could be UVP, OVP, AC loss, or AC missing cycle and will inhibit the power supply.
18	Latched VCC Fault, a high indicate VCC has exceeded range. Range is 9.6 V to 29 VDC and will inhibit the power supply.
17	Latched VEE Fault, a high indicate VEE has exceeded range. Range is –2.8 V to –20 VDC and will inhibit the power supply.
16	Real time Delta Fault, a high indicate power supply exceeded max voltage swing for VCC and VEE. The max voltage swing is 34 VDC and will inhibit the power supply.
15	A high indicate that the Power supply is inhibited.
14	A high indicate that Latched DC Monitor will inhibit the power supply.
13	A high indicate that Latched AC Monitor will inhibit the power supply.
12	A high indicate that the Power supply fault.
11	A high if detected master Power interface board.
10	A high indicates that both Vcc and VEE are enabled
9	A high indicates that the power supply is enabled
8	A high indicates that VDD is enabled
7	Real time DC Monitor, a high indicate a fault has occurred. Fault could be UVP, OVP, output or internal DC out of regulation.
6	Real time AC Monitor, a high indicate a fault has occurred. Fault could be UVP, OVP, AC loss, or AC missing cycle.
5	A high indicates that real time faults (bits 1-4) below are disabled from inhibiting the power supply.
4	Real time Pin Electronics Fault, a high fault has occurred at the Pin Electronics. Faults could be over temp or over current.
3	Real time VCC Fault, a high indicate VCC has exceeded range. Range is 9.6 V to 29 VDC.
2	Real time VEE Fault, a high indicate VEE has exceeded range. Range is –2.8 V to –20 VDC.
1	Real time Delta Fault, a high indicate power supply exceeded max voltage swing for VCC and VEE. The max voltage swing is 34 VDC.
0	A high indicates the presence of a Power Interface Board on the backplane.

Example

```
See the DioPowerSupplySetRailsVoltage for a complete example.
```

See Also

DioPowerSupplyIsSupported, DioPowerSupplyMeasure, DioPowerSupplyResetFault, DioPowerSupplySetRailsState, DioPowerSupplySetRailsVoltage, DioGetErrorString

DioPowerSupplyIsSupported

Applies To

GX5055

Purpose

Return if the currnet chassis is a high power chassis.

Syntax

DioPowerSupplyIsSupported (*nHandle*, *pbSupported*, *pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Session identifier: Board handle is used when communicating with the hardware. The Board handle session identifier is returned by calling DioInitialize or DioSetupInitialization.
pbSupported	PBOOL	Return if the currnet chassis is a high power chassis. 0. The currnet chassis is NOT a high power chassis 1. The currnet chassis is a high power chassis.
pnStatus	PSHORT	Returned status: 0 on success, negative value on failure.

Comments

The Marvin Test Solutions Gx7015A and Gx7005A high powered chassis include a built in, user programmable power supply system. They use a special backplane to deliver high power to the GX5055 pin electronics via the J5 PXI connector. The power supply is made up of two VCC high rail modules, one VEE low rail module, and one VDD module.

The GX5055 DIO is capable of controlling these modules by setting their voltages, and monitoring their voltage and current output.

Example

```
See the DioPowerSupplySetRailsVoltage for a complete example.
```

See Also

DioPowerSupplyGetStatus, DioPowerSupplyMeasure, DioPowerSupplyResetFault, DioPowerSupplySetRailsState, DioPowerSupplySetRailsVoltage, DioGetErrorString

DioPowerSupplyMeasure

Applies To

GX5055

Purpose

Take measurement of the specified high power supply source.

Syntax

DioPowerSupplyMeasure (*nHandle, nSource, pdMeasurement, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Session identifier: Board handle is used when communicating with the hardware. The Board handle session identifier is returned by calling DioInitialize or DioSetupInitialization.
nSource	SHORT	Measurement source: 0. DIO_POWER_SUPPLY_VCC1_VOLTAGE: High Power supply First VCC module. First VCC module is controlled by the Gx5055 board installed in slot 5. 1. DIO_POWER_SUPPLY_VCC2_VOLTAGE: High Power supply Second VCC module. Second VCC module is controlled by the Gx5055 board installed in slot 13. 2. DIO_POWER_SUPPLY_VEE_VOLTAGE: High Power supply VEE module. VEE module is controlled by the Gx5055 board installed in slot 5. 3. DIO_POWER_SUPPLY_VDD_VOLTAGE: High Power supply VDD module. VDD module is controlled by the Gx5055 board installed in slot 5. 10. DIO_POWER_SUPPLY_VCC1_CURRENT: High Power supply First VCC currnet module. First VCC currnet module is controlled by the Gx5055 board installed in slot 5. 11. DIO_POWER_SUPPLY_VCC2_CURRENT: High Power supply Second VCC currnet module. Second VCC currnet module is controlled by the Gx5055 board installed in slot 13. 12. DIO_POWER_SUPPLY_VEE_CURRENT: High Power supply VEE current module. VEE current module is controlled by the Gx5055 board installed in slot 5. 13. DIO_POWER_SUPPLY_VDD_CURRENT: High Power supply VDD current module. VDD current module is controlled by the Gx5055 board installed in slot 5.
pdMeasurement	PDOUBLE	Returned measruemnt, all measurements are rounded to 10mV and 10mA.
pnStatus	PSHORT	Returned status: 0 on success, negative value on failure.

Comments

The function takes measurement of the specified high power supply source:

The Marvin Test Solutions Gx7015A and Gx7005A high powered chassis include a built in, user programmable power supply system. They use a special backplane to deliver high power to the GX5055 pin electronics via the J5 PXI connector. The power supply is made up of two VCC high rail modules, one VEE low rail module, and one VDD module.

The GX5055 DIO is capable of controlling these modules by setting their voltages, and monitoring their voltage and current output.

Example

```
See the DioPowerSupplySetRailsVoltage for a complete example.
```

See Also

DioPowerSupplyGetStatus, DioPowerSupplyIsSupported, DioPowerSupplyResetFault, DioPowerSupplySetRailsState, DioPowerSupplySetRailsVoltage, DioGetErrorString

DioPowerSupplyResetFault

Applies To

GX5055

Purpose

Reset the high powered chassis power supply fault.

Syntax

DioPowerSupplyResetFault (*nHandle, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Session identifier: Board handle is used when communicating with the hardware. The Board handle session identifier is returned by calling DioInitialize or DioSetupInitialization.
pnStatus	PSHORT	Returned status: 0 on success, negative value on failure.

Comments

The function resets the power supply fault condition, disconnects all Gx5960, and set the power supply rail voltages as follows:

VCC=+12V

VEE=-12V

The Marvin Test Solutions Gx7015A and Gx7005A high powered chassis include a built in, user programmable power supply system. They use a special backplane to deliver high power to the GX5055 pin electronics via the J5 PXI connector. The power supply is made up of two VCC high rail modules, one VEE low rail module, and one VDD module.

The GX5055 DIO is capable of controlling these modules by setting their voltages, and monitoring their voltage and current output.

Example

```
See the DioPowerSupplySetRailsVoltage for a complete example.
```

See Also

DioPowerSupplyGetStatus, DioPowerSupplyIsSupported, DioPowerSupplyMeasure, DioPowerSupplySetRailsState, DioPowerSupplySetRailsVoltage, DioGetErrorString

DioPowerSupplySetRailsState

Applies To

GX5055

Purpose

Programs the high powered chassis power supply state.

Syntax

DioPowerSupplySetRailsState (*nHandle, nState, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Session identifier: Board handle is used when communicating with the hardware. The Board handle session identifier is returned by calling DioInitialize or DioSetupInitialization.
nState	SHORT	Used to set the high powered chassis power supply state. When enabled the VCC, VEE, and VDD modules are powered on. 0. DIO_DISABLED: VCC, VEE, and VDD modules are shut down 1. DIO_ENABLED: VCC, VEE, and VDD modules are turned on
pnStatus	PSHORT	Returned status: 0 on success, negative value on failure.

Comments

The Marvin Test Solutions Gx7015A and Gx7005A high powered chassis include a built in, user programmable power supply system. They use a special backplane to deliver high power to the GX5055 pin electronics via the J5 PXI connector. The power supply is made up of two VCC high rail modules, one VEE low rail module, and one VDD module.

The GX5055 DIO is capable of controlling these modules by setting their voltages, and monitoring their voltage and current output.

Example

See DioPowerSupplySetRailsVoltage API for a complete example.

See Also

DioPowerSupplyGetStatus, DioPowerSupplyIsSupported, DioPowerSupplyMeasure, DioPowerSupplyResetFault, DioPowerSupplySetRailsVoltage, DioGetErrorString

DioPowerSupplySetRailsVoltage

Applies To

GX5055

Purpose

Programs the high powered chassis power supply VCC and VEE voltage levels.

Syntax

DioPowerSupplySetRailsVoltage (*nHandle, dVcc, dVee, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Session identifier: Board handle is used when communicating with the hardware. The Board handle session identifier is returned by calling DioInitialize or DioSetupInitialization.
dVcc	DOUBLE	Used to specify the positive voltage rail level to be program, The *dVcc* may be between 10V to +28V. The maximum voltage swing between VCC and VEE may not exceed 33.5 V.
dVee	DOUBLE	Used to specify the negative voltage rail level to be program. The V*ee voltage* must may not exceed 18V and must not be less than 3V. The maximum voltage swing between VCC and VEE may not exceed 33.5 V.
pnStatus	PSHORT	Returned status: 0 on success, negative value on failure.

Comments

The Marvin Test Solutions Gx7015A and Gx7005A high powered chassis include a built in, user programmable power supply system. They use a special backplane to deliver high power to the GX5055 pin electronics via the J5 PXI connector. The power supply is made up of two VCC high rail modules, one VEE low rail module, and one VDD module.

The GX5055 DIO is capable of controlling these modules by setting their voltages, and monitoring their voltage and current output.

Example

The following example:

1. Check if high power supply is supported.
2. Sets the power connection to the backplane and returns the power connection settings
3. Enables the high powered chassis power supply.
4. Read the power supply's status, if fault is detected, then call DioPowerSupplyResetFault.
5. Program VCC voltage rail to 18.0V and VEE to 14.0.
6. Read back each of the 4 voltages
7. Read back each of the 4 power supply current

```
SHORT nStatus;
SHORT nPowerState;
DWORD dwStatus;
DOUBLE dVoltage, dCurrent;
DOUBLE dVcc, dVee;
BOOL bSupported;

DioPowerSupplyIsSupported(nHandle, &bSupported, &nStatus);

If (bSupported)
{
    DioSetPowerConnect (nHandle, DIO_POWER_FROM_BACKPLANE, &nStatus);
    DioGetPowerConnect (nHandle, &nPowerState, &nStatus);

    DioPowerSupplySetState(nHandle, DIO_ENABLED, &nStatus);
    DioPowerSupplySetRailsVoltage (nHandle, 18.0V, 14.0 pnStatus)

    DioPowerSupplyGetStatus(nHandle, &dwStatus, &nStatus);
    if (dwStatus & 0x1F9000) // detected any of the faults
        DioPowerSupplyResetFault(nHandle, &nStatus);


    Read back the voltages:
    DioPowerSupplySetRailsVoltage (nHandle, &dVcc, &dVee, pnStatus)

    Measure power supplies 4 voltages:
    DioPowerSupplyMeasure(nHandle, DIO_POWER_SUPPLY_VCC1_VOLTAGE, &dVoltage, &nStatus);
    DioPowerSupplyMeasure(nHandle, DIO_POWER_SUPPLY_VCC1_VOLTAGE, &dVoltage, &nStatus);
    DioPowerSupplyMeasure(nHandle, DIO_POWER_SUPPLY_VEE_VOLTAGE, &dVoltage, &nStatus);
    DioPowerSupplyMeasure(nHandle, DIO_POWER_SUPPLY_VDD_VOLTAGE, &dVoltage, &nStatus);

    Measure power supplies 4 voltage railes currents:
    DioPowerSupplyMeasure(nHandle, DIO_POWER_SUPPLY_VCC1_CURRENT, &dCurrent, &nStatus);
    DioPowerSupplyMeasure(nHandle, DIO_POWER_SUPPLY_VCC1_CURRENT, &dCurrent, &nStatus);
    DioPowerSupplyMeasure(nHandle, DIO_POWER_SUPPLY_VEE_CURRENT, &dCurrent, &nStatus);
    DioPowerSupplyMeasure(nHandle, DIO_POWER_SUPPLY_VDD_CURRENT, &dCurrent, &nStatus);
}
```

See Also

DioPowerSupplyGetStatus, DioPowerSupplyIsSupported, DioPowerSupplyMeasure, DioPowerSupplyResetFault, DioPowerSupplySetRailsState DioGetErrorString

DioReadCtrlCommand

Applies To

GX5280, GX5290, GX5290e, GX5295, File

Purpose

Read the control command parameters from the specified step.

Syntax

DioReadCtrlCommand *(nHandle, dwStep, pnOpCode, pdwParam1, pdwParam2, pdwParam3, pdwParam4, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	DIO or File board handle.
dwStep	DWORD	Vector Step number
pnOpCode	PSHORT	Operation code
pdwParam1	PDWORD	Parameter 1 (see comments for details)
pwParam2	PDWORD	Parameter 2 (see comments for details)
pdwParam3	PDWORD	Parameter 3 (see comments for details)
pwParam4	PDWORD	Parameter 4 (see comments for details)
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in a HALT state before calling this function.

GX5281/GX5282:

The Operation code (pnOpCode) can be one of the following:

Op Code #	Op Code constant	Description
0	DIO_COMMAND_NONE	No Operation code
7	DIO_COMMAND_PAUSE	Pause
8	DIO_COMMAND_HALT	Halt
1000	DIO_COMMAND_CLEAR_ALL	Clear all the control commands.

GX5283/GX5291/GX5292/GX5293/GX5291e/GX5292e/GX5293e/GX5295:

The Operation code (*pnOpCode*) can be one of the following:

Op Code #	Op Code constant	Description
0	DIO_COMMAND_NONE	No Operation code
3	DIO_COMMAND_LOOP	Loop specified number of times between two predefined steps. *pdwParam1*: Last step *pdwParam2*: Number of loops

Op Code #	Op Code constant	Description
7	DIO_COMMAND_PAUSE	Pause
8	DIO_COMMAND_HALT	Halt
11	DIO_COMMAND_CONTINUOUS_LOOP	Loops Continuously between two predefined steps. *pdwParam1*: Last step
1000	DIO_COMMAND_CLEAR_ALL	Clear all the control commands.

When using DIO_COMMAND_LOOP or DIO_COMMAND_CONTINUOUS_LOOP commands the first and last steps are set on 32-bit boundaries, i.e. channels I/O vector data is always on boundaries of 32 bits. As a result when setting the I/O Channels configuration (width) sing **DioSetupIOConfiguration** function the loop command start and last steps need to be set according to the following guidelines:

*I/O channels (width)	First Step	Last Step
32	Any Even step number	Any Odd step number higher than First Ste
16	Any step number that when divided by 2 the integer part of the result is an even number. E.g. 5 is a valid first step number since when dividing by 2 the integer part is even number of 2, but 3 is invalid value since when dividing by 2 the integer part is 1 (not an even number).	Any step number that when divided by 2 the integer part of the result is an odd number. E.g. 1022 is a valid last step number since when dividing by 2 the integer part is an even number of 511, however 1024 is invalid value since when dividing by 2 the integer part is 512 (not an odd number).
8	Any step number that when divided by 4 the integer part of the result is an even number. E.g. 10 is a valid first step number since when dividing by 4 the integer part is even number of 2, but 6 is invalid value since when dividing by 4 the integer part is 1 (not an even number).	Any step number that when divided by 4 the integer part of the result is an odd number. E.g. 2044 is a valid last step number since when dividing by 4 the integer part is an even number of 511, however 2048 is invalid value since when dividing by 4 the integer part is 512 (not an odd number).
4	Any step number that when divided by 8 the integer part of the result is an even number.	Any step number that when divided by 8 the integer part of the result is an odd number.
2	Any step number that when divided by 16 the integer part of the result is an even number.	Any step number that when divided by 16 the integer part of the result is an odd number.
1	Any step number that when divided by 32 the integer part of the result is an even number.	Any step number that when divided by 32 the integer part of the result is an odd number.

Example

The following example reads the control command parameters from step number 100:

```
SHORT nOpCode;
DWORD dwParam1, dwParam2, dwParam3, dwParam4;
DioReadCtrlCommand (nHandle, 100, &nOpCode, &dwParam1, &dwParam2, &dwParam3, &dwParam4,
                    &nStatus);
```

See Also

DioWriteCtrlCommand, DioGetNextCtrlCommandStep, DioGetErrorString

DioReadCtrlMemory

Applies To

GC5050, GX5050, GX5055, GX5150, File

Purpose

Reads a block of control memory.

Syntax

DioReadCtrlMemory *(nHandle, pvMemory, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	Comments
NHandle	SHORT	DIO or File board handle.
pvMemory	PVOID	Pointer to an array of type void. The array type depends on the board type as follows: **GX5055:** Array data type is Double Word and maximum array size is 512K of double words. **GC5050, GX5050:** Array data type is Double Word and maximum array size is 1M of double words. **GX5150:** Array type must be BYTE.
dwStart	DWORD	Starting address to read.
DwSize	DWORD	Number of steps to read.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GC5050/GX5050/GX5055: Each step is controlled by one control (DWORD).

GX5150: Each control byte controls eight data bytes in the I/O memory as follows:

- I/O memory width was set to 32-bit - each control byte controls two steps.
- I/O memory width was set to 16-bit - each control byte controls four steps.
- I/O memory width was set to 8-bit - each control byte controls eight steps.

See *Appendix B* for control memory content.

Note: The board must be in HALT state before calling this function.

The program counter will is set to zero upon return.

Example

The following example reads 64 steps from the Master board input memory beginning at step 128:

```
DWORD adwControl[64];
DioReadCtrlMemory (nMasterHandle, adwControl, 128, 64, &nStatus);
```

See Also

DioWriteCtrlMemory, DioGetErrorString

DioReadDirectionMemory

Applies To

GX5055, GX5290, GX5290e, GX5295, File

Purpose

Reads a block of direction memory.

Syntax

DioReadDirectionMemory *(nHandle, pvMemory, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	DIO or File board handle.
pvMemory	PVOID	Pointer to an array, array data type needs to comply with the I/O width settings and board type, see comments for details. Each index in the control memory array sets the direction for all the active channels (as was set by the I/O width settings) for the specified step. Each bit corresponds to a channel, i.e. bit 0 sets channel 0 and so on. Direction bits values are as follow: GX5290/GX5290e/GX5295: • Bit low – output. • Bit high – input. GX5055: • Bit low – input. • Bit high – output.
dwStart	DWORD	Starting address to read.
dwSize	DWORD	Number of steps to read.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The program counter will is set to zero upon return.

GX5055:

Array data type is Double Word (DWORD) and maximum array size is 512K of double words.

GX5290/GX5290e/GX5295:

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291/GX5291e: Maximum array size is 32M of double words. GX5292/GX5292e: Maximum array size is 64M of double words. GX5293/GX5293e: Maximum array size is 64M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). GX5291/GX5291e: Maximum array size is 64M words. GX5292/GX5292e: Maximum array size is 128M words. GX5293/GX5293e: Maximum array size is 128M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). GX5291/GX5291e: Maximum array size is 128MB. GX5292/GX5292e: Maximum array size is 256MB. GX5293/GX5293e: Maximum array size is 256MB.
4, 2, 1	When configuring the Number of channels to be less than 8 bits (4, 2 or 1) the data can be unpacked after reading it using a Double Word array by using DioDataUnpack function. Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291/GX5291e: Maximum array size is 32M of double words. GX5292/GX5292e: Maximum array size is 64M of double words. GX5293/GX5293e: Maximum array size is 64M of double words.

Note: The board must be in HALT state before calling this function.

Number of channels	Input memory Max Number of steps	Output memory Max Number of steps	Control memory Max Number of steps
32	67108608	67108864	67108864
16	134217216	134217728	134217728
8	268434432	268435456	268435456
4	536868864	536870912	536870912
2	1073737728	1073741824	1073741824
1	2147475456	2147483648	2147483648

Example

The following example reads 64 steps from the Master board direction memory beginning at step 128:

```
DWORD adwControl[64];
DioReadDirectionMemory (nMasterHandle, adwControl, 128, 64, &nStatus);
```

See Also

DioWriteCtrlMemory, DioGetErrorString

DioReadInMemory

Applies To

GC5050, GX5050, GX5055, GX5290, GX5290e, GX5295, File

Purpose

Reads a block of input memory from the specified board (Master or Slave).

Syntax

DioReadInMemory *(nHandle, pvMemory, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	**Comments**
nHandle	SHORT	DIO or File board handle.
pvMemory	PVOID	Pointer to an array, array data type needs to comply with the I/O width settings and board type, see comments for details.
dwStart	DWORD	Starting step to read from.
dwSize	DWORD	Number of steps to read.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The program counter will is set to zero upon return.

GX5050/GX5050:

Array data type is Double Word (DWORD) and maximum array size is 1M of double words.

GX5055:

Array data type is Double Word (DWORD) and maximum array size is 512K of double words.

GX5290/GX5290e/GX5295:

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291/GX5291e: Maximum array size is 32M of double words. GX5292/GX5292e: Maximum array size is 64M of double words. GX5293/GX5293e: Maximum array size is 64M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). GX5291/GX5291e: Maximum array size is 64M words. GX5292/GX5292e: Maximum array size is 128M words. GX5293/GX5293e: Maximum array size is 128M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). GX5291/GX5291e: Maximum array size is 128MB. GX5292/GX5292e: Maximum array size is 256MB. GX5293/GX5293e: Maximum array size is 256MB.
4, 2, 1	When configuring the Number of channels to be less than 8 bits (4, 2 or 1) the data can be unpacked after reading it using a Double Word array by using DioDataUnpack function. Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291/GX5291e: Maximum array size is 32M of double words. GX5292/GX5292e: Maximum array size is 64M of double words. GX5293/GX5293e: Maximum array size is 64M of double words.

Note: The board must be in HALT state before calling this function.

Number of channels	Input memory Max Number of steps	Output memory Max Number of steps	Control memory Max Number of steps
32	67108608	67108864	67108864
16	134217216	134217728	134217728
8	268434432	268435456	268435456
4	536868864	536870912	536870912
2	1073737728	1073741824	1073741824
1	2147475456	2147483648	2147483648

Example

The following example reads 64 steps from the Master board input memory beginning at step 128:

```
DWORD adwInput[64];
DioReadInMemory (nHandle, adwInput, 128, 64, &nStatus);
```

See Also

DioWriteInMemory, DioWriteOutMemory, DioGetErrorString

DioReadIOMemory

Applies To

GX5150, GX5280, File

Purpose

Reads a block of data from the specified board (Master or Slave).

Syntax

DioReadIOMemory *(nHandle, pvMemory, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	DIO or File board handle.
pvMemory	PVOID	Pointer to an array, array data type needs to comply with the I/O width settings and board type, see comments for details.
dwStart	DWORD	Starting step to read from.
dwSize	DWORD	Number of steps to store in array.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The program counter will is set to zero upon return.

GX5150:

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). Maximum array size is 32M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). Maximum array size is 64M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). Maximum array size is 128MB.

GX5280:

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5281: Maximum array size is 32M of double words. GX5282: Maximum array size is 64M of double words. GX5283: Maximum array size is 128M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). GX5281: Maximum array size is 64M words. GX5282: Maximum array size is 128M words. GX5283: Maximum array size is 256M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). GX5281: Maximum array size is 128MB. GX5282: Maximum array size is 256MB. GX5283: Maximum array size is 512MB.
4, 2, 1	When configuring the Number of channels to be less than 8 bits (4, 2 or 1) the data can be unpacked after reading it using a Double Word array by using DioDataUnpack function. Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5281: Maximum array size is 32M of double words. GX5282: Maximum array size is 64M of double words. GX5283: Maximum array size is 128M of double words.

Note: The board must be in HALT state before calling this function.

Example

The following example reads a block of 64 DWORDs starting at step 128 (width is set to 32-bit):

```
SHORT   nStatus, nTermination;
DWORD   dwSteps[64];
DioReadIOMemory(nHandle, dwSteps, 128, 64, &nStatus);
```

See Also

DioWriteIOMemory, DioSetupIOConfiguration, DioGetIOConfiguration, DioGetErrorString

DioReadIOPinsValidity

Applies To

GX5055, GX5295

Purpose

Read back the validity of the current input pins levels.

Syntax

DioReadIOPinsValidity *(nHandle, pdwValidity, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or Slave board handle.
pdwValidity	PDWORD	Read back the validity of the current input pins levels. Each bit represents a channel, e.g. bit 0 represents the validity of the input voltage at channels 0. A low represent valid input voltage and a high represent invalid input voltage, see comments for details.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function should be used with conjunction of **DioReadIOPinsValue** in order to get a complete picture of each input voltage level state. The following table summarizes the different results on a per bit basis:

Bit x read with DioReadIOPinsValidity	Bit x read with DioReadIOPinsValue	Comments
0	0	Valid input voltage and input voltage level is less than the input low threshold level.
	1	Valid input voltage and input voltage level is greater than the input high threshold level.
1	x	Invalid input voltage, input voltage is greater than input low threshold level and less then high threshold level

Example

The following example reads the I/O pins values:

```
SHORT   nStatus;
DWORD   dwValidity;
DioReadIOPinsValidity (nHandle, &dwValidity, &nStatus);
```

See Also

DioWriteIOPinsValue, DioReadIOPinsValue, DioGetIOPinsStaticDirection, DioSetupIOPinsStaticDirection, DioGetErrorString

DioReadIOPinsValue

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Read back the current input pins value.

Syntax

DioReadIOPinsValue *(nHandle, pdwValue, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or Slave board handle.
pdwValue	PDWORD	Data read from the I/O pins.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GC5050, GX5050, GX5150, GX5280:

- The board must be in a HALT or PAUSE state in order for the user to read the I/O lines.
- If the memory I/O configuration width was set to 16-bit or 8-bit, then only the 16 or 8 lower bits are applicable and the remaining bits return 0.

GX5055:

- The board must be in a HALT or PAUSE state in order for the user to read the I/O lines.
- If the memory I/O configuration width was set to 16-bit or 8-bit, then only the 16 or 8 lower bits are applicable and the remaining bits return 0.

The function should be used with conjunction of **DioReadIOPinsValidity** in order to get a complete picture of each input voltage level state. The following table summarizes the different results on a per bit basis:

Bit x read with DioReadIOPinsValidity	Bit x read with DioReadIOPinsValue	Comments
0	0	Valid input voltage and input voltage level is less than the input low threshold level.
0	1	Valid input voltage and input voltage level is greater than the input high threshold level.
1	X	Invalid input voltage, input voltage is greater than input low threshold level and less then high threshold level

GX5290/GX5290e:

- Firmware version 0x8828 and above: allow reading the I/O pins data when the card is in PUASE/HALT state, other wise the board must be in a HALT state.
- **GX5295:**
- The board must be in a HALT or PAUSE state in order for the user to read the I/O lines.

The function should be used with conjunction of **DioReadIOPinsValidity** in order to get a complete picture of each input voltage level state. The following table summarizes the different results on a per bit basis:

Bit x read with DioReadIOPinsValidity	Bit x read with DioReadIOPinsValue	Comments
0	0	Valid input voltage and input voltage level is less than the input low threshold level.
0	1	Valid input voltage and input voltage level is greater than the input high threshold level.
1	X	Invalid input voltage, input voltage is greater than input low threshold level and less then high threshold level

Example

The following example reads the I/O pins values:

```
SHORT   nStatus;
DWORD   dwValue;
DioReadIOPinsValue (nHandle, &dwValue, &nStatus);
```

See Also

DioWriteIOPinsValue, DioSetupIOConfiguration, DioGetIOConfiguration, DioGetErrorString

DioReadInputState

Applies To

GX5055, GX5295

Purpose

Reads the specified channel's input signal.

Syntax

DioReadInputState *(nHandle, nChaanel, pnState, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or Slave board handle.
nChannel	SHORT	Specified channel in the DIO domain, channel can be between 0 to the last channel number in the domain. E.g. if the domain has two DIO boards, then the channel range is 0-63. GX5295: Auxiliary channels (Master only): DIO_AUX_CHANNEL_0 (1000) to DIO_AUX_CHANNEL_3 (1003).
pnState	PSHORT	Reads the channel's input state: 0=Input signal is lower than low and high threshold value. -1=Input signal is between the low and high threshold value (invalid signal level). 1=Input signal is above the low and high threshold value.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function can be called at any point, even while the DIO domain is running.

NOTE: it is recommended to use this function in order to read the channels' input states rather then using **DioReadIOPinsValue** and **DioReadIOPinsValidity** APIs.

Example

The following example reads the input of channel 0 and print out the channel's state:

```
SHORT   nStatus;
SHORT   nState;
DioReadInputState (nHandle, 0, &nState, &nStatus);
if (nState==0)
    printf ("Channel 0 input is valid logic low");
else if (nState==-1)
    printf ("Channel 0 input is invalid");
else if (nState==1)
    printf ("Channel 0 input is valid logic high");
```

See Also

DioWriteIOPinsValue, DioReadIOPinsValidity, DioSetupIOConfiguration, DioGetIOConfiguration, DioGetErrorString

DioReadOutMemory

Applies To

GC5050, GX5050, GX5055, GX5290, GX5290e, GX5295, File

Purpose

Reads a block of output memory from the specified board (Master or Slave).

Syntax

DioReadOutMemory *(nHandle, pvMemory, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	DIO or File board handle.
pvMemory	PVOID	Pointer to an array, array data type needs to comply with the I/O width settings and board type, see comments for details.
dwStart	DWORD	Starting step to read from.
dwSize	DWORD	Number of steps to read.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The program counter will is set to zero upon return.

GX5050/GX5050:

Array data type is Double Word (DWORD) and maximum array size is 1M of double words.

GX5055:

Array data type is Double Word (DWORD) and maximum array size is 512K of double words.

GX5290/GX5290e/GX5295:

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291/GX5291e: Maximum array size is 32M of double words. GX5292/GX5292e: Maximum array size is 64M of double words. GX5293/GX5293e: Maximum array size is 64M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). GX5291/GX5291e: Maximum array size is 64M words. GX5292/GX5292e: Maximum array size is 128M words. GX5293/GX5293e: Maximum array size is 128M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). GX5291/GX5291e: Maximum array size is 128MB. GX5292/GX5292e: Maximum array size is 256MB. GX5293/GX5293e: Maximum array size is 256MB.
4, 2, 1	When configuring the Number of channels to be less than 8 bits (4, 2 or 1) the data can be unpacked after reading it using a Double Word array by using DioDataUnpack function. Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291/GX5291e: Maximum array size is 32M of double words. GX5292/GX5292e: Maximum array size is 64M of double words. GX5293/GX5293e: Maximum array size is 64M of double words.

Note: The board must be in HALT state before calling this function.

Number of channels	Input memory Max Number of steps	Output memory Max Number of steps	Control memory Max Number of steps
32	67108608	67108864	67108864
16	134217216	134217728	134217728
8	268434432	268435456	268435456
4	536868864	536870912	536870912
2	1073737728	1073741824	1073741824
1	2147475456	2147483648	2147483648

Example

The following example reads 64 steps from the Master board input memory beginning at step 128:

```
DWORD adwInput[64];
DioReadOutMemory (nHandle, adwInput, 128, 64, &nStatus);
```

See Also

DioWriteInMemory, DioWriteOutMemory, DioGetErrorString

DioReadProgramCounter

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Returns the program counter value of the specified Master board.

Syntax

DioReadProgramCounter *(nMasterHandle, pdwCounter, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
pdwCounter	PDWORD	Returned program counter value (address).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GC5050/GX5050:

Set the address of the input, output and control memories. The program counter may be changed only by a control memory command in the RUN state, or by calling a function that resets the program counter to 0, such as **DioReset**, **DioSaveFile**, etc. The control memory commands that change the value of the program counter are JUMP, LOOP, CALL and RETURN.

GX5055:

Set the address of the input, output, direction, valid data and control memories. The program counter may be changed only by a control memory command in the RUN state, or by calling a function that resets the program counter to 0, such as **DioReset**, **DioSaveFile**, etc. The control memory commands that change the value of the program counter are JUMP, LOOP, CALL and RETURN.

GX5150:

Set the address of the I/O and control memories. The program counter may be changed only by a control memory command in the RUN state, or by calling a function that resets the program counter to 0, such as **DioReset**, **DioSaveFile**, etc. The control memory commands that change the value of the program counter are JUMP, LOOP, CALL and RETURN.

GX5280/GX5290/GX5290e/GX5295:

GX5280: Sets the address of the I/O memory.

GX5290/GX5290e/GX5295: Sets the address of the I/O and direction memories. The program counter may be changed only by a control memory command in the RUN state, or by calling a function that resets the program counter to 0, such as **DioReset**, **DioSaveFile**, etc.

When the number of active channels is less than 32, the program counter value will be rounded to be on a 32 channels boundaries. E.g. if the I/O configuration was set to 16 channels and the program counter value was set to 1025 the value will be rounded to 1024.

Note: when reading back the program counter value it will correspond to the actual active number of channels as follows:

Number of active channels	Program Counter range
32	GX5281: 0 to 32M GX5282: 0 to 64M GX5283: 0 to 128M GX5291/GX5291e: 0 to 32M GX5292/GX5292e: 0 to 64M GX5293/GX5293e: 0 to 64M GX5295: 0 to 64M
16	GX5281: 0 to 64M GX5282: 0 to 128M GX5283: 0 to 256M GX5291/GX5291e: 0 to 64M GX5292/GX5292e: 0 to 128M GX5293/GX5293e: 0 to 128M GX5295: 0 to 128M
8	GX5281: 0 to 128M GX5282: 0 to 256M GX5283: 0 to 512M GX5291/GX5291e: 0 to 128M GX5292/GX5292e: 0 to 256M GX5293/GX5293e: 0 to 256M GX5295: 0 to 256M
4	GX5281: 0 to 256M GX5282: 0 to 512M GX5283: 0 to 1G GX5291/GX5291e: 0 to 256M GX5292/GX5292e: 0 to 512M GX5293/GX5293e: 0 to 512M GX5295: 0 to 512M
2	GX5281: 0 to 512M GX5282: 0 to 1G GX5283: 0 to 2G GX5291/GX5291e: 0 to 512M GX5292/GX5292e: 0 to 1G GX5293/GX5293e: 0 to 1G GX5295: 0 to 1G
1	GX5281: 0 to 1G GX5282: 0 to 2G

GX5283: 0 to 4G
GX5291/GX5291e: 0 to 1G
GX5292/GX5292e: 0 to 2G
GX5293/GX5293e: 0 to 2G
GX5295: 0 to 2G

Example

The following example waits until the board is in the HALT state and then retrieves the last program counter value:

```
#define HALT_ADDRESS_ERROR1   0x128
DioTrig (nMasterHandle, &nStatus);
do
{  DioReadStatusRegister (nMasterHandle, &wData,     &nStatus);
}while (wData & 0x1C ==0);  // mask the STATE bits
DioReadProgramCounter (nMasterHandle, &dwCounter,    &nStatus);if (dwCounter ==
HALT_ADDRESS_ERROR1)
   // Do something
```

See Also

DioWriteProgramCounter, **DioReset, DioGetErrorString**

DioReadValidDataMemory

Applies To

GX5055, File

Purpose

Reads a block of data to the valid data memory of the specified board (Master or Slave).

Syntax

DioReadValidDataMemory *(nHandle, pvMemory, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
pvMemory	PVOID	Pointer to an array, array data type is Double Word (DWORD) and maximum array size is 512K of double words.
dwStart	DWORD	Starting address to write.
dwSize	DWORD	Number of steps to write.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The program counter will is set to zero upon return.

The input voltage can be inside one of the following ranges:

- Input voltage is higher than high voltage threshold, input is logic high. Data will be logged as logic high to the input memory and 0 to the invalid logic level input memory.
- Input voltage is lower than low voltage threshold, input is logic low. Data will be logged as logic low to the input memory and 0 to the invalid logic level input memory.
- Input voltage is higher than low voltage threshold and lower then high voltage threshold, input is invalid. Data will be logged as logic low to the input memory and 1 to the invalid logic level input memory.

Note: The board must be in HALT state before calling this function.

Example

The following example writes 64 steps to the Master board's valid data memory beginning at step 128:

```
DWORD adwData[64];
DioReadValidDataMemory (nHandle, adwData, 128, 64, &nStatus);
```

See Also

DioWriteValidDataMemory, DioWriteCtrlMemory, DioReadCtrlMemory, DioReadInMemory, DioReadOutMemory

DioReadStatusRegister

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Reads the status register from the specified board.

Syntax

DioReadStatusRegister *(nHandle, lpwData, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
lpwData	PWORD	Returns status register value. See Comments.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GC5050, GX5050:

The status register is a 16-bit register (bits 0 – 15) configured as follows:

Bits	15 12	11 10	9	8	7	6 5	4 2	1 0
Mnemonic	BRDID	R	OVF	L	C	DEQ	STATE	TM

Where:

Mnemonic	Comments
BRDID	4 bits (15-12). Board to ID number. The following values are available: 0xA (1010) - GT-50 Master board 0xB (1011) - GT-50 Slave board. 0x2 (0010) - GT-25 Master board 0x3 (0011) - GT-25 Slave board.
R	Reserved.
OVF	1 bit (9). Address overflow for program counter overflow.
L	1 bit (8). Loop overflow. This bit is set to 1 when a register used by the current command is set to 0.
C	1 bit (7). Condition state. This bit is set to 0 when the current command condition is TRUE.
DEQ	2 bits (6-5). These bits are set to 0 when external event lines ANDed with D mask register are equal to the D event register. Each bit represents 8 lines (byte) where bit 5 is for event lines 0-7 (LSB) and bit 5 is for event lines 8-15 (MSB).
STATE	3 bits (4-2). These bits represent the current board state. The following binary values are available: 000 - HALT The board is in HALT state. 001 - WAIT D Waiting for D level trigger. 010 - WAIT T Waiting for T level trigger. 011 - PAUSE The board is in PAUSE state. 1xx - RUN The board is in RUN state.

TM	2 bits (1-0). These bits represent the current trigger mode. See **DioSetupTriggerMode** for more information. The following values are available: 00 - Trigger mode is 1 (D). 01 - Trigger mode is 2 (T). 10 - Trigger mode is 3 (D and T). 11 - Trigger mode is 4 (T and D).

GX5055:

The status register is a 16-bit register (bits 0 – 15) configured as follows:

Bits	15 12	11 10	9	8	7	6 5	4 2	1 0
Mnemonic	BRDID	R	OVF	L	C	RFU	STATE	RFU

Where:

Mnemonic	Comments
BRDID	4 bits (15-12). Board to ID number. The following values are available: 0xA (1010) - GT-50 Master board 0xB (1011) - GT-50 Slave board. 0x2 (0010) - GT-25 Master board 0x3 (0011) - GT-25 Slave board.
R	Reserved.
OVF	1 bit (9). Address overflow for program counter overflow.
L	1 bit (8). Loop overflow. This bit is set to 1 when a register used by the current command is set to 0.
C	1 bit (7). Condition state. This bit is set to 0 when the current command condition is TRUE.
STATE	3 bits (4-2). These bits represent the current board state. The following binary values are available: 000 - HALT The board is in HALT state. 001 - WAIT D Waiting for D level trigger. 010 - WAIT T Waiting for T level trigger. 011 - PAUSE The board is in PAUSE state. 1xx - RUN The board is in RUN state.

GX5150:

The status register is a 16-bit register (bits 0 – 15) configured as follows:

Bits	10	9	8	7	6	5	4 2	1 0
Mnemonic	Seq	AOF	Chalt	CPuase	RFlag	DEQ	State	TM

Where:

Mnemonic	Comments
SEQ	1 Bit [10]. Sequencer state. 0 Disable 1 Enable
AOF	1 Bit (9). Address Counter Overflow. Read back of '1' indicates an overflow condition.
CHLT	1 Bit (8). HALT Condition. Read back of '1' indicates the vector set the sequencer to HALT state.
CPSE	1 Bit (7). PAUSE Condition. Read back of '1' indicates the vector set the sequencer to PAUSE state.
RFLG	1 Bit (6). Run Flag Indicator. Read back of '1' indicates a RUN mode has occurred since the last read of this register bit. This bit is used in conditions where a short RUN sequence can end before the PC is able to read the real-time 'RUN' bit.
DEQ	1 bit (5). This bit is set to 1 when external event lines ANDed with D mask register are equal to the D event register.
STATE	3 bits (4-2). These bits represent the current board state. The following binary values are available: 000 - HALT THE BOARD IS IN HALT STATE. 001 - WAIT D Waiting for D level trigger. 010 - WAIT T Waiting for T level trigger. 011 - PAUSE The board is in PAUSE state. 1xx - RUN The board is in RUN state.
TM	2 bits (1-0). These bits represent the current trigger mode. See **DioSetupTriggerMode** for more information. The following values are available: 00 - Trigger mode is 1 (D). 01 - Trigger mode is 2 (T). 10 - Trigger mode is 3 (D and T). 11 - Trigger mode is 4 (T and D).

GX528X/GX529X/GX529Xe/GX5295:

The status register is a 16-bit register (bits 0 – 15) configured as follows:

Bits	9	8 – 7	6	5	4 2	1 0
Mnemonic	AOF	Not Used	RFlag	DEQ	State	TM

Where:

Mnemonic	Comments
AOF	1 Bit (9). Address Counter Overflow. Read back of '1' indicates an overflow condition.
RFLG	1 Bit (6). Run Flag Indicator. Read back of '1' indicates a RUN mode has occurred since the last read of this register bit. This bit is used in conditions where a short RUN sequence can end before the PC is able to read the real-time 'RUN' bit.
DEQ	1 bit (5). This bit is set to 1 when external event lines ANDed with D mask register are equal to the D event register.
STATE	3 bits (4-2). These bits represent the current board state. The following binary values are available: 000 - HALT THE BOARD IS IN HALT STATE. 001 - WAIT D Waiting for D level trigger. 010 - WAIT T Waiting for T level trigger. 011 - PAUSE The board is in PAUSE state. 1xx - RUN The board is in RUN state.
TM	2 bits (1-0). These bits represent the current trigger mode. See **DioSetupTriggerMode** for more information. The following values are available: 00 - Trigger mode is 1 (D). 01 - Trigger mode is 2 (T). 10 - Trigger mode is 3 (D and T). 11 - Trigger mode is 4 (T and D).

Example

The following example waits until the board is in the HALT state:

```
do
{ DioReadStatusRegister (nMasterHandle, &wData, &nStatus);
} while (wData & 0x1C == 0);  // mask the STATE bits
```

See Also

DioSetupTriggerMode, DioGetTriggerMode, DioTrig, DioArm, DioHalt, DioGetErrorString

DioReadXRegister

DioWriteXRegister

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Reads or Writes the X register to/from the specified Master board.

Syntax

DioReadXRegister *(nMasterHandle, pwData, pnStatus)*

DioWriteXRegister *(nMasterHandle, wData, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board or File board handle.
pwData	PWORD	Returned X register value.
wData	WORD	Register X value to be written.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function is used to simulate external event line values. **DioSetupTriggerXEventSource** must be called in order to simulate the external event lines.

Example

The following example sets the X register event source to internal, writes to the X register the value 0xA5A5, and reads it back from the X register to verify its content:

```
SHORT   nStatus;
WORD    wData;
DioSetupTriggerXEventSource(nMasterHandle, 1, &nStatus);
DioWriteXRegister(nMasterHandle, 0xA5A5, &nStatus);
DioReadXRegister(nMasterHandle, &wData, &nStatus);
```

See Also

DioSetupTriggerXEventSource, DioGetErrorString

DioRealTimeCompareClearChannelsStatus

Applies To

GX5295

Purpose

Clears all the channnels' in the domain per channel failed status.

Syntax

DioRealTimeCompareClearChannelsStatus (*nMasterHandle, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Board handle.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

All boards in the domain have to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

All boards in the domain have to set the Real Time Compare mode to DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL mode using **DioRealTimeCompareSetupMode** API prior calling this function. See the **DioRealTimeCompareSetupMode** function for more details.

Note: Real Time Compare functionality is supported only by GX529x boards with firmware versions 0x8A00 and above.

Note: Real Time Compare with per channel fail functionality is supported only by GX5295 boards with firmware versions 0xF605 and above.

Arming clears all the channnels' in the domain per channel failed status.

Example

```
DioRealTimeCompareClearResultMemory (nHandle, &nStatus);
```

See Also

DioRealTimeCompareReadChannelsStatus, DioRealTimeCompareSetupActiveChannels, DioRealTimeCompareSetupMode, DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareClearResultMemory

Applies To

GX5290, GX5295, File

Purpose

Clears all the onboard comparison result memory.

Syntax

DioRealTimeCompareClearResultMemory (*nHandle, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

This function is called by the driver internally whenever the **DioArm** command is called.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

The onboard comparison result memory is cleared each time the board is triggered.

Example

```
DioRealTimeCompareClearResultMemory (nHandle, &nStatus);
```

See Also

DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareGetActiveChannels

Applies To

GX5295, File

Purpose

Returns the specified board active channels when in real time compare per channel fail mode.

Syntax

DioRealTimeCompareGetActiveChannels (*nHandle, pdwActiveChannels, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
pdwActiveChannels	PDWORD	Returns the specified board active channels when in real time compare per channel fail mode. Each bit represents a channel, channel 0 = bit 0, channel 31 = bit 31 etc. Bit Low (0): The respected channel will no be compared. Bit High (1): The respected channel will no be compared.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

All boards in the domain have to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

All boards in the domain have to set the Real Time Compare mode to DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL mode using **DioRealTimeCompareSetupMode** API prior calling this function. See the **DioRealTimeCompareSetupMode** function for more details.

Note: Real Time Compare functionality is supported only by GX529x boards with firmware versions 0x8A00 and above.

Note: Real Time Compare with per channel fail functionality is supported only by GX5295 boards with firmware versions 0xF605 and above.

Example

The following example initializes a master board in slot 6 and a slave board in slot 7. Sets both boards to operate in Real Time Compare mode. Sets the domain to be in Real Time Compare mode, Set the Real Time Compare mode to DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL mode, and enables all even channels in the master board. And reads the board active channels real time compare pass fail statues

```
SHORT nMasterHandle1, nSlave1, nDensity, nBanks, nStatus;
DWORD dwChannelsStatus, dwActiveChannels;

DioSetupInitialization (0, 1, 0x106, &nDensity, &nBanks, &nMasterHandle1, &nStatus);
DioSetupInitialization (nMasterHandle1, 1, 0x107, &nDensity, &nBanks, &nSlave11, &nStatus);
DioDomainSetupOperatingMode (nMasterHandle, DIO_OPERATING_MODE_REAL_TIME_COMPARE, &nStatus);
DioRealTimeCompareSetupMode (nMasterHandle, DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL, &nStatus);
DioRealTimeCompareClearResultMemory (nMasterHandle, DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL,
&nStatus);
DioRealTimeCompareSetupActiveChannels(nMasterHandle, 0xAAAAAAAA, &nStatus);
DioRealTimeCompareGetActiveChannels(nMasterHandle, &dwActiveChannels, &nStatus);
DioRealTimeCompareReadChannelsStatus (nMasterHandle, &dwChannelsStatus, pnStatus)
```

See Also

DioRealTimeCompareClearChannelsStatus, DioRealTimeCompareReadChannelsStatus, DioRealTimeCompareSetupActiveChannels, DioRealTimeCompareSetupMode, DioRealTimeCompareReadChannelsStatus, DioRealTimeCompareSetupActiveChannels, DioRealTimeCompareSetupMode, DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareGetConditionValue

Applies To

GX5290, GX5295, File

Purpose

Returns the specified condition value.

Syntax

DioRealTimeCompareGetConditionValue (*nHandle, nCondition, pdwValue, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
nCondition	SHORT	Condition: 0. DIO_RTC_CONDITION_FAILURE_COUNT: the DIO domain will PAUSE/HALT once the number of detected errors equals the condition count value. 1. DIO_RTC_CONDITION_DATA: the DIO domain will PAUSE/HALT once the vector data matches the condition data value. 2. DIO_RTC_CONDITION_PROGRAM_COUNTER: the DIO domain will PAUSE/HALT once the vector step number matches the condition address value.
pdwValue	PDWORD	Condition value. If *nCondition* is set to DIO_RTC_CONDITION_FAILURE_COUNT then the condition value can be 1 to 1024. If *nCondition* is set to DIO_RTC_CONDITION_DATA then the condition value can be between 0 to 0xFFFFFFFF. If *nCondition* is set to DIO_RTC_CONDITION_PROGRAM_COUNTER then the condition value can be between 0 to last step of 67108863, or higher if number of channels was set to less than 32.
pdwValue	PDWORD	Condition value.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each DIO in the domain can have its own active condition to stop on as it was set by calling **DioRealTimeCompareSetupStopMode** function.

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example returns the condition count value:

```
SHORT   nStatus;
DWORD   dwValue;
DioRealTimeCompareGetConditionValue (nHandle, DIO_RTC_CONDITION_FAILURE_COUNT, &dwValue,
pnStatus);
```

See Also

DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareGetDataDelay

Applies To

GX5290, GX5295, File

Purpose

Returns the specified data source clock cycles delay.

Syntax

DioRealTimeCompareGetDataDelay (*nHandle, nDataType, pdwCycles, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
nDataType	SHORT	Data type: 0. DIO_RTC_EXPECTED_DATA: sets the number of clock cycles that the expected memory data will be delayed by from the strobe. 1. DIO_RTC_MASKED_DATA: sets the number of clock cycles that the mask memory data will be delayed by from the clock.
pdwCycles	PDWORD	Data clock cycles delay value. Value can be between 0-15
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function sets the number of clocks cycles that the expected and masked memory data will be delayed by. Since each data in the comparison process has its own delay the function can be used to align the three different data sources, input data, masked data and expected data to be correctly compared.

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example Returns the expected memory data source clock cycles delay:

```
DWORD  dwCycles;
DioRealTimeCompareGetDataDelay (nHandle, DIO_RTC_EXPECTED_DATA, &dwCycles, &nStatus);
```

See Also

DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareGetDomainFailStatus

Applies To

GX5295

Purpose

Returns a boolean value if the last run had any real time compare failures.

Syntax

DioRealTimeCompareGetActiveChannels (*nHandle, pbFailStatus, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
pbFailStatus	PBOOL	Returns a boolean value if the last run had any real time compare failures. 0 No real time compare failures 1 Atleast one real time compare failure.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

All boards in the domain have to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

All boards in the domain have to set the Real Time Compare mode to DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL mode using **DioRealTimeCompareSetupMode** API prior calling this function. See the **DioRealTimeCompareSetupMode** function for more details.

Note: Real Time Compare functionality is supported only by GX529x boards with firmware versions 0x8A00 and above.

Note: Real Time Compare with per channel fail functionality is supported only by GX5295 boards with firmware versions 0xF605 and above.

Example

```
The following example initializes a master board in slot 6 and a slave board in slot 7. Sets both
boards to operate in Real Time Compare mode. Sets the domain to be in Real Time Compare mode, Set
the Real Time Compare mode to DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL mode, and enables all even
channels in the master board. And reads the board active channels real time compare pass fail
statues

SHORT nMasterHandle1, nSlave1, nDensity, nBanks, nStatus;
DWORD dwChannelsStatus, dwActiveChannels;

DioSetupInitialization (0, 1, 0x106, &nDensity, &nBanks, &nMasterHandle1, &nStatus);
DioSetupInitialization (nMasterHandle1, 1, 0x107, &nDensity, &nBanks, &nSlave11, &nStatus);
DioDomainSetupOperatingMode (nMasterHandle, DIO_OPERATING_MODE_REAL_TIME_COMPARE, &nStatus);
DioRealTimeCompareSetupMode (nMasterHandle, DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL, &nStatus);
DioRealTimeCompareClearResultMemory (nMasterHandle, DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL,
&nStatus);
DioRealTimeCompareSetupActiveChannels(nMasterHandle, 0xAAAAAAAA, &nStatus);
DioRealTimeCompareGetActiveChannels(nMasterHandle, &dwActiveChannels, &nStatus);
DioRealTimeCompareReadChannelsStatus (nMasterHandle, &dwChannelsStatus, pnStatus)
```

See Also

DioRealTimeCompareClearChannelsStatus, DioRealTimeCompareReadChannelsStatus, DioRealTimeCompareSetupActiveChannels, DioRealTimeCompareSetupMode, DioRealTimeCompareReadChannelsStatus, DioRealTimeCompareSetupActiveChannels, DioRealTimeCompareSetupMode, DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareGetFailureCount

Applies To

GX5290, GX5295, File

Purpose

Returns the number of comparison that failed**.**

Syntax

DioRealTimeCompareGetFailureCount (*nHandle, pdwCount, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
pdwCount	PDWORD	Returned status: 0 on success, negative number on failure.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example returns the number of comparison that failed:

```
DWORD   dwCount;
DioRealTimeCompareGetFailureCount (nHandle, &dwCount, &nStatus);
```

See Also

DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareGetInputDataClockEdge

Applies To

GX5290, GX5295, File

Purpose

Returns the input data strobe edge.

Syntax

DioRealTimeCompareGetInputDataClockEdge *(nHandle, pnClockEdge, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
pnClockEdge	PSHORT	Input data clock edge: 0. DIO_RISING_EDGE: Input data clock rising edge 1. DIO_FALLING_EDGE: Input data clock falling edge
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The response data which is clocked by the strobe is fed to the real time comparator using the same clock as the output data and the mask data. The user can set the input data (response data) clock edge to either capture the input data on the falling or rising edge. This can assist in aligning the input data with the other two sources of data, the masked and expected memory data to correctly compare.

The Rising/Falling parameter defines the edge of the data clock that is used to transfer the data sampled by the strobe into the input pipeline. This parameter is used to avoid meta stable conditions where the sample strobe and the data clock might overlap, causing ambiguous data storage. Using the falling edge of the data clock to transfer sampled data to the input pipeline places the data in the pipeline one cycle earlier than using the rising edge of the data clock (it uses the falling edge of the previous vector clock cycle not the rising edge of the current vector clock cycle).

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example returns the input data clock edge:

```
SHORT   nClockEdge;
DioRealTimeCompareGetInputDataClockEdge (nHandle, &nClockEdge, &nStatus);
```

See Also

DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareGetMode

Applies To

GX5295, File

Purpose

Returns the Real Time Compare mode.

Syntax

DioRealTimeCompareGetMode (*nHandle, pnMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
pnMode	PSHORT	Returns the Real Time Compare mode, modes cane be as follows: 0 DIO_RTC_MODE_DEFAULT: Default Real Time Compare mode operating mode. Supported by supported only by GX529x boards with firmware versions 0x8A00 and above. 1 DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL: In this mode all default Real Time Compare operating mode are supported, but in addition when running vectors only the first failure per channels will be logged. Supported by supported only by GX5295 boards with firmware versions 0xF605 and above.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL Real Time Compare mode is useful for a go-no-go testing where the user wants to know which channels failed at least once and the failure vector address.

NOTE: All boards in the domain have to set the Real Time Compare mode to DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL mode using **DioRealTimeCompareSetupMode** API prior calling this function. See the **DioRealTimeCompareSetupMode** function for more details.

Example

```
The following example initializes a master board in slot 6 and a slave board in slot 7. Sets both
boards to operate in Real Time Compare mode. Sets the domain to be in Real Time Compare mode, Set
the Real Time Compare mode to DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL mode, and enables all even
channels in the master board. And reads the board active channels real time compare pass fail
statues


SHORT nMasterHandle1, nSlave1, nDensity, nBanks, nStatus;
DWORD dwChannelsStatus, dwActiveChannels;


DioSetupInitialization (0, 1, 0x106, &nDensity, &nBanks, &nMasterHandle1, &nStatus);
DioSetupInitialization (nMasterHandle1, 1, 0x107, &nDensity, &nBanks, &nSlave11, &nStatus);
DioDomainSetupOperatingMode (nMasterHandle, DIO_OPERATING_MODE_REAL_TIME_COMPARE, &nStatus);
DioRealTimeCompareSetupMode (nMasterHandle, DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL, &nStatus);
DioRealTimeCompareClearResultMemory (nMasterHandle, DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL,
&nStatus);
DioRealTimeCompareSetupActiveChannels(nMasterHandle, 0xAAAAAAAA, &nStatus);
DioRealTimeCompareGetActiveChannels(nMasterHandle, &dwActiveChannels, &nStatus);
DioRealTimeCompareReadChannelsStatus (nMasterHandle, &dwChannelsStatus, pnStatus)
```

See Also

DioRealTimeCompareClearChannelsStatus, DioRealTimeCompareReadChannelsStatus, DioRealTimeCompareSetupActiveChannels, DioRealTimeCompareSetupMode, DioRealTimeCompareReadChannelsStatus, DioRealTimeCompareSetupActiveChannels, DioRealTimeCompareSetupMode, DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareGetResultsDataType

Applies To

GX5290, GX5295, File

Purpose

Return the compared results type.

Syntax

DioRealTimeCompareGetResultsDataType *(nHandle, pnDataType, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
pnDataType	PSHORT	Returned compared results type: 0. DIO_RTC_SAVE_COMPARED_DATA: saved data is the result of the incoming data masked by the mask memory and compared with the expected memory (default). 1. DIO_RTC_SAVE_INPUT_DATA: saved data is the raw input data that failed comparison.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The maximum member of compared data points is 1K. See the **DioRealTimeCompareReadResults** function for more details.

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example return the compared results type:

```
SHORT   nDataType;
DioRealTimeCompareGetResultsDataType (nHandle, &nDataType, &nStatus);
```

See Also

DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareGetStopCondition

Applies To

GX5290, File

Purpose

Returns the comparison stop condition.

Syntax

DioRealTimeCompareGetStopCondition (*nHandle, pnCondition, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
pnCondition	PSHORT	Comparison stop mode. 0. DIO_RTC_STOP_ON_FAILURES_COUNT: In this stop condition the board stops when the total number of failures is equal to the number of failures count. If the number of detected failures is less the condition value the board will run to the end of the vector. All boards in the domain will stop when any of the boards meet this condition. 1. DIO_RTC_STOP_ON_FAILURES_COUNT_SAVE_DATA: In this stop condition the board continuously comparing and recording the input data. The trigger for the board to stop is when the detected failures number is equal to the condition failures count number. At that point the DIO will continue to record and compare data until there is no more room in the results memory. In this condition the result memory is operating like a FIFO until stop condition is met. All boards in the domain will stop when any of the boards meet this condition. 2. DIO_RTC_STOP_ON_DATA_VALUE: In this stop condition the board continuously comparing and recording the input data. The trigger for the board to stop is when the input data is equal to the condition data value At that point the DIO will continue to record and compare data until there is no more room in the results memory. In this condition the result memory is operating like a FIFO until stop condition is met. All boards in the domain will stop when any of the boards meet this condition. 4. DIO_RTC_STOP_ON_PROGRAM_COUNTER: In this stop condition the board continuously comparing and record the input data. The trigger for the board to stop is when the program counter value is equal to the condition data value At that point the DIO will continue to record and compare data until there is no more room in the results memory. In this condition the result memory is operating like a FIFO until stop condition is met. All boards in the domain will stop when any of the boards meet this condition.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each boar in the DIO domain can have its own stop condition. As a result the first board that its stop condition is met will stop all other boards in the domain even if their stop condition is not met yet.

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example returns the comparison stop condition:

```
SHORT   nCondition;
DioRealTimeCompareGetStopCondition (nHandle, &nCondition, &nStatus);
```

See Also

DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareReadChannelsStatus

Applies To

GX5295, File

Purpose

Returns the specified board active channels real time compare pass fail statues.

Syntax

DioRealTimeCompareReadChannelsStatus (*nHandle, pdwChannelsStatus, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
pdwChannelsStatus	PDWORD	Returns the specified board active channels real time compare pass fail statues. Each bit represents a channel, channel 0 = bit 0, channel 31 = bit 31 etc. Bit Low (0): The respected channel passed real time compare. Bit High (1): The respected channel failed real time compare.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

All boards in the domain have to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

All boards in the domain have to set the Real Time Compare mode to DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL mode using **DioRealTimeCompareSetupMode** API prior calling this function. See the **DioRealTimeCompareSetupMode** function for more details.

Note: Real Time Compare functionality is supported only by GX529x boards with firmware versions 0x8A00 and above.

Note: Real Time Compare with per channel fail functionality is supported only by GX5295 boards with firmware versions 0xF605 and above.

Example

```
The following example initializes a master board in slot 6 and a slave board in slot 7. Sets both
boards to operate in Real Time Compare mode. Sets the domain to be in Real Time Compare mode, Set
the Real Time Compare mode to DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL mode, and enables all even
channels in the master board. And reads the board active channels real time compare pass fail
statues


SHORT nMasterHandle1, nSlave1, nDensity, nBanks, nStatus;
DWORD dwChannelsStatus, dwActiveChannels;


DioSetupInitialization (0, 1, 0x106, &nDensity, &nBanks, &nMasterHandle1, &nStatus);
DioSetupInitialization (nMasterHandle1, 1, 0x107, &nDensity, &nBanks, &nSlave11, &nStatus);
DioDomainSetupOperatingMode (nMasterHandle, DIO_OPERATING_MODE_REAL_TIME_COMPARE, &nStatus);
DioRealTimeCompareSetupMode (nMasterHandle, DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL, &nStatus);
DioRealTimeCompareClearResultMemory (nMasterHandle, DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL,
&nStatus);
DioRealTimeCompareSetupActiveChannels(nMasterHandle, 0xAAAAAAAA, &nStatus);
DioRealTimeCompareGetActiveChannels(nMasterHandle, &dwActiveChannels, &nStatus);
DioRealTimeCompareReadChannelsStatus (nMasterHandle, &dwChannelsStatus, pnStatus)
```

See Also

DioRealTimeCompareClearChannelsStatus, DioRealTimeCompareSetupActiveChannels, DioRealTimeCompareSetupMode, DioRealTimeCompareReadChannelsStatus, DioRealTimeCompareSetupActiveChannels, DioRealTimeCompareSetupMode, DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareReadExpectedMemory

Applies To

GX5290, GX5295, File

Purpose

Reads a block of expected memory from the specified board (Master or Slave).

Syntax

DioRealTimeCompareReadExpectedMemory (*nHandle, pvVector, dwStart, dwSize, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO or File board handle.
pvVector	PVOID	Pointer to an array, array data type needs to comply with the I/O width settings and board type, see comments for details.
dwStart	DWORD	Starting step to read from.
dwSize	DWORD	Number of steps to read.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The program counter will is set to zero upon return.

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291: Maximum array size is 32M of double words. GX5292: Maximum array size is 64M of double words. GX5293: Maximum array size is 64M of double words. GX5295: Maximum array size is 64M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). GX5291: Maximum array size is 64M words. GX5292: Maximum array size is 128M words. GX5293: Maximum array size is 128M words. GX5295: Maximum array size is 128M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). GX5291: Maximum array size is 128MB. GX5292: Maximum array size is 256MB. GX5293: Maximum array size is 256MB. GX5293: Maximum array size is 256MB.

Note: The board must be in HALT state before calling this function.

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example reads 64 steps from the Master board expected memory beginning at step 128:

```
DWORD adwMemory[64];
DioRealTimeCompareReadExpectedMemory (nHandle, adwMemory, 128, 64, &nStatus);
```

See Also

DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareReadMaskMemory

Applies To

GX5290, GX5295, File

Purpose

Reads a block of mask memory from the specified board (Master or Slave).

Syntax

DioRealTimeCompareReadMaskMemory (*nHandle, pvMemory, dwStart, dwSize, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO or File board handle.
pvMemory	PVOID	Pointer to an array, array data type needs to comply with the I/O width settings and board type, see comments for details.
dwStart	DWORD	Starting step to read from.
dwSize	DWORD	Number of steps to read.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The program counter will is set to zero upon return.

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291: Maximum array size is 32M of double words. GX5292: Maximum array size is 64M of double words. GX5293: Maximum array size is 64M of double words. GX5295: Maximum array size is 64M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). GX5291: Maximum array size is 64M words. GX5292: Maximum array size is 128M words. GX5293: Maximum array size is 128M words. GX5295: Maximum array size is 128M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). GX5291: Maximum array size is 128MB. GX5292: Maximum array size is 256MB. GX5293: Maximum array size is 256MB. GX5295: Maximum array size is 256MB.

Note: The board must be in HALT state before calling this function.

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example reads 64 steps from the Master board expected memory beginning at step 128:

```
DWORD adwMemory[64];
DioRealTimeCompareReadMaskMemory (nHandle, adwMemory, 128, 64, &nStatus);
```

See Also

DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareReadResults

Applies To

GX5290, GX5295, File

Purpose

Read a block of data from the onboard comparison result memory.

Syntax

DioRealTimeCompareReadResults (*nHandle, pdwAddress, pdwData, pbComparedStatus, dwStart, pdwSize, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
pdwAddress	PDWORD	Pointer to an array of type DWORD, array size has to at minimum equal to the number of steps to read.
pdwData	PDWORD	Pointer to an array of type DWORD, array size has to at minimum equal to the number of steps to read.
pbComparedStatus	PBOOL	Pointer to an array of type BOOL, array size has to at minimum equal to the number of steps to read. This array contains the comparison result for the specified Program counter and data. A value of TRUE
dwStart	DWORD	Result array index to start reading from.
pdwSize	PDWORD	Number of items to read, the maximum number of comparison results is 1024. All of the three arrays need to be of the same size and at least equal to the *pdwSize*. The returned data will be the lower of the two: if the specified *pdwSize* is less than the actual number of recorded comparisons or if the actual number of recorded comparisons is less then *pdwSize*. Upon return *pdwSize* will be updated with the actual number of data points.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

The onboard comparison result memory is cleared each time the board is triggered.

Example

The following example reads 64 data points from the onboard comparison result memory beginning at 128:

```
DWORD   dwAddress [64], dwData[64], dwSize=64;
BOOL    bComparedStatus[64];
DioRealTimeCompareReadResults (nHandle, dwdAddress, dwData, bComparedStatus, 128, &dwSize,
&nStatus);
```

See Also

DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType,

DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareSetupActiveChannels

Applies To

GX5295, File

Purpose

Returns the specified board active channels when in real time compare per channel fail mode.

Syntax

DioRealTimeCompareSetupActiveChannels (*nHandle, dwActiveChannels, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
dwActiveChannels	DWORD	Sets the specified board active channels when in real time compare per channel fail mode. Each bit represents a channel, channel 0 = bit 0, channel 31 = bit 31 etc. Bit Low (0): The respected channel will no be compared. Bit High (1): The respected channel will no be compared.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

All boards in the domain have to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

All boards in the domain have to set the Real Time Compare mode to DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL mode using **DioRealTimeCompareSetupMode** API prior calling this function. See the **DioRealTimeCompareSetupMode** function for more details.

Note: Real Time Compare functionality is supported only by GX529x boards with firmware versions 0x8A00 and above.

Note: Real Time Compare with per channel fail functionality is supported only by GX5295 boards with firmware versions 0xF605 and above.

Example

```
The following example initializes a master board in slot 6 and a slave board in slot 7. Sets both
boards to operate in Real Time Compare mode. Sets the domain to be in Real Time Compare mode, Set
the Real Time Compare mode to DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL mode, and enables all even
channels in the master board. And reads the board active channels real time compare pass fail
statues


SHORT nMasterHandle1, nSlave1, nDensity, nBanks, nStatus;
DWORD dwChannelsStatus, dwActiveChannels;


DioSetupInitialization (0, 1, 0x106, &nDensity, &nBanks, &nMasterHandle1, &nStatus);
DioSetupInitialization (nMasterHandle1, 1, 0x107, &nDensity, &nBanks, &nSlave11, &nStatus);
DioDomainSetupOperatingMode (nMasterHandle, DIO_OPERATING_MODE_REAL_TIME_COMPARE, &nStatus);
DioRealTimeCompareSetupMode (nMasterHandle, DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL, &nStatus);
DioRealTimeCompareClearResultMemory (nMasterHandle, DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL,
&nStatus);
DioRealTimeCompareSetupActiveChannels(nMasterHandle, 0xAAAAAAAA, &nStatus);
DioRealTimeCompareGetActiveChannels(nMasterHandle, &dwActiveChannels, &nStatus);
DioRealTimeCompareReadChannelsStatus (nMasterHandle, &dwChannelsStatus, pnStatus)
```

See Also

DioRealTimeCompareClearChannelsStatus, DioRealTimeCompareReadChannelsStatus, DioRealTimeCompareSetupMode, DioRealTimeCompareReadChannelsStatus, DioRealTimeCompareSetupActiveChannels, DioRealTimeCompareSetupMode, DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareSetupConditionValue

Applies To

GX5290, GX5295, File

Purpose

Sets the specified condition value.

Syntax

DioRealTimeCompareSetupConditionValue (*nHandle, nCondition, dwValue, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
nCondition	SHORT	Condition: 0. DIO_RTC_CONDITION_FAILURE_COUNT: the DIO domain will PAUSE/HALT once the number of detected errors equals the condition count value. 1. DIO_RTC_CONDITION_DATA: the DIO domain will PAUSE/HALT once the vector data matches the condition data value. 2. DIO_RTC_CONDITION_PROGRAM_COUNTER: the DIO domain will PAUSE/HALT once the vector step number matches the condition address value.
dwValue	DWORD	Condition value. If *nCondition* is set to DIO_RTC_CONDITION_FAILURE_COUNT then the condition value can be 1 to 1024. If *nCondition* is set to DIO_RTC_CONDITION_DATA then the condition value can be between 0 to 0xFFFFFFFF. If *nCondition* is set to DIO_RTC_CONDITION_PROGRAM_COUNTER then the condition value can be between 0 to last step of 67108863, or higher if number of channels was set to less than 32.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each DIO in the domain can have its own active condition to stop on as it was set by calling **DioRealTimeCompareSetupStopMode** function.

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example sets the condition count value to 10:

```
SHORT   nStatus;
DioRealTimeCompareSetupConditionValue (nHandle, DIO_RTC_CONDITION_FAILURE_COUNT, 10, pnStatus);
```

See Also

DioRealTimeCompareGetConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareSetupDataDelay

Applies To

GX5290, GX5295, File

Purpose

Sets the specified data source clock cycles delay.

Syntax

DioRealTimeCompareSetupDataDelay (*nHandle, nDataType, dwCycles, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
nDataType	SHORT	Data type: 0. DIO_RTC_EXPECTED_DATA: sets the number of clock cycles that the expected memory data will be delayed by from the strobe. 1. DIO_RTC_MASKED_DATA: sets the number of clock cycles that the mask memory data will be delayed by from the clock.
dwCycles	DWORD	Data clock cycles delay value. Value can be between 0-15
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function sets the number of clocks cycles that the expected and masked memory data will be delayed by. Since each data in the comparison process has its own delay the function can be used to align the three different data sources, input data, masked data and expected data to be correctly compared.

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example sets the expected memory data source clock cycles delay to 2:

```
DioRealTimeCompareSetupDataDelay (nHandle, DIO_RTC_EXPECTED_DATA, 2, &nStatus);
```

See Also

DioRealTimeCompareSetupConditionValue, DioRealTimeCompareGetDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareSetupInputDataClockEdge

Applies To

GX5290, GX5295, File

Purpose

Sets the input data strobe edge.

Syntax

DioRealTimeCompareSetupInputDataClockEdge *(nHandle, nEdge, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
nEdge	SHORT	Input data clock edge: 0. DIO_RISING_EDGE: Input data clock rising edge 1. DIO_FALLING_EDGE: Input data clock falling edge
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The response data which is clocked by the strobe is fed to the real time comparator using the same clock as the output data and the mask data. The user can set the input data (response data) clock edge to either capture the input data on the falling or rising edge. This can assist in aligning the input data with the other two sources of data, the masked and expected memory data to correctly compare.

The Rising/Falling parameter defines the edge of the data clock that is used to transfer the data sampled by the strobe into the input pipeline. This parameter is used to avoid meta stable conditions where the sample strobe and the data clock might overlap, causing ambiguous data storage. Using the falling edge of the data clock to transfer sampled data to the input pipeline places the data in the pipeline one cycle earlier than using the rising edge of the data clock (it uses the falling edge of the previous vector clock cycle not the rising edge of the current vector clock cycle).

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example sets the input data clock edge:

```
DioRealTimeCompareSetupInputDataClockEdge (nHandle, DIO_RISING_EDGE, &nStatus);
```

See Also

DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareGetInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareSetupMode

Applies To

GX5295, File

Purpose

Sets the Real Time Compare mode.

Syntax

DioRealTimeCompareSetupMode (*nHandle, nMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
nMode	SHORT	Sets the Real Time Compare mode, modes cane be as follows: 2 DIO_RTC_MODE_DEFAULT: Default Real Time Compare mode operating mode. Supported by supported only by GX529x boards with firmware versions 0x8A00 and above. 3 DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL: In this mode all default Real Time Compare operating mode are supported, but in addition when running vectors only the first failure per channels will be logged. Supported by supported only by GX5295 boards with firmware versions 0xF605 and above.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL Real Time Compare mode is useful for a go-no-go testing where the user wants to know which channels failed at least once and the failure vector address.

NOTE: All boards in the domain have to set the Real Time Compare mode to DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL mode using **DioRealTimeCompareSetupMode** API prior calling this function. See the **DioRealTimeCompareSetupMode** function for more details.

Example

```
The following example initializes a master board in slot 6 and a slave board in slot 7. Sets both
boards to operate in Real Time Compare mode. Sets the domain to be in Real Time Compare mode, Set
the Real Time Compare mode to DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL mode, and enables all even
channels in the master board. And reads the board active channels real time compare pass fail
statues


SHORT nMasterHandle1, nSlave1, nDensity, nBanks, nStatus;
DWORD dwChannelsStatus, dwActiveChannels;


DioSetupInitialization (0, 1, 0x106, &nDensity, &nBanks, &nMasterHandle1, &nStatus);
DioSetupInitialization (nMasterHandle1, 1, 0x107, &nDensity, &nBanks, &nSlave11, &nStatus);
DioDomainSetupOperatingMode (nMasterHandle, DIO_OPERATING_MODE_REAL_TIME_COMPARE, &nStatus);
DioRealTimeCompareSetupMode (nMasterHandle, DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL, &nStatus);
DioRealTimeCompareClearResultMemory (nMasterHandle, DIO_RTC_MODE_FIRST_FAIL_PER_CHANNEL,
&nStatus);
DioRealTimeCompareSetupActiveChannels(nMasterHandle, 0xAAAAAAAA, &nStatus);
DioRealTimeCompareGetActiveChannels(nMasterHandle, &dwActiveChannels, &nStatus);
DioRealTimeCompareReadChannelsStatus (nMasterHandle, &dwChannelsStatus, pnStatus)
```

See Also

DioRealTimeCompareClearChannelsStatus, DioRealTimeCompareReadChannelsStatus, DioRealTimeCompareSetupActiveChannels, DioRealTimeCompareReadChannelsStatus, DioRealTimeCompareSetupActiveChannels, DioRealTimeCompareSetupMode, DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareSetupResultsDataType

Applies To

GX5290, GX5295, File

Purpose

Sets the compared results type.

Syntax

DioRealTimeCompareSetupResultsDataType *(nHandle, nDataType, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
nDataType	SHORT	Compared results type: 0. DIO_RTC_SAVE_COMPARED_DATA: saved data is the result of the incoming data masked by the mask memory and compared with the expected memory (default). 1. DIO_RTC_SAVE_INPUT_DATA: saved data is the raw input data that failed comparison.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The maximum member of compared data points is 1K. See the **DioRealTimeCompareReadResults** function for more details.

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example sets the compared results type:

```
DioRealTimeCompareSetupResultsDataType (nHandle, DIO_RTC_SAVE_COMPARED_DATA, &nStatus);
```

See Also

DioRealTimeCompareGetResultsDataType, DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareSetupStopCondition

Applies To

GX5290, GX5295, File

Purpose

Sets the comparison stop condition.

Syntax

DioRealTimeCompareSetupStopCondition (*nHandle, nCondition, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
nCondition	SHORT	Comparison stop mode. 0. DIO_RTC_STOP_ON_FAILURES_COUNT: In this stop condition the board stops when the total number of failures is equal to the number of failures count. If the number of detected failures is less the condition value the board will run to the end of the vector. 1. DIO_RTC_STOP_ON_FAILURES_COUNT_SAVE_DATA: In this stop condition the board continuously comparing and recording the input data. The trigger for the board to stop is when the detected failures number is equal to the condition failures count number. At that point the DIO will continue to record and compare data until there is no more room in the results memory. In this condition the result memory is operating like a FIFO until stop condition is met. 2. DIO_RTC_STOP_ON_DATA_VALUE: In this stop condition the board continuously comparing and recording the input data. The trigger for the board to stop is when the input data is equal to the condition data value At that point the DIO will continue to record and compare data until there is no more room in the results memory. In this condition the result memory is operating like a FIFO until stop condition is met. 4. DIO_RTC_STOP_ON_PROGRAM_COUNTER: In this stop condition the board continuously comparing and record the input data. The trigger for the board to stop is when the program counter value is equal to the condition data value At that point the DIO will continue to record and compare data until there is no more room in the results memory. In this condition the result memory is operating like a FIFO until stop condition is met.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each boar in the DIO domain can have its own stop condition. As a result the first board that its stop condition is met will stop all other boards in the domain even if their stop condition is not met yet.

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example sets the comparison stop condition:

```
DioRealTimeCompareSetupStopCondition (nHandle, DIO_RTC_STOP_ON_FAILURES_COUNT, &nStatus);
```

See Also

DioRealTimeCompareGetStopCondition, DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareWriteExpectedMemory

Applies To

GX5290, GX5295, File

Purpose

Writes a block of expected memory to the specified board (Master or Slave).

Syntax

DioRealTimeCompareReadExpectedMemory (*nHandle, pvVector, dwStart, dwSize, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO or File board handle.
pvVector	PVOID	Pointer to an array, array data type needs to comply with the I/O width settings and board type, see comments for details.
dwStart	DWORD	Starting step to read from.
dwSize	DWORD	Number of steps to read.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The program counter will is set to zero upon return.

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291: Maximum array size is 32M of double words. GX5292: Maximum array size is 64M of double words. GX5293: Maximum array size is 64M of double words. GX5295: Maximum array size is 64M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). GX5291: Maximum array size is 64M words. GX5292: Maximum array size is 128M words. GX5293: Maximum array size is 128M words. GX5295: Maximum array size is 128M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). GX5291: Maximum array size is 128MB. GX5292: Maximum array size is 256MB. GX5293: Maximum array size is 256MB. GX5295: Maximum array size is 256MB.
4	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-3). Bits 4-7 of the array are not used and treated as don't care. GX5291: Maximum array size is 256MB GX5292: Maximum array size is 512MB

GX5293: Maximum array size is 512MB

GX5295: Maximum array size is 512MB

2 Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-1). Bits 2-7 of the array are not used and treated as don't' care.

GX5291: Maximum array size is 512MB

GX5292: Maximum array size is 1GB

GX5293: Maximum array size is 1GB

GX5295: Maximum array size is 1GB

1 Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channel 0). . Bits 1-7 of the array are not used and treated as don't' care.

GX5291: Maximum array size is 1GB

GX5292: Maximum array size is 2GB

GX5293: Maximum array size is 2GB

GX5295: Maximum array size is 2GB

Note: The board must be in HALT state before calling this function.

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example writes 64 steps from the Master board expected memory beginning at step 128:

```
DWORD adwMemory[64];
DioRealTimeCompareWriteExpectedMemory (nHandle, adwMemory, 128, 64, &nStatus);
```

See Also

DioRealTimeCompareReadExpectedMemory, DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteMaskMemory, DioRealTimeCompareReadResults

DioRealTimeCompareWriteMaskMemory

Applies To

GX5290, GX5295, File

Purpose

Writes a block of mask memory to the specified board (Master or Slave).

Syntax

DioRealTimeCompareWriteMaskMemory (*nHandle, pvMemory, dwStart, dwSize, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO or File board handle.
pvMemory	PVOID	Pointer to an array, array data type needs to comply with the I/O width settings and board type, see comments for details.
dwStart	DWORD	Starting step to read from.
dwSize	DWORD	Number of steps to read.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The program counter will is set to zero upon return.

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291: Maximum array size is 32M of double words. GX5292: Maximum array size is 64M of double words. GX5293: Maximum array size is 64M of double words. GX5295: Maximum array size is 64M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). GX5291: Maximum array size is 64M words. GX5292: Maximum array size is 128M words. GX5293: Maximum array size is 128M words. GX5295: Maximum array size is 128M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). GX5291: Maximum array size is 128MB. GX5292: Maximum array size is 256MB. GX5293: Maximum array size is 256MB. GX5295: Maximum array size is 256MB.
4	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-3). Bits 4-7 of the array are not used and treated as don't care. GX5291: Maximum array size is 256MB GX5292: Maximum array size is 512MB

GX5293: Maximum array size is 512MB

GX5295: Maximum array size is 512MB

2 Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-1). Bits 2-7 of the array are not used and treated as don't' care.

GX5291: Maximum array size is 512MB

GX5292: Maximum array size is 1GB

GX5293: Maximum array size is 1GB

GX5295: Maximum array size is 1GB

1 Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channel 0). . Bits 1-7 of the array are not used and treated as don't' care.

GX5291: Maximum array size is 1GB

GX5292: Maximum array size is 2GB

GX5293: Maximum array size is 2GB

GX5295: Maximum array size is 2GB

Note: The board must be in HALT state before calling this function.

The board has to be in Real Time Compare operating mode prior calling this function. See the **DioDomainSetupOperatingMode** function for more details.

Note: the Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example writes 64 steps from the Master board expected memory beginning at step 128:

```
DWORD adwMemory[64];
DioRealTimeCompareWriteMaskMemory (nHandle, adwMemory , 128, 64, &nStatus);
```

See Also

DioRealTimeCompareReadResults, DioRealTimeCompareSetupConditionValue, DioRealTimeCompareSetupDataDelay, DioRealTimeCompareSetupInputDataClockEdge, DioRealTimeCompareSetupResultsDataType, DioRealTimeCompareSetupStopCondition, DioRealTimeCompareWriteExpectedMemory, DioRealTimeCompareWriteMaskMemory

DioRemapChannel

Applies To

GX5280, GX5290, GX5295, File

Purpose

Remap the specified channel's data in a DIO domain.

Syntax

DioRemapChannel (*nHandle, nFirstChannel, nSecondChannel, nMode, dwMemory, dwStartStep, pdwNumSteps, dwOptions, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO or File board handle.
nFirstChannel	SHORT	Specified the first channel in the DIO domain or DIO File to remap.
nSecondChannel	SHORT	Specified the second channel in the DIO domain or DIO File to remap.
nMode	SHORT	Remap operation mode: 1. DIO_SWAP_CHANNELS: swap the data of the first and second channels. 2. DIO_MOVE_CHANNELS: move the first channel to the second channel location while pushing the channels in between up or down. E.g. passing first channel=3 and second channel=0, then the channels contents will be: 3,0,1,2. 3. DIO_COPY_CHANNELS: copy the first channel's data to the second channel.
dwMemory	DWORD	Selects the target memories to be affected by the remap operation, if selecting memory target other then 0 (Auto mode), memories can be ORred. See comments for details: 0=DIO_REMAP_MEMORY_AUTO: will affect all required memories based on the remap operation and the board's type. E.g. if the board type is Gx529x, then selecting the copy channels operation it will copy the output memory as well as the direction memory to the second channel. 1=DIO_REMAP_IN_MEMORY: Input memory apples to Gx529x board type only. 2=DIO_REMAP_OUT_MEMORY: Output memory apples to Gx529x board type only. 4=DIO_REMAP_DATA_MEMORY: Data memory apples to Gx528x board type only. 8=DIO_REMAP_CTRL_MEMORY: Control memory apples to Gx5055 board type only. 0x10=DIO_REMAP_DIRECTION_MEMORY: Direction memory apples to Gx529x board type only. 0x20=DIO_REMAP_RTC_EXPECTED_MEMORY: Expected memory apples to Gx529x board type only when in Real time compare operation mode. 0x40=DIO_REMAP_RTC_INPUT_MASK_MEMORY: Mask memory apples to Gx529x board type only when in Real time compare operation mode.

dwStartStep	DWORD	Specified starting step in the DIO board or File.
pdwNumSteps	PDWORD	Specified the number of steps, passing -1 selects all the steps in the DIO board or File from start step to the end.
dwOptions	DWORD	Currently not used, pass as 0.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Note: The board must be in HALT state before calling this function.

Note: Real Time Compare functionality is supported by GX529x boards with firmware versions 0x8A00 and above.

Example

The following example copies 124 steps of channel 0 to channel 3 using auto mode for memory selection:

```
DWORD dwNumSteps;
DioRemapChannel (nHandle, 0 , 3, DIO_COPY_CHANNELS, DIO_REMAP_MEMORY_AUTO 0, 124, 0, &nStatus);
```

See Also

DioReadCtrlMemory, DioWriteCtrlMemory, DioReadDirectionMemory, DioWriteDirectionMemory, DioReadInMemory, DioWriteInMemory, DioReadIOMemory, DioWriteIOMemory, DioReadOutMemory, DioWriteOutMemory, DioGetErrorString

DioReset

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Changes the state of the specified Master board to HALT and restores Master and Slaves to the default setup.

Syntax

DioReset *(nMasterHandle, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Board handle.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The following is the default setup:

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295:

- Board is in a HALT state.
- Program counter is set to 0.
- Clock and Strobe are internal.
- Default Clock:=10MHz.
- Strobe timing = 10 nSec.
- Trigger mode is disabled.
- D, T and P mask register values are 0xFFFF.
- D, T and P event registers are 0x0000.
- External event source is external event lines (0).
- X register value is 0.

GX5150:

- I/O pins direction is set to Output.
- I/O memory width is 32-bit.
- External trigger transient level is set to low level.
- Jump register A trigger mode is edge trigger.
- Halt output mode is Hold Last.
- Sequencer Overflow mode set to enable.
- Clk Delay and Out Clock Delay are set to 12 nSec.
- Pause Counter Mode is disabled.
- StepCounter and Pause Count are zero.

GX5280/GX5290/GX5290e/GX5295:

- Clk Delay set to 5 nSec
- Strobe Delay set to 10 nSec.
- I/O pins direction is set to Input.
- External trigger transient level is set to low level.
- Halt output mode is Hold Last.

Example

The following example resets the board:

```
SHORT   nStatus;
DioReset(nHandle, &nStatus);
```

See Also

DioHalt, DioInitialize, DioGetErrorString

DioResetChannels

Applies To

GX5295

Purpose

Sets the specified channels range to default settings without resetting the whole board.

Syntax

DioResetChannels (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel*, *panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel*, *panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The following are the default settings:

- Channel mode set to Output enabled.
- High output voltage set to 2.5V
- Low output voltage set to 0.0V
- High input threshold voltage set to 2.5V
- Low input threshold voltage set to 0.0V
- Sink input load current set to 8.0Ma
- Source input load current set to 8.0mA
- Input Load Commutating Voltage set to 2.5V
- Pmu Forced Current set to 0Ma.
- Pmu Forced Current range set to -32mA to +32mA.
- Pmu Forced Current Commutating high Voltage set to 2.5V.
- Pmu Forced Current Commutating low Voltage set to 0V.
- Pmu Forced Voltage set to 0V.

Example

The following example uses an array of channels list to be reset:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioResetChannels (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0,
DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT, &nStatus);
```

The following example reset all the board's channels:

```
DioResetChannels (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0,
DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT, &nStatus);
```

The following example resets all boards' channels:

```
DioResetChannels (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0,
DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT, &nStatus);
```

The following example resets channels 5 to 10

```
DioResetChannels (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10,
DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT, &nStatus);
```

See Also

DioSetupOutputVoltages, DioSetupInputThresholdVoltages, DioPmuSetupForcedCurrent, DioPmuSetupForcedCurrentCommutatingVoltage, DioPmuSetupForcedVoltage, DioReset, DioGetErrorString

DioResetOutputOverCurrentStates

Applies To

GX5055, File.

Purpose

Resets all 32 channels over current high and low flags states.

Syntax

DioResetOutputOverCurrentStates (*nHandle, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

NOTE over current condition can only be cleared by calling **DioResetOutputOverCurrentStates**. After the over current is cleared the channel output driver will be active again.

Channels over current high and low flags states can be reset dynamically at any time even while the DIO is running mode.

Example

The following example resets all 32 channels over current high and low flags states:

```
DioResetOutputOverCurrentStates (nHandle, &nStatus);
```

See Also

DioSetupOutputDataFormat, DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioSaveDomainConfiguration

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Save the domain configuration associated the specified master handle to the GtDio.ini file.

Syntax

DioSaveDomainConfiguration *(nMasterHandle, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Save the domain configuration associated the specified master handle to the GtDio.ini file to be used by **DioInitialize** API.

The **DioInitialize** can be used to initialize the DIO system configuration (Master and Slave boards) with one command. The required parameters (base address, density, banks etc.) are taken from the last configuration set by one of the following:

- A previous call to DioSetupInitialization.
- DioPanel configure push button.
- GTDIO.INI file.

The function calls **DioSetupInitialization** for the Master and its Slaves.

Example

The following example demonstrates how to initialize a system that a Master board in chassis 1 slot 6 and a slave at chassis 1 slot 7 and save the configuration:

```
SHORT nMasterHandle1, nSlave1, nDensity, nBanks, nStatus;

DioSetupInitialization (0, 1, 0x106, &nDensity, &nBanks, &nMasterHandle1, &nStatus);
DioSetupInitialization (nMasterHandle1, 1, 0x107, &nDensity, &nBanks, &nSlave11, &nStatus);
DioSaveDomainConfiguration (nMasterHandle, &nStatus)
```

See Also

DioInitialize, DioSetupInitialization, DioGetSlaveHandle

DioSaveFile

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Saves steps to a DIO file from the specified Master board and its Slaves.

Syntax

DioSaveFile *(nMasterHandle, szFileName, dwStart, pdwSize, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
szFileName	LPSTR	The destination file name for the results. The file format used is the **DIOEasy3.0** default file format.
dwStart	DWORD	Starting step.
pdwSize	PDWORD	Number of steps to load. When 0 is used, all the memory steps are saved. Upon return, the variable holds the number of steps actually saved to the file. In GX5150/the current step width is used.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function saves data starting at step *dwStart* until *dwSize* steps are saved in file ***szFileName.***

The board(s) must be in the HALT state before calling this function, otherwise the function returns an error.

The default ***DIOEasy3.0*** file format is used by this function. Once the file is saved it can no longer be read by ***DIOEasy1.x,*** although version ***1.x*** files can be loaded. After the function is called, the program counter is set to zero.

The function also saves the control memory for each step, as well as the current board setting. See Appendix B for a complete listing of the saved settings.

Example

The following example initializes the Master board and its Slaves, loads a DIO file, arms, triggers, and waits until the board state is HALT state, then saves the results to a file:

```
DWORD dwSize=0;
DioInitialize (0, &nMasterHandle, &nStatus);
DioLoadFileVector (nMasterHandle, "busc.dio", 0, 0, &dwSize, &nStatus);
DioArm (nMasterHandle, &nStatus);
DioTrig (nMasterHandle, &nStatus);
Do
    {  DioReadStatusRegister (nMasterHandle, &wData, &nStatus);
    }while (wData &0x1C ==0);
    // wait to HALT state
DioSaveFile (nMasterHandle, "busxout.di", 0, &dwSize, &nStatus);
```

See Also

DioLoadFile, DioCompareFiles, DioGetErrorString

DioSelfTest

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Tests all the board memory banks and registers. If the test failed, then the first memory failed step number is returned.

Syntax

DioSelfTest (*nHandle, nTestMode, pdwFailedStepNum, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
nTestMode	SHORT	Test modes are as follow: 0 TEST_MEMORY: Test Memory. 1 TEST_REGISTERS: Test 2 TEST_ALL: Test both memory and registers. See Comments for details.
pdwFailedStepNum	PDWORD	The first failed step number. If no errors were found, the returned value would be set to 0xFFFFFFFF.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function tests the registers and/or the memory.

In case the memory is tested then all the physical memory will be tested. The memory data will not be changed upon completing the test successfully.

In case the registers are tested then all the registers will be tested.

The board must be in HALT mode.

Example

In the following example, the board is set to the HALT state and then DioSelfTest is called with the mode set to complete test:

```
DWORD dwFailedStepNum;
SHORT nStatus;
DioHALT (nHandle, &nStatus);
DioSelfTest (nHandle, TEST_ALL, &dwFailedStepNum, &nStatus);
```

See Also

DioGetErrorString

DioSetJumpAddress

Applies To

GX5150, File

Purpose

Sets the specified register jump address.

Syntax

DioSetJumpAddress *(nMasterHandle, nRegister, dwStep, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
nRegister	SHORT	Register values are: 0 Register A. 1 Register B.
dwStep	DWORD	Step number jump target when the trigger condition is met.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The jump address can be any step in the physical memory range.

The board must be in either HALT or PAUSE mode.

Example

The following example sets the register A jump address to 123:

```
SHORT  nStatus;
DioGetJumpAddress(nMasterHandle, 0, 123, &nStatus);
```

See Also

DioGetJumpAddress, DioGetErrorString

DioSetOperationMode

Applies To

GC5050, GX5050.

Purpose

Sets the current operation mode.

Syntax

DioSetOperationMode *(nHandle, nMode, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or Slave board handle.
nMode	SHORT	Operation mode are: OP_MODE_NORMAL: 0x0 (default) OP_MODE_FAST_ARM: 0x10 OP_MODE_FAST_TRIG: 0x20 OP_MODE_FAST_HALT: 0x40
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function accelerates the execution of ARM and/or TRIG commands when the running in frequencies less then 200KHz. The command is executed immediately with no delays or testing.

Users can accelerate the execution time of the ARM,.TRIG and HALT commands by setting (or combing) the operation modes.

Operation modes can be combined using Boolean OR operation.

Example

The following example sets the GC5050 operation mode for fast ARM and TRIG and then returned the operation mode:

```
SHORT  nStatus, nOutputState;
DioSetOperationMode (nHandle, OP_MODE_FAST_ARM | OP_MODE_FAST_TRIG, &nStatus);
DioGetOperationMode (nHandle, &nOutputState, &nStatus);
```

See Also

DioGetOperationMode, DioGetErrorString

DioSetPauseCount

Applies To

GX5150, File.

Purpose

Sets the Pause count value.

Syntax

DioGetPauseCount(*nMasterHandle, dwCount, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
dwCount	DWORD	Count value, can be 0 to 4,294,967,295 (0 to 0xFFFFFFFF).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board will pause after executing the specified number of steps if the pause on step mode was previously enabled by calling the “DioSetPauseCounterMode” function.

The board must be in a HALT or PAUSE mode.

Example

The following example sets the PAUSE count to 1024:

```
SHORT  nStatus;
DioSetPauseCount (nMasterHandle, 1024, &nStatus);
```

See Also

DioGetPauseCount, DioSetPauseCounterMode, DioGetPauseCounterMode, DioGetStepCounter, DioGetErrorString

DioSetPauseCounterMode

Applies To

GX5150, File.

Purpose

Sets the Pause counter mode.

Syntax

DioGetPauseCounterMode (*nMasterHandle, nMode, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
nMode	SHORT	Pause counter modes are: 0. Disabled. 1. Enabled. 2. Zero on Pause.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in a HALT or PAUSE mode.

When in "Zero on Pause" mode, the Pause counter will be set to zero each time the board is paused by the Pause counter.

Example

The following example enables the Pause counter mode:

```
SHORT   nStatus, nMode;
DioSetPauseCounterMode (nHandle, 1, &nStatus);
```

See Also

DioGetPauseCounterMode, DioGetPauseCount, DioSetPauseCount, DioGetStepCounter, DioGetErrorString

DioSetPowerConnect

Applies To

GX5055

Purpose

Sets the front-end board power connect relay state

Syntax

DioSetPowerConnect (*nHandle, nState, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Returned Handle (session identifier) that can be used to call any other operations of that resource
nState	SHORT	Sets the front-end power relay state. 0. DIO_POWER_DISCONNECT: Disconnect power 1. DIO_POWER_FROM_FRONT: Connect to front J7 Power Connector 2. DIO_POWER_FROM_BACKPLANE: Connect to the backplane for power
pnStatus	PSHORT	Returned status: 0 on success, negative value on failure.

Comments

The pin electronics require the rail voltages, (high rail voltage (VCC) and low rail voltage (VEE) to be present in order to operate correctly. These rails are connected by way of a relay switch to either the J7 front panel connector or to the backplane when using a special high powered chassis with the J5 connecter present. When setting the connection state to DIO_POWER_FROM_FRONT or DIO_POWER_FROM_BACKPLANE, the Vmid will be calculated based on what rail voltages are detected. The rail voltages should be present before connecting the power so the Vmid can be calculated correctly.

Prior to connecting the power, the external power supply needs to be turned on with 0V to avoid over-current condition.

Example

The following example connects the power to the front panel J7 Connector:

```
SHORT nStatus;

DioSetPowerConnect(nHandle, DIO_POWER_FROM_FRONT, &nStatus);
```

See Also

DioGetPowerConnect, DioGetErrorString

DioSetupAuxiliaryToTimingInput

Applies To

GX5295, File.

Purpose

Sets the specified Auxiliary channel connections to the timing inputs signals.

Syntax

DioSetupAuxiliaryToTimingInput (*nHandle, nAuxChannel, dwInputComparator, dwInputSignal, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nAuxChannel	SHORT	Specified Auxiliary Pin Electronics channel in the DIO board, channel can be as follows: 1000 DIO_AUX_CHANNEL_0: Auxiliary Pin Electronics channel 0. 1001 DIO_AUX_CHANNEL_1: Auxiliary Pin Electronics channel 1. 1002 DIO_AUX_CHANNEL_2: Auxiliary Pin Electronics channel 2. 1003 DIO_AUX_CHANNEL_3: Auxiliary Pin Electronics channel 3.
dwInputComparator	DWORD	The Auxiliary channel inputs signal can be programmatically connected to either the low voltage level comparator or the high voltage level comparator. Each bit represents one of the possible inputs as defined in *dwInputSignal*. See Comments for details.
dwInputSignal	DWORD	The Auxiliary channel can be connected to any of the following signals (see comments for details): • DIO_AUX_TO_EXTERNAL_INPUT_DISABLED = 0x0 • DIO_AUX_TO_EXTERNAL_CLOCK = 0x1 • DIO_AUX_TO_EXTERNAL_STROBE = 0x2 • DIO_AUX_TO_EXTERNAL_TRIGGER = 0x4 • DIO_AUX_TO_EXTERNAL_PAUSE = 0x8 • DIO_AUX_TO_EXTERNAL_CLOCK_ENABLE = 0x10 • DIO_AUX_TO_EXTERNAL_STROBE_ENABLE = 0x20 • DIO_AUX_TO_EXTERNAL_EVENT_LINE_0 = 0x100 • DIO_AUX_TO_EXTERNAL_EVENT_LINE_1 = 0x200 • DIO_AUX_TO_EXTERNAL_EVENT_LINE_2 = 0x400 • DIO_AUX_TO_EXTERNAL_EVENT_LINE_3 = 0x800
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The Auxiliary channel can be connected to configure to any of the following timing signals:

- DIO_AUX_TO_EXTERNAL_INPUT_DISABLED = 0x0: Disconnect all external input signals to the specified Auxiliary channel, default state after calling **DioReset** API.
- DIO_AUX_TO_EXTERNAL_CLOCK = 0x1: Auxiliary channel is connected to External Clock Input (reference to the internal clock PLL or sample clock source) pin number 31 on the Timing connector (J3). If *dwInputComparator*=0x1 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_STROBE = 0x2: Auxiliary channel is connected to External Strobe Input pin number 32 on the Timing connector (J3). If *dwInputComparator*=0x2 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_TRIGGER = 0x4: Auxiliary channel is connected to External Trigger Input (Overrides Run Command) pin number 26 on the Timing connector (J3). If *dwInputComparator*=0x4 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_PAUSE = 0x8: Auxiliary channel is connected to External Pause Input (Overrides Pause Command) pin number 27 on the Timing connector (J3). If *dwInputComparator*=0x8 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_CLOCK_ENABLE = 0x10: Auxiliary channel is connected to External Clock Enable Input (active low) pin number 28 on the Timing connector (J3). If *dwInputComparator*=0x10 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_STROBE_ENABLE = 0x20: Auxiliary channel is connected to External Strobe. If *dwInputComparator*=0x20 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_EVENT_LINE_0 = 0x100: Auxiliary channel is connected to Input External event 0 pin number 1 on the Timing connector (J3). If *dwInputComparator*=0x100 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_EVENT_LINE_1 = 0x200: Auxiliary channel is connected to Input External event 1 pin number 2 on the Timing connector (J3) If *dwInputComparator*=0x200 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_EVENT_LINE_2 = 0x400: Auxiliary channel is connected to Input External event 2 pin number 3 on the Timing connector (J3). If *dwInputComparator*=0x400 high voltage level comparator, otherwise uses the low voltage level comparator.
- DIO_AUX_TO_EXTERNAL_EVENT_LINE_3 = 0x800: Auxiliary channel is connected to Input External event 3 pin number 4 on the Timing connector (J3) If *dwInputComparator*=0x800 high voltage level comparator, otherwise uses the low voltage level comparator.

Example

The following example sets Auxiliary channel 0 connections to external clock the using the high voltage comparator:

```
DioGetAuxiliaryToTimingInput (nHandle, DIO_AUX_CHANNEL_0, 1, DIO_AUX_TO_EXTERNAL_CLOCK,
                              &nStatus);
```

See Also

DioSetupAuxiliaryToTimingInput, DioSetupAuxiliaryToTimingOutput, DioGetErrorString

DioSetupAuxiliaryToTimingOutput

Applies To

GX5295, File.

Purpose

Sets the specified Auxiliary channel connections to the timing output signals.

Syntax

DioSetupAuxiliaryToTimingOutput(*nHandle, nAuxChannel, dwOutputSignal, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nAuxChannel	SHORT	Specified Auxiliary Pin Electronics channel in the DIO board, channel can be as follows: 1000 DIO_AUX_CHANNEL_0: Auxiliary Pin Electronics channel 0. 1001 DIO_AUX_CHANNEL_1: Auxiliary Pin Electronics channel 1. 1002 DIO_AUX_CHANNEL_2: Auxiliary Pin Electronics channel 2. 1003 DIO_AUX_CHANNEL_3: Auxiliary Pin Electronics channel 3.
dwOutputSignal	DWORD	The Auxiliary channel can be connected to configured to any of the following timing signals: • DIO_AUX_TO_EXTERNAL_OUT_DISABLED = 0x0: Disconnect all specified Auxiliary channel to external output signals, default state after calling **DioReset** API. • DIO_AUX_TO_CLOCK_OUT = 0x1: Auxiliary channel is connected to User Output Clock (OClk) pin number 22 on the Timing connector (J3). • DIO_AUX_TO_STROBE_OUT = 0x2: Auxiliary channel is connected to Strobe Output (OStb) pin number 24 on the Timing connector (J3). • DIO_AUX_TO_RUN_OUT = 0x4: Auxiliary channel is connected to Output Run Indication (ORun) pin number 18 on the Timing connector (J3). • DIO_AUX_TO_ARM_OUT = 0x8: Auxiliary channel is connected to Output Arm Indication (OArm) pin number 17 on the Timing connector (J3).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Using the auxiliary channels capabilities of programmed output levels and skew delays the user can better conditioning those signals to meet his UUT requirements.

Example

The following example sets Auxiliary channel 0 connections Clock Out:

```
DioSetupAuxiliaryToTimingOutput (nHandle, DIO_AUX_CHANNEL_0, DIO_AUX_TO_CLOCK_OUT, &nStatus);
```

See Also

DioGetAuxiliaryToTimingInput, DioGetAuxiliaryToTimingOutput, DioGetErrorString

DioSetupBClkFrequency

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Sets up the Master B clock frequency.

Syntax

DioSetupBClkFrequency *(nMasterHandle, dwFrequency, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
dwFrequency	DWORD	B clock frequency value: GC5050/GX5050: 350KHz to 110MHz (10MHz default). GX5150: 1MHz to 120MHz (10MHz default).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The B clock is a general-purpose, external, programmable clock.

GC5050/GX5050: This function uses the internal reference clock to program the B clock frequency. If **DioSetupClkStrobeSource** is set to programmable using the external clock (for example, *nSource* is 2), then the B clock frequency is unpredictable.

GX5150: The B clock uses the internal reference clock.

The board must be in the HALT state or this function returns an error.

Example

The following example sets the Master B clock source frequency to 1MHz:

```
SHORT  nStatus;
DioSetupBClkFrequency (nMasterHandle, 1e6, &nStatus);
```

See Also

DioGetBClkFrequency, DioSetupClkStrobeSource, **DioSetupFrequency, DioGetErrorString**

DioSetupChannelMode

Applies To

GX5295

Purpose

Sets the specified channel operating mode.

Syntax

DioSetupChannelMode (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, nMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nMode	SHORT	Channel operating mode are 0 DIO_CHANNEL_MODE_DYNAMIC_IO: default operating mode, channel is in dynamic I/O mode and output is enabled. 1 DIO_CHANNEL_MODE_DISABLED: channel's output is disabled (Tri-State). 2 DIO_CHANNEL_MODE_OUTPUT_LOW: channel's output is set to low; the output low voltage corresponds to the output low voltage settings. 3 DIO_CHANNEL_MODE_OUTPUT_HIGH: channel's output is set to high; the output high voltage corresponds to the output high voltage settings. 4 DIO_CHANNEL_MODE_PMU_FORCE_CURRENT: channel's output is set to PMU (Parametric Measurement Unit) forced current mode. The channel's forced current is set to zero 5 DIO_CHANNEL_MODE_PMU_FORCE_VOLTAGE: channel's output is set to PMU (Parametric Measurement Unit) forced voltage mode. The channel's forced voltage is set to zero volts.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

For detail regarding additional setting when set to PMU (Parametric Measurement Unit) forced current and forced voltage mode see **DioPmuSetupForcedCurrent** and **DioPmuSetupForcedVoltage** functions..

Example

The following example uses an array of channels list to be set to output low mode:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioSetupChannelMode (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0,
DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT, &nStatus);
```

The following example sets all the board's channels to output low:

```
DioSetupChannelMode (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0,
DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT, &nStatus);
```

The following example sets all the boards channels to output low:

```
DioSetupChannelMode (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0,
DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT, &nStatus);
```

The following example sets channels 5 to 10 to output low:

```
DioSetupChannelMode (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10,
DIO_PMU_COMPARATORS_SOURCE_FORCED_CURRENT, &nStatus);
```

See Also

DioGetChannelMode, DioPmuSetupForcedCurrent, DioPmuSetupForcedVoltage, DioGetErrorString

DioSetupChannelsOutputStates

Applies To

GX5280, GX5290, GX5290e, GX5295, File.

Purpose

Sets the state of each output channel to be enabled or disabled.

Syntax

DioSetupChannelsOutputStates (*nHandle, dwStates, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
dwStates	DWORD	Each of the 32 bits represents channel's output state. • Bit low channel's output disabled • Bit high channel's output enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function enabled or disabled each channel output. Disabled channels in output mode are in Tri-State. This is useful for connecting to a user bus

Example

The following example sets enable channels 0, 5, and 10 outputs and disable all other:

```
SHORT  nStatus;
DioSetupChannelsOutputStates (nMasterHandle, 0x421, &nStatus);
```

See Also

DioGetChannelsOutputStates, DioGetErrorString

DioSetupChannelsVoltageLevel

Applies To

GX5280, GX5290, GX5295, GX5290e, File.

Purpose

Sets all channels voltage level.

Syntax

DioSetupChannelsVoltageLevel (*nHandle, dVoltage, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File Board handle.
dVoltage	DOUBLE	Sets all channels voltage level in the range of 1.4V to 3.6V with resolution of 10mV.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function program the voltage level, applied for all 32 channels, in the TTL Standard connector.

Example

The following example sets all channels voltage level to 2.0V:

```
SHORT  nStatus;
DioSetupChannelsVoltageLevel (nMasterHandle, 2.0, &nStatus);
```

See Also

DioGetChannelsVoltageLevel, DioGetErrorString

DioSetupClkStrobeDelay

Applies To

GX5150, GC5050, GX5050, GX5055, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Returns the Main Clock, Out Clock or Strobe delay value.

Syntax

DioSetupClkStrobeDelay (*nHandle, nClock, dDelay, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	GX5150, GC5050, GX5050, GX5055: Master or File board handle. GX5280, GX5290, GX5290e, GX5295: Board or File board handle.
nClock	SHORT	Clock/Strobe can be as follow: 0 CLK_DELAY: Main Clock. 1 USER_CLK_DELAY: Out Clock 2 STROBE_DELAY: Strobe 3 BOARD_OFFSET_DELAY (Gx528X /Gx529X only): Additional offset that advance or delay the specifed DIO board's Clock and Strobe. The offset values vacan be set between -3.0 nSec to +3.0 nSec with resolution of 0.25nSec. The main purpose of this delaty is to ease timing aligment between DIO boards in a domain.
dDelay	DOUBLE	Delay values are in nSec GX5150, GC5050, GX5050 GX5055 Out Clock and Strobe: 0.0 to 64 nSec with increments of 0.25. GX5280/GX5290/GX5290e/GX5295 Main Clock: and Strobe, any of the following values ranges: 0-3 ns with increments of 0.25 ns. 4-7 ns with increments of 0.25 ns. 8-11 ns with increments of 0.25 ns. 12-15 ns with increments of 0.25 ns. 16-19 ns with increments of 0.25 ns. 20-23 ns with increments of 0.25 ns. 24-27 ns with increments of 0.25 ns. If *nClock* = BOARD_OFFSET_DELAY (Gx528X /Gx529X only): -3.0 to +3.0 ns with increments of 0.25 ns.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function adds a delay between the timing board clock source (i.e. programmable clock) and the following clocks:

0. Main clock that stimulate the DIO.
1. Out Clock signal (timing connector).
2. Strobe Clock signal (timing connector).
3. Board offset delay.

NOTE: For the GX5280/GX5290/GX5290e family when running at frequencies above 100MHz only 0-3nS delay range is available.

Example

The following example sets the main clock delay value to 15 nSec:

```
SHORT  nStatus;
DioSetupClkStrobeDelay (nMasterHandle, 0, 15, &nStatus);
```

See Also

DioGetClkStrobeDelay, DioSetupClkStrobeSource, DioGetClkStrobeSource, DioGetErrorString

DioSetupClkStrobeExternalGateMode

Applies To

GX5280, GX5290, GX5290e, GX5295, File

Purpose

Sets the specified DIO board Clock or Strobe external gate mode.

Syntax

DioSetupClkStrobeExternalGateMode (*nHandle, nClkStrobe, nMode, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	DIO board or File board handle.
nClkStrobe	SHORT	0. DIO_INTERNAL_CLOCK: Internal Clock. 1. DIO_INTERNAL_STROBE: Internal Strobe
nMode	SHORT	External gate mode can be: 0. Disabled 1. Enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function enables full control on the timing of when data is captured or clocked out for each DIO board that resides in the same domain using external input signal.

DIO_INTERNAL_CLOCK:

Enables or disables gating the internal Clock of the specified DIO board. For this mode to be active the direction of all the board's channels must be set to output. When the mode is enabled the External Clock Enable input (J3 pin 28) will gate the clock of the specified DIO board. When the External Clock Enable input is set to low the specified DIO board's internal Clock will be active and the DIO will run, when the External Clock Enable input will be high (default state since the input has IS pulled up to VCC). The DIO will be in Pause state.

DIO_INTERNAL_STROBE:

Enables or disables gating the internal Strobe of the specified DIO board. For this mode to be active the direction of all the board's channels must be set to input. When the mode is enabled the External Strobe Enable input (J3 pin 29) will gate the strobe of the specified DIO board. When the External Strobe Enable input is set to low the specified DIO board's internal Strobe will be active and the DIO will run, when the External Strobe Enable input will be high (default state since the input has IS pulled up to VCC). The DIO will be in Pause state

Example

The following example enables the internal clock external gate mode:

```
SHORT  nStatus;
DioGetClkStrobeExternalGateMode (nHandle, DIO_INTERNAL_CLOCK, 1, &nStatus);
```

See Also

DioGetClkStrobeExternalGateMode, **DioSetupClkStrobeDelay, DioSetupClkStrobeSource, DioGetClkStrobeSource, DioGetErrorString**

DioSetupClkStrobeSource

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Sets the specified Master board's clock and strobe source.

Syntax

DioSetupClkStrobeSource *(nMasterHandle, nSource, pnStatus)*

Parameters

Name	Type	Comments
NMasterHandle	SHORT	Master or File board handle.
nSource	SHORT	Clock and strobe source values are as follow: 0. Internal Clock and Strobe (default). Clock is programmable using an internal reference. Strobe is internal in accordance with strobe timing (default). 1. External Clock, internal Strobe. External clock. Strobe is internal in accordance with strobe timing. This setting enables the External Clock input without the need to pull down the External Clock Enable line. 2. Clock programmed by external, internal Strobe. Clock is Programmable using external clock reference. Strobe is internal in accordance with strobe timing. **Note**: not supported by Gx5055. 3. External Clock and Strobe. Both clock and strobe are external. This setting enables the External Clock and External Strobe inputs without the need to pull down the External Clock Enable and External Strobe Enable lines. 4. Internal Clock, external Strobe. Clock is programmable using an internal reference Strobe is external, setting enables the External Strobe input without the need to pull down the External Strobe Enable line (GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e and GX5295).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The **DioSetupFrequency** function can be used to set the frequency when the selected clock source (*nSource*) is set to 0. **DioSetupClkStrobeDelay** may be called to set the time interval between the strobe and clock signals.

Example

The following example sets the clock to external and strobe to internal:

```
SHORT   nStatus;
DioSetupClkStrobeSource (nMasterHandle, 1, &nStatus);
```

See Also

DioGetClkStrobeSource, DioSetupClkStrobeExternalGateMode, DioGetErrorString

DioSetupControlToPxiTriggerBusLineMode

Applies To

GX5290e, File

Purpose

Sets the specified control signal to PXI Trigger Line and its state.

Syntax

DioSetupControlToPxiTriggerBusLineMode (*nHandle, nControl, nLine, bEnable, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board or File board handle.
nControl	SHORT	PXI Trigger Bus Control line: 0 DIO_TRIGGER_BUS_RUN_CTRL: Run signal to PXI Trigger Bus line. 1 DIO_TRIGGER_BUS_ARM_CTRL: Arm signal to PXI Trigger Bus line. 2 DIO_TRIGGER_BUS_EXT_CLK_CTRL: External Clock signal to PXI Trigger Bus line. 3 DIO_TRIGGER_BUS_EXT_STROBE_CTRL: External Strobe signal to PXI Trigger Bus line
nLine	SHORT	Control to PXI Trigger Bus Line: 0 DIO_PXI_TRIGGER_BUS_LINE0 1 DIO_PXI_TRIGGER_BUS_LINE1 2 DIO_PXI_TRIGGER_BUS_LINE2 3 DIO_PXI_TRIGGER_BUS_LINE3 4 DIO_PXI_TRIGGER_BUS_LINE4 5 DIO_PXI_TRIGGER_BUS_LINE5 6 DIO_PXI_TRIGGER_BUS_LINE6 7 DIO_PXI_TRIGGER_BUS_LINE7
bEnable	BOOL	PXI Trigger Bus Line state: 0 The specified control to PXI Trigger Line is disabled. 1 The specified control to PXI Trigger Line is enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The DIO domain control signals are using the PXI Trigger Bus lines to distribute those signals.

When using internal clock and strobe only two PXI Trigger Bus lines have to be assigned for the Run and Arm control signals.

Additional lines will be required to be assigned for the external clock and/or external strobe.

The assigned lines will be saved to the INI file to be automatically loaded and applied whenever the DIO is initialized.

By default the DIO will use PXI Trigger Bus lines 0 and 1 for the Run and Arm signals.

NOTE: The same PXI Trigger Bus line will be used by all DIO boards (Master and Slaves) in the domain.

Example

The following example assigns the PXI Trigger Bus line 0 to the RUN signal:

```
SHORT   nStatus;


DioSetupControlToPxiTriggerBusLineMode(nMasterHandle, DIO_TRIGGER_BUS_RUN_CTRL,
DIO_PXI_TRIGGER_BUS_LINE0, TRUE, &nStatus);
```

See Also

DioGetControlToPxiTriggerBusLineMode, DioGetErrorString

DioSetupCounterOverflowMode

Applies To

GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Enables or disables the specified Master board's program counter overflow.

Syntax

DioSetupCounterOverflowMode *(nMasterHandle, nOverflowMode, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
nOverflowMode	SHORT	Program counter overflow modes are: 0 Program counter overflow disabled. 1 Program counter overflow enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the overflow mode is enabled then the DIO will not stop at the last step but instead wrap around to step zero and continue. If the overflow mode is disabled, then the sequencer will go to the HALT state at the last step.

The board must be in the HALT or PAUSE state.

Example

The following example enables the specified board's program counter overflow mode:

```
SHORT  nStatus;
DioSetupCounterOverflowMode (nHandle, 1, &nStatus);
```

See Also

DioGetCounterOverflowMode, DioGetErrorString

DioSetupExternalJumpATriggerMode

Applies To

GX5150, File

Purpose

Sets the trigger mode for the external Jump A line.

Syntax

DioSetupExternalJumpATriggerMode *(nMasterHandle, nTriggerMode, pnStatus)*

Parameters

Name	Type	Comments
nMsterHandle	SHORT	Master or File board handle.
nTriggerMode	SHORT	Trigger modes are as follow: 0. Disable external Jump A. 1. Enable external Jump A to trigger when low level is present. 2. Enable external Jump A to trigger when low to high transient occurs.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the selection is "Enable external", then the rising edge of the external signal will create the event. If the selection is low to high transient, then a low level will enable the rising edge internal clock to create the event.

This function applies only for "External Jump A" line located on the I/O Control connector.

The board must be in either the **HALT** or **PAUSE** state.

Example

The following example enables an external Jump A for level sense mode. A constant low level will cause a continuous execution of Jump A. Calls the **DioGetJumpATriggerMode** function to verify the setting.

```
SHORT  nStatus, nMode;
DioSetupExternalJumpATriggerMode (nHandle, 2, &nStatus);
DioGetExternalJumpATriggerMode (nHandle, &nMode, &nStatus);
```

See Also

DioGetExternalJumpATriggerMode, DioGetErrorString

DioSetupExternalPauseAndTriggerMode

Applies To

GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Sets the External PAUSE and TRIG lines mode.

Syntax

DioSetupExternalPauseAndTriggerMode *(nMasterHandle, pnTriggerMode, pnStatus)*

Parameters

Name	Type	Comments
nMsterHandle	SHORT	Master or File board handle.
pnTriggerMode	PSHORT	Trigger modes are as follow: 0. External PAUSE and TRIG lines trigger when low level is present (default). 1. External PAUSE and TRIG lines trigger when low to high transient occur.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the selection is edge, then the rising edge of the external signal will create the event. If the selection is level, then a low level will enable the rising edge internal clock to create the event.

This function applies to the External PAUSE and External TRIG lines located on the Timing connector.

Example

The following example sets the External PAUSE and TRIG lines to trigger when low to high transient occurs, call the **DioGetExternalPauseAndTriggerMode** function to verify the setting.

```
SHORT  nStatus, nMode;
DioSetupExternalPauseAndTriggerMode (nHandle, 1, &nStatus);
DioGetExternalPauseAndTriggerMode (nHandle, &nMode, &nStatus);
```

See Also

DioGetExternalPauseAndTriggerMode, DioGetErrorString

DioSetupExternalRefClkFrequency

Applies To

GX5280, GX5290, GX5290e, GX5295, file

Purpose

Sets the External reference clock frequency.

Syntax

DioSetupExternalRefClkFrequency (*nHandle, nClock, dwFrequency, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Master board handle.
nClock	SHORT	Not used, set to 0.
dwFrequency	DWORD	External reference clock frequency
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The user can provide an external reference clock to the master DIO's on-board PLL. The driver then will use that value when programming the on-board PLL different frequencies.

In order to use that feature the clock source needs to be set to CLK_PRG_EXTERNAL_STROBE_INTERNAL.

The external reference clock range is 7.5MHz to 15MHz.

Example

The following example sets the External reference clock frequency value to 10MHz:

```
SHORT  nStatus;
DioSetupExternalRefClkFrequency (nMasterHandle, 0, 10000000, &nStatus);
```

See Also

DioGetExternalRefClkFrequency, DioSetupFrequency, DioSetupClkStrobeSource, DioGetClkStrobeSource, DioGetErrorString

DioSetupFrequency

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Sets up the clock frequency.

Syntax

DioSetupFrequency *(nMasterHandle, dwFrequency, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
dwFrequency	DWORD	Return frequency range. GC5050: 5Hz to 60MHz. GX5050: 5Hz to 60MHz. GX5055: 5Hz to 50MHz. GX5150: 5Hz to 60MHz. GX5281: 5Hz to 50MHz. GX5282: 5Hz to 100MHz. GX5283: 5Hz to 200MHz. GX5291/GX5291e: 1Hz to 100MHz. GX5292/GX5292e: 1Hz to 100MHz. GX5293/GX5293e: 1Hz to 200MHz. GX5295: 1Hz to 100MHz.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function sets the board clock frequency using the internal reference clock.

The function does not change the current clock source setting.

Note: using driver v4.0 (Build 48) and above frequency can be set regardless of the board's state (PAUSE/HALT/RUN).

GX5150:

After calling **DioReset** or power up, the internal clock is set to 10 MHz.

Calling this function overwrites previous setting made by this function.

GC5050, GX5050, GX5055, GX5280, GX5290, GX5290e, GX5295:

After calling **DioReset** or power up, the internal clock is set to 10 MHz.

GX5293/GX5293e: for frequencies above 100MHz the user must set the number of active I/O channels (wide) to be 16 or less, see the **DioSetupIOConfiguration** function for more details.

GX5283/GX5293/GX5293e: for frequencies above 100MHz all clock and strobe delays have to be between 0 and 3nSec, see **DioSetupClkStrobeDelay** function for more details.

Example

The following example sets the board clock and strobe source to internal, and sets the frequency to 2MHz:

```
SHORT nStatus;
DioSetupClkStrobeSource(nMasterHandle, 0, &nStatus);
DioSetupFrequency (nMasterHandle, 2e6, &nStatus);
```

See Also

DioGetFrequency, DioSetupClkStrobeSource

DioSetupGroupsStaticModeOutputState

Applies To

GX5150

Purpose

Enables or Disables all the groups' outputs mode when in static mode.

Syntax

DioSetupGroupsStaticModeOutputState *(nHandle, nGroupsState, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
nGroupsState	SHORT	Each bit represents Groups' output mode when in Static Mode: Enables or disables the outputs of all 4 I/O channels groups when in static mode, i.e. when reading and writing to the I/O pins statically. Each bit represents a group of 8 channels as follows: 0. Group 0 (channels 0-7): 0- Channels outputs are enabled, 1—Channels outputs are disabled. 1. Group 1 (channels 8-15) 0- Channels outputs are enabled, 1—Channels outputs are disabled. 2. Group 2 (channels 16-23) 0- Channels outputs are enabled, 1—Channels outputs are disabled. 3. Group 3 (channels 24-31) 0- Channels outputs are enabled, 1—Channels outputs are disabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in a HALT or PAUSE state in order for the user to read the I/O lines.

If the memory I/O configuration width was set to 16-bit or 8-bit, then only the 16 or 8 lower bits are applicable and the remaining bits return 0.

The DIO is operating in Static Mode whenever using **DioWriteIOPinsValue or DioReadIOPinsValue** functions.

Example

The following example enables groups 0 and 2 outputs and disables groups 1 and 3 outputs channels:

```
SHORT nStatus;
DioSetupGroupsStaticModeOutputState (nHandle, 0x5, &nStatus);
```

See Also

DioGetGroupsStaticModeOutputState, DioWriteIOPinsValue, DioReadIOPinsValue, DioSetupIOConfiguration, DioGetIOConfiguration, DioGetErrorString

DioSetupInitialization

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Initializes the driver for the specified PXI slot using the HW device driver.

Syntax

DioSetupInitialization (*nMaster, nBrdNum, nBase, pnDensity, pnBanks, nHandle, pnStatus*)

Parameters

Name	Type	Comments
nMaster	SHORT	Specifies the Master number of the Slave board being initialized. If the board is a Master board, then nMaster should be zero and *nBrdNum* will specify the master number.
nBrdNum	SHORT	Specifies the Master or Slave board number. Valid numbers are 1-16 for Master boards and 1-7 for Slave boards (representing I/O pins 32-256 for a domain).
nBase	SHORT	Board slot number.
pnDensity	PSHORT	Memory module size densities
pnBanks	PSHORT	Number of memory modules installed
nHandle	PSHORT	Returns a handle for later reference with this board, 0 for error.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

All PXI/PCI cards:

The Marvin Test Solutions HW device driver is installed with the driver and is the default device driver. The function returns a handle that for use with other Counter functions to program the board. The function does not change any of the board settings.

The specified PXI slot number is displayed by the **PXI/PCI Explorer** applet that can be opened from the Windows **Control Panel**. You may also use the label on the chassis below the PXI slot where the board is installed. The function accepts two types of slot numbers:

- A combination of chassis number (chassis # x 256) with the chassis slot number. For example 0x105 (chassis 1 slot 5).
- Legacy nSlot as used by earlier versions of HW/VISA. The slot number contains no chassis number and can be changed using the **PXI/PCI Explorer** applet (1-255).

DioSetupInitialization must be called for each board (Master and Slave) in order to initialize all boards.

GC5050/GX5050/GX5150:

- Calling this function will not change the current configuration of the board and will set the board to the HALT mode.
- The memory module densities and the number of memory modules installed *are* read from the board.
- Density values are read back from the board and can be as follow:
    0. 256K by 32-bit
    1. 1M by 32-bit
    2. 2M by 32-bit
    3. 4M by 32-bit
- The number of Banks is read back from the board can be 1 to 8.

GX5055:

- Calling this function will not change the current configuration of the board and will set the board to the HALT mode.
- The memory module densities and the number of memory modules installed *are* read from the board.
- Density values are read back from the board and can be as follow:
    0 512 by 32-bit

GX5280/GX5290/GX5290e/GX5295:

- Calling this function will not change the current configuration of the board and will set the board to the HALT mode.
- The memory module densities and the number of memory modules installed *are* read from the board.
- Density values are read back from the board and can be as follow:
    0 32M by 32-bit
    1 64M by 32-bit
    2 128M by 32-bit

Example

The following example demonstrates how to initialize a system that a Master board in chassis 1 slot 6 and a slave at chassis 1 slot 7:

```
SHORT nMasterHandle1, nSlave1, nDensity, nBanks, nStatus;

DioSetupInitialization (0, 1, 0x106, &nDensity, &nBanks, &nMasterHandle1, &nStatus);
DioSetupInitialization (nMasterHandle1, 1, 0x107, &nDensity, &nBanks, &nSlave11, &nStatus);
```

See Also

DioInitialize, DioGetSlaveHandle, DioGetMemoryBankSize, DioGetErrorString

DioSetupInitializationVisa

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Initialize the driver for the specified PXI slot using the default VISA provider.

Syntax

DioSetupInitializationVisa (*nMasterHandle, nBrdNum, szVisaResource, pnDensity, pnBanks, nHandle, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Specifies the Master board's handle of the Slave board being initialized. If the board is a Master board, then nMasterHandle should be zero.
nBrdNum	SHORT	Specifies the Master or Slave board number. Valid numbers are 1-16 for Master boards and 1-7 for Slave boards (representing I/O pins 32-256 for a domain).
szVisaResource	LPCTSTR	String identifying the location of the specified board in order to establish a session.
pnDensity	PSHORT	Memory module size densities
pnBanks	PSHORT	Number of memory modules installed
pnHandle	PSHORT	Returned Handle (session identifier) that can be used to call any other operations of that resource
pnStatus	PSHORT	Returned status: 0 on success, 1 on failure.

Comments

The function opens a VISA session to the specified resource. The function uses the default VISA provider configured in your system to access the board. You must ensure that the default VISA provider support PXI/PCI devices and that the board is visible in the VISA resource manager before calling this function.

The first argument *szVisaResource* is a string that is displayed by the VISA resource manager such as NI Measurement and Automation (NI_MAX). It is also displayed by Marvin Test Solutions PXI/PCI Explorer as shown in the prior figure. The VISA resource string can be specified in several ways as follows:

- Using chassis, slot: "PXI0::CHASSIS1::SLOT5"
- Using the PCI Bus/Device combination: "PXI9::13::INSTR" (bus 9, device 9).
- Using alias: "COUNTER1". Use the PXI/PCI Explorer to set the device alias.

The function returns a board handle (session identifier) that can be used to call any other operations of that resource. The session is opened with VI_TMO_IMMEDIATE and VI_NO_LOCK VISA attributes. On terminating the application the driver automatically invokes **viClose**() terminating the session.

GC5050/GX5050/GX5150:

- Calling this function will not change the current configuration of the board and will set the board to the HALT mode.
- The memory module densities and the number of memory modules installed *are* read from the board.
- Density values are read back from the board and can be as follow:
  - 0 256K by 32-bit
  - 1 1M by 32-bit
  - 2 2M by 32-bit
  - 3 4M by 32-bit
- The number of Banks is read back from the board can be 1 to 8.

GX5055:

- Calling this function will not change the current configuration of the board and will set the board to the HALT mode.
- The memory module densities and the number of memory modules installed *are* read from the board.
- Density values are read back from the board and can be as follow:
  - 1 512 by 32-bit

GX5280/GX5290/GX5290e/GX5295:

- Calling this function will not change the current configuration of the board and will set the board to the HALT mode.
- The memory module densities and the number of memory modules installed *are* read from the board.
- Density values are read back from the board and can be as follow:
  - 0 32M by 32-bit
  - 1 64M by 32-bit
  - 2 128M by 32-bit

Example

The following example initializes a master board at PXI bus 5 and device 11.

```
SHORT nHandle, nStatus;
DioSetupInitializationVisa (0, 1, "PXI5::11::INSTR", &nHandle, &nStatus);
if (nHandle==0)
{
    printf("Unable to Initialize the board")
    return;
}
```

See Also

DioSetupInitialization, DioInitialize, DioGetSlaveHandle, DioGetMemoryBankSize, DioGetErrorString

DioSetupInputDataSource

Applies To

GX5295, File.

Purpose

Sets the input data comparator source.

Syntax

DioSetupInputDataSource (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, nInputDataSource, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nInputDataSource	PSHORT	Input data comparator can be as follows: 0 DIO_LOW_THRESHOLD_COMPARATOR: data will be the result of the comparison done on the input data by the low threshold comparator, logic levels are as follows: Logic Low: whenever the input signal is below the low comparator threshold. Logic High: whenever the input signal is above the low comparator threshold 1 DIO_HIGH_THRESHOLD_COMPARATOR: data will be the result of the comparison done on the input data by the high threshold comparator, logic levels are as follows: Logic Low: whenever the input signal is below the high comparator threshold. Logic High: whenever the input signal is above the high comparator threshold.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Example

The following example uses an array of channels list to their input data comparator to the low threshold:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioSetupInputDataSource (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0,
DIO_INPUT_DATA_COMPARATOR_LOW_THRESHOLD, &nStatus);
```

The following example sets all the board's channels input data comparator to the low threshold:

```
DioSetupInputDataSource (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0,
DIO_INPUT_DATA_COMPARATOR_LOW_THRESHOLD, &nStatus);
```

The following example sets all the boards channels input data comparator to the low threshold:

```
DioSetupInputDataSource (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0,
DIO_INPUT_DATA_COMPARATOR_LOW_THRESHOLD, &nStatus);
```

The following example sets channels 5 to 10 input data comparator to the low threshold:

```
DioSetupInputDataSource (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10,
DIO_INPUT_DATA_COMPARATOR_LOW_THRESHOLD, &nStatus);
```

See Also

DioGetInputDataSource, DioSetupInputLoadCurrent , DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioSetupInputInterface

Applies To

GX5280, GX5290, GX5290e, File.

Purpose

Sets the input data active interface.

Syntax

DioSetupInputInterface (*nHandle, nInterface, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
nInterface	SHORT	Input Active Interface: 0 DIO_IO_INTERFACE_TTL: Input TTL connector is the active interface. 1 DIO_IO_INTERFACE_LVDS: Input LVDS connector is the active interface.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function select the active input connector LVDS or TTL Standard (3.3V, 2.5V, 1.8V or 1.5V) for each groups of channels when in Input mode.

Example

The following example sets the LVDS as the Input Active Interface:

```
SHORT  nStatus;
DioSetupInputInterface (nHandle, DIO_IO_INTERFACE_LVDS, &nStatus);
```

See Also

DioGetInputInterface, DioGetErrorString

DioSetupInputLoadCurrent

Applies To

GX5055, GX5295, File.

Purpose

Sets the specified channels range input source and sink load currents.

Syntax

DioSetupInputLoadCurrent (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, dISink, dISource, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
dISink	DOUBLE	Input channel constant current sink value in mA. Input channel constant current sink value can be set from 0mA to 24mA with 0.3662 µA resolution. GX5055: The input channel current sink force the specified constant current to be active when the input voltage is below the low commutating voltage value. GX5295: The input channel current sink force the specified constant current to be active when the input voltage is below the commutating voltage value.
dISource	DOUBLE	Input channel constant current source value in mA. Input channel constant current source value can be set from 0mA to 24mA with 0.3662 µA resolution. GX5055: The input channel current source force the specified constant current to be active when the input voltage is above the high commutating voltage value. GX5295: The input channel current source force the specified constant current to be active when the input voltage is above the commutating voltage value.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5055:

Each channel has an independent load with the following capabilities:

- Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
- Maintain high impedance over a wide voltage.
- Separate high and low commutating voltages – **DioSetupInputLoadCommutatingVoltage** function.
- Channels' Input Resistive load options - **DioSetupInputLoadResistance** function.

With independent high and low commutating voltages, the source and sink currents each have their own threshold voltage. If the voltage on the input, when the load is activated, is between the two commutating voltages, the load will remain in a high impedance state.

GX5295:

Each channel has an independent load with the following capabilities:

- Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
- Maintain high impedance over a wide voltage.
- Single commutating voltage – **DioSetupInputLoadCommutatingVoltage** function.

With a single commutating voltage, the source and sink currents have a single threshold voltage.

Channels input source and sink constant currents can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example uses an array of channels list to set the Input sink current to 10mA and sink current to 5mA:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioSetupInputLoadCurrent (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0, 0.01,
0.005, &nStatus);
```

The following example sets all the board's channels set the Input sink current to 10mA and sink current to 5mA:

```
DioSetupInputLoadCurrent (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0, 0.01, 0.005,
&nStatus);
```

The following example sets all the boards channels set the Input sink current to 10mA and sink current to 5mA:

```
DioSetupInputLoadCurrent (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0, 0.01,
0.005, &nStatus);
```

The following example sets channels 5 to 10 set the Input sink current to 10mA and sink current to 5mA:

```
DioSetupInputLoadCurrent (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10, 0.01, 0.005,
&nStatus);
```

See Also

DioSetupInputLoadCurrent , DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioSetupInputLoadCommutatingVoltage

Applies To

GX5055, GX5295, File.

Purpose

Sets the specified channels range input constant current commutating voltage.

Syntax

DioSetupInputLoadCommutatingVoltage (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, dVComHi, dVComLo, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
dVComHi	DOUBLE	GX5055: Input active load sink and source currents high commutating voltages value, voltage settings and range depends on supply rail voltages see comments for details. GX5295: Input active load sink and source currents commutating voltages value, voltage can be set from --2V to +7V.
dVComLo	DOUBLE	GX5055: Input active load sink and source currents low commutating voltages value, voltage settings and range depends on supply rail voltages see comments for details. GX5295: Not used should be set to 0.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5055:

Each channel has an independent load with the following capabilities:

- Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
- Maintain high impedance over a wide voltage.
- Separate high and low commutating voltages – **DioSetupInputLoadCommutatingVoltage** function.
- Channels' Input Resistive load options - **DioSetupInputLoadResistance** function.

Min/Max commutating voltages:

	Min	Max
dVComHi	Low Rail Supply +2V	High Rail Supply -7V
dVComLo	Low Rail Supply +2V	High Rail Supply -7V

E.g. if high rail supply = 18V and low rail supply = -14V then min high commutating voltage >=-11V and max high commutating voltage <=12V.

With independent high and low commutating voltages, the source and sink currents each have their own threshold voltage. If the voltage on the input, when the load is activated, is between the two commutating voltages, the load will remain in a high impedance state.

GX5295:

Each channel has an independent load with the following capabilities:

- Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
- Maintain high impedance over a wide voltage.
- Single commutating voltage – **DioSetupInputLoadCommutatingVoltage** function.

With a single commutating voltage, the source and sink currents have a single threshold voltage.
Channels input constant current commutating voltage can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example uses an array of channels list to set their input current commutating voltage high to 4.2V and voltage low to -5.25V:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioSetupInputLoadCommutatingVoltage (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8,
anChannelList, 0, 4.2, -5.25, &nStatus);
```

The following example sets all the board's channels comparators' input current commutating voltage high to 4.2V and voltage low to -5.25V:

```
DioSetupInputLoadCommutatingVoltage (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0,
4.2, -5.25, &nStatus);
```

The following example sets all the boards channels comparators' input current commutating voltage high to 4.2V and voltage low to -5.25V:

```
DioSetupInputLoadCommutatingVoltage (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0,
4.2, -5.25, &nStatus);
```

The following example sets channels 5 to 10 input current commutating voltage high to 4.2V and voltage low to -5.25V:

```
DioSetupInputLoadCommutatingVoltage (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10,
4.2, -5.25, &nStatus);
```

See Also

DioSetupInputLoadCurrent , DioGetInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioSetupInputLoadResistance

Applies To

GX5055, File.

Purpose

Sets the specified channels range input pull-up and pull down resistive loads.

Syntax

DioSetupInputLoadResistance (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, dPullup, dPulldown, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero.
panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL.

nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero.
dPullup	DOUBLE	Input pull-up resistive loads can be one of the following: • DIO_RESISTIVE_LOAD_240_OHMS = 240 • DIO_RESISTIVE_LOAD_290_OHMS = 290 • DIO_RESISTIVE_LOAD_1K_OHMS = 1000 • DIO_RESISTIVE_LOAD_OPEN = -1
dPulldown	DOUBLE	Input pull down resistive loads can be one of the following: • DIO_RESISTIVE_LOAD_240_OHMS = 240 • DIO_RESISTIVE_LOAD_290_OHMS = 290 • DIO_RESISTIVE_LOAD_1K_OHMS = 1000 • DIO_RESISTIVE_LOAD_OPEN = -1
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each channel has an independent load with the following capabilities:

0. Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
1. Maintain high impedance over a wide voltage.
2. Separate high and low commutating voltages – **DioSetupInputLoadCommutatingVoltage** function.
3. Channels' Input Resistive load options - **DioSetupInputLoadResistance** function.

With independent high and low commutating voltages, the source and sink currents each have their own threshold voltage. If the voltage on the input, when the load is activated, is between the two commutating voltages, the load will remain in a high impedance state.

The input resistive load is useful in applications with very low DUT output swings (where a traditional active load will not switch on and off completely or quickly) and also as a means of forcing the DUT to a known voltage when the DUT is in HiZ.

NOTE: The source and sink currents should be programmed to 0 during normal operation of the resistive load.

When the resistive load is placed in HiZ (DIO_RESISTIVE_LOAD_OPEN) it maintains a low leakage current when the input voltage is between the supply rails.

Channels input pull-up and pull down resistive loads can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example uses an array of channels list to set the input pull-up to 240 ohms and pull down resistive to open:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioSetupInputLoadResistance (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0,
DIO_RESISTIVE_LOAD_240_OHMS, DIO_RESISTIVE_LOAD_OPEN, &nStatus);
```

The following example sets all the board's channels the input pull-up to 240 ohms and pull down resistive to open:

```
DioSetupInputLoadResistance (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0,
DIO_RESISTIVE_LOAD_240_OHMS, DIO_RESISTIVE_LOAD_OPEN, &nStatus);
```

The following example sets all the boards channels the input pull-up to 240 ohms and pull down resistive to open:

```
DioSetupInputLoadResistance (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0,
DIO_RESISTIVE_LOAD_240_OHMS, DIO_RESISTIVE_LOAD_OPEN, &nStatus);
```

The following example sets channels 5 to 10 the input pull-up to 240 ohms and pull down resistive to open:

```
DioSetupInputLoadResistance (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10,
DIO_RESISTIVE_LOAD_240_OHMS, DIO_RESISTIVE_LOAD_OPEN, &nStatus);
```

See Also

DioSetupInputLoadCurrent , DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioSetupInputLoadState

Applies To

GX5055, GX5295, File.

Purpose

Sets the specified channel input load states.

Syntax

DioSetupInputLoadState (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, nState, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nState	SHORT	Load states: 0. DIO_LOAD_DISCONNECTED: Load is not connected. 1. DIO_LOAD_CONNECTED: Load is connected.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5055:

The function enables all the loads to be applied to the specified input channels (constant source and the resistive load). When the load is enabled there is a ~40Ohm resistor in between the input And the loads.

Each channel has an independent load with the following capabilities:

- Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
- Maintain high impedance over a wide voltage.
- Separate high and low commutating voltages – **DioSetupInputLoadCommutatingVoltage** function.
- Channels' Input Resistive load options - **DioSetupInputLoadResistance** function.

With independent high and low commutating voltages, the source and sink currents each have their own threshold voltage. If the voltage on the input, when the load is activated, is between the two commutating voltages, the load will remain in a high impedance state.

GX5295:

Each channel has an independent load with the following capabilities:

- Channels' Input constant source and sink currents up to 24 mA – **DioSetupInputLoadCurrent** function.
- Maintain high impedance over a wide voltage.
- Single commutating voltage – **DioSetupInputLoadCommutatingVoltage** function.
- With a single commutating voltage, the source and sink currents have a single threshold voltage.

Channels input source and sink constant currents can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example uses an array of channels list to enables the Input load state:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioSetupInputLoadState (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0,
DIO_LOAD_CONNECTED, &nStatus);
```

The following example enables all the board's channels Input load states:

```
DioSetupInputLoadState (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0,
DIO_LOAD_CONNECTED, &nStatus);
```

The following example enables all the boards channels Input load states:

```
DioSetupInputLoadState (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0,
DIO_LOAD_CONNECTED, &nStatus);
```

The following example enables channels 5 to 10 Input load state:

```
DioSetupInputLoadState (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10,
DIO_LOAD_CONNECTED, &nStatus);
```

See Also

DioSetupInputLoadCurrent, DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioSetupInputThresholdVoltages

Applies To

GX5055, GX5295, File.

Purpose

Sets the specified channels range input low and high threshold voltages.

Syntax

DioSetupInputThresholdVoltages (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, dHiLevel, dLoLevel, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
dHiLevel	DOUBLE	GX5055: Input high voltage threshold, voltage settings and range depends on supply rail voltages see comments for details. GX5295: Input high voltage threshold, value can be -2.0V to +7.0V and higher then Input low voltage threshold.
dLoLevel	DOUBLE	GX5055: Input low voltage threshold, voltage settings and range depends on supply rail voltages see comments for details. GX5295: Input low voltage threshold, value can be -2.0V to +7.0V and lower then Input high voltage threshold.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5055:

The input voltage can be inside one of the following ranges:

0. Input voltage is higher than high voltage threshold, input is logic high. Data will be logged as logic high to the input memory and 0 to the invalid logic level input memory.
1. Input voltage is lower than low voltage threshold, input is logic low. Data will be logged as logic low to the input memory and 0 to the invalid logic level input memory.
2. Input voltage is higher than low voltage threshold and lower then high voltage threshold, input is invalid. Data will be logged as logic low to the input memory and 1 to the invalid logic level input memory.

Min/Max threshold voltages:

	Min	**Max**
dHiLevel	Low Rail Supply +2V	High Rail Supply -7V
dLoLevel	Low Rail Supply +2V	High Rail Supply -7V

E.g. if high rail supply = 18V and low rail supply = -14V then min high threshold voltage >=-11V and max high threshold voltage <=12V.

GX5295:

The board can only save the input data for all input channels of one of the two input threshold. The user can specify the desired input threshold data to be saved by calling the **DioSetupInputDataSource** function.

Input data source was set to the low input threshold:

0. Input voltage is higher than low voltage threshold, input is logic high. Data will be logged as logic high to the input memory.
1. Input voltage is lower than low voltage threshold, input is logic low. Data will be logged as logic low to the input memory

Input data source was set to the high input threshold:

0. Input voltage is higher than high voltage threshold, input is logic high. Data will be logged as logic high to the input memory.
1. Input voltage is lower than high voltage threshold, input is logic low. Data will be logged as logic low to the input memory

Channels input low and high threshold voltages can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example uses an array of channels list to set their input high thresholds to 6.25V and input low threshold to -2.25V:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioSetupInputThresholdVoltages (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0, 6.25, -2.25, &nStatus);
```

The following example sets all the board's channels input high thresholds to 6.25V and input low threshold to -2.25V:

```
DioSetupInputThresholdVoltages (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0, 6.25, -2.25, &nStatus);
```

The following example sets all the boards channels input high thresholds to 6.25V and input low threshold to -2.25V:

```
DioSetupInputThresholdVoltages (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0, 6.25, -2.25, &nStatus);
```

The following example sets channels 5 to 10 input high thresholds to 6.25V and input low threshold to -2.25V:

```
DioSetupInputThresholdVoltages (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10, 6.25, -2.25, &nStatus);
```

See Also

DioSetupInputLoadState, DioSetupInputLoadCurrent, DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioSetupIOConfiguration

Applies To

GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Sets the I/O memory width and direction.

Syntax

DioSetupIOConfiguration *(nHandle, nWidth, nDirection, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or File board handle.
nWidth	SHORT	**GX5150**: 0. 32 bits wide 1. 16 bits wide 2. bits wide **GX5283:** 0 32 bits wide, maximum number of steps is 128M, channels 0-31 1 16 bits wide, maximum number of steps is 256M, channels 0-15 2 8 bits wide, maximum number of steps is 512M, channels 0-7 3 4 bits wide, maximum number of steps is 1G, channels 0-3 4 2 bits wide, maximum number of steps is 2G, channels 0-1 5 1 bit wide, maximum number of steps is 4G, channel 0 **GX5292/GX5293/GX5292e/GX5293e/GX5295:** 0. 32 bits wide, maximum number of steps is 64M, channels 0-31 1. 16 bits wide, maximum number of steps is 128M, channels 0-15 2. bits wide, maximum number of steps is 256M, channels 0-7 3. bits wide, maximum number of steps is 512G, channels 0-3 4. bits wide, maximum number of steps is 1G, channels 0-1 5. bit wide, maximum number of steps is 2G, channel 0

nDirection	SHORT	**GX5150:** Selects whether the I/O Memory configured as input or output. 0 Input mode. 1 Output mode. **GX5280:** I/O Group channel direction bits 0-3. Each bit represents the direction of Group of 8 channels. 0 Input mode. 1 Output mode. **GX5290/GX5290e/GX5295:** Not applicable.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in HALT state before calling this function.

The I/O memory width determines the number of I/O pins used for all steps for the specified board.

All not used I/O channels pins will be set to Tri-state.

The last setting of the I/O configuration is saved to the DIO.INI file, to be used whenever calling **DioInitialize**.

GX5150:

If **Frequency Doubler** I/O module is installed, then the function will check if the current setting is valid. If width was set to 2, then the function returns an error. Valid setting for the **Frequency Doubler** I/O module width are 1 (16 channels) and 2-(8 channels).

GX5283/GX5292/GX5293/GX5292e/GX5293e/GX5295:

The I/O memory width determines the number of I/O pins used for all steps for the specified board. If the I/O Memory width was set to Word or Byte, the remaining I/O pins will be set to Tri-state. Calling this function will reset the all the DIO boards in the domain and will clear all the commands.

GX5293/GX5293e: for frequencies above 100MHz the user must set the number of active I/O channels (wide) to be 16 or lees, see the **DioSetupIOConfiguration** function for more details.

The last setting of the I/O configuration is saved to the GTDIO.INI file, to be used whenever calling **DioInitialize**.

Example

The following example sets the I/O memory width to 32-bit and the direction to input:

```
SHORT  nStatus;
DioSetupIOConfiguration (nHandle, 0, 1, pnStatus):
```

See Also

DioGetIOConfiguration, DioGetErrorString

DioSetupIOPinsStaticDirection

Applies To

GX5055, GX5295

Purpose

Sets the I/O pins channels direction when in static mode.

Syntax

DioSetupIOPinsStaticDirection *(nHandle, dwDirection, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
dwDirection	DWORD	The 32 I/O channels directions are represented by the 32-bits. Each bit represents a channel starting from 0, e.g. channel zero is represented by bit 0. A bit high means the channel is set to output while a bit low means channel is set to input.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Example

The following example sets the I/O pins channels direction when in static mode:

```
SHORT  nStatus;
DioSetupIOPinsStaticDirection (nHandle, 0x55AA55AA, &nStatus);
```

See Also

DioGetIOPinsStaticDirection, DioReadIoPinsValidity, DioReadIOPinsValue DioWriteIOPinsValue, DioGetErrorString

DioSetupOutputClocksState

Applies To

GX5280, GX5290, GX5290e, GX5295

Purpose

Sets the specified output clock pin state.

Syntax

DioSetupOutputClocksState *(nHandle, nClock, nState, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or Slave board handles.
nClock	SHORT	specified output clocks are: 0. DIO_OUT_CLOCK – output clock (J3 pin 22). 1. DIO_OUT_STROBE – output clock (J3 pin 24). 2. DIO_OUT_B_CLOCK – output clock (J3 pin 20). 3. DIO_OUT_LVDS_CLOCK – output clock (J4 pins 4 and 38).
nState	SHORT	States values are: 0. DIO_CLOCK_OUTPUT_DISABLE – specified clock output pin is disabled. 1. DIO_CLOCK_OUTPUT_ENABLE– specified clock output pin is enabled.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The DIO needs to be in HALT mode before calling this function.

This function sets the specified output clock signal pin state to be enabled or disabled state.

NOTE: settings will be overwritten whenever calling reset, settings main clock frequency or settings B clock frequency.

Example

The following example enables the output strobe signal:

```
SHORT  nStatus, nOutputState;
DioSetupOutputClocksState (nHandle, DIO_OUT_STROBE, DIO_CLOCK_OUTPUT_ENABLE, &nStatus);
```

See Also

DioGetOutputClocksState, DioWriteIOPinsValue, DioReadIOPinsValue, DioGetErrorString

DioSetupOutputDataFormat

Applies To

GX5055, File.

Purpose

Sets the specified channel output data format.

Syntax

DioSetupOutputDataFormat (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, nDataFormat, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero.
nDataFormat	SHORT	Output data format can be as follows: 0. DIO_OUTPUT_DATA_FORMAT_NR: No Return 1. DIO_OUTPUT_DATA_FORMAT_R0: Return to Zero 2. DIO_OUTPUT_DATA_FORMAT_R1: Return to One 3. DIO_OUTPUT_DATA_FORMAT_RZ: Return to Hi-Z 4. DIO_OUTPUT_DATA_FORMAT_RC: Return to Complement
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

DIO_OUTPUT_DATA_FORMAT_NR: No Return, the output logic level stay either high or low for the duration of the clock period.

DIO_OUTPUT_DATA_FORMAT_R0: Return to Zero. The signal returns to zero between consecutive data clocks at 50% of the specified output frequency clock duty cycle. This takes place even if a number of consecutive 0's or 1's occur in the signal. The signal is self clocking and does not require any additional settings.

DIO_OUTPUT_DATA_FORMAT_R1: Return to One. The signal returns to one between consecutive data clocks at 50% of the specified output frequency clock duty cycle. This takes place even if a number of consecutive 0's or 1's occur in the signal. The signal is self clocking and does not require any additional settings.

DIO_OUTPUT_DATA_FORMAT_RZ: Return to Hi-Z .The signal returns to hi-z between consecutive data clocks at 50% of the specified output frequency clock duty cycle. This takes place even if a number of consecutive 0's or 1's occur in the signal. The signal is self clocking and does not require any additional settings.

DIO_OUTPUT_DATA_FORMAT_RC: Return to Complement (RC). In Return to Complement (also called Manchester code) each transmitted data bit has at least one transition at 50% of the specified output frequency clock duty cycle. It is, therefore, self-clocking, which means that a clock signal can be recovered from the encoded data. Return to Complement ensures frequent line voltage transitions, directly proportional to the clock rate, this helps clock recovery. Logic low is expressed by a low-to-high transition. Logic high is expressed by high-to-low transition. The transitions which signify logic high or low occur at the midpoint of a period, the direction of the mid-bit transition indicates the data.

NOTE: the specified channel data format will be applied to all the channels' steps.

Channels output data format can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example uses an array of channels list sets their input data format to No Return:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioSetupOutputDataFormat(nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0,
DIO_OUTPUT_DATA_FORMAT_NR, &nStatus);
```

The following example sets all the board's channels input data format to No Return:

```
DioSetupOutputDataFormat(nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0,
DIO_OUTPUT_DATA_FORMAT_NR, &nStatus);
```

The following example sets all the boards channels input data format to No Return:

```
DioSetupOutputDataFormat(nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0,
DIO_OUTPUT_DATA_FORMAT_NR, &nStatus);
```

The following example sets channels 5 to 10 input data format to No Return:

```
DioSetupOutputDataFormat(nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10,
DIO_OUTPUT_DATA_FORMAT_NR, &nStatus);
```

See Also

DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioSetupOutputOverCurrentEnable

Applies To

GX5055

Purpose

Sets the channels over-current enable states.

Syntax

DioSetupOutputOverCurrentEnable (*nHandle, dwOverCurrentEnable, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
dwOverCurrentEnable	DWORD	Over-current flag state, each bit corresponds to a channel i.e. bit 0 represents channel 0 and bit 31 represents channel 31. • A bit with logic high bit will enable the over current circuit, when output over current condition is detected, sink or source, it will automatically set to its output in Hi-Z. Enabling channels outputs which were set to Hi-Z as a result of over current can only be reset by calling **ResetOutputOverCurrentStates** function. • A bit with logic low will ignore over current condition and will not set it to Hi-Z.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

NOTE over current condition can only be cleared by calling **DioResetOutputOverCurrentStates**. After the over current is cleared the channel output driver will be active again.

Channels over current high and low flags states can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example enables first 4 channels over current protection flags:

```
DioSetupOutputOverCurrentEnable (nHandle, 0x0000000F, &nStatus);
```

See Also

DioGetOutputOverCurrentEnable, DioSetupOutputDataFormat, DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioSetupOutputSlewRate

Applies To

GX5055, File.

Purpose

Sets the specified channel output rising and falling slew rate.

Syntax

DioSetuptOutputSlewRate (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, dRisingEdge, dFallingEdge, dBias, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero.
dRisingEdge	DOUBLE	Output rising slew rate in Volts per nannosec, values are from 0.1V/ns to 1.2V/ns in steps of 0.1 V/ns.
dFallingEdge	DOUBLE	Output falling slew rate will have the same settings as rising edge. Currently not used.
dBias	DOUBLE	The driver output stage has a programmable bias current to allow applications that require slower edge rates to consume less power. Currently not use.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each channel output has rising and falling slew rate capability with the following characteristics:

- Each output driver has a separate and independent adjustments for the rising and falling slew rate
- Each output driver output stage has a programmable bias current to allow applications that require slower edge rates to consume less power.
- Separate and independent delay circuitry for each channel.

Channels output rising and falling slew rates can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example uses an array of channels listsets output rising and falling slew rate:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioSetupOutputSlewRate (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0, 0.5, 0,
0, &nStatus);
```

The following example sets all the board's channels output rising and falling slew rate:

```
DioSetupOutputSlewRate (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL, 0, 0.5, 0, 0,
&nStatus);
```

The following example sets all the boards channels output rising and falling slew rate:

```
DioSetupOutputSlewRate (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL, 0, 0.5, 0, 0,
&nStatus);
```

The following example sets channels 5 to 10 output rising and falling slew rate:

```
DioSetupOutputSlewRate (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10, 0.5, 0, 0,
&nStatus);
```

See Also

DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString

DioSetupOutputState

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e

Purpose

Sets the output pins state.

Syntax

DioSetupOutputState *(nHandle, nOutputState, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or Slave board handle.
nOutputState	SHORT	States values are: 0. Enabled-Outputs retain last value. 1. Disabled-Outputs are set to Tri-state. 2. Disabled on Halt/Pause-Outputs are set to Tri-state only when in HALT or PAUSE. 3. Outputs are set to zero. 4. Outputs are set to one (GX5280/GX5290 only).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function sets the output driver to enable or disable state. It has no effect on the DIO running mode or the loaded vector. When issued, this command sets all 32 I/O lines to the specified state regardless of DIO state. That is, if the board outputs are set to zero while the board is in RUN state, then the outputs will remain at zero while the board is still running.

The state setting does not have any affect if the board is in input mode.

Example

The following example sets the outputs to retain last value:

```
SHORT   nStatus, nOutputState;
DioSetupOutputState (nHandle, 0, &nStatus);
```

See Also

DioGetOutputState, DioWriteIOPinsValue, DioReadIOPinsValue, DioGetErrorString

DioSetupOutputVoltages

Applies To

GX5055, GX5295, File.

Purpose

Setup the specified channel output driver high and low voltages.

Syntax

DioSetupOutputVoltages (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, dHiLevel, dLoLevel, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, than it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
dHiLevel	DOUBLE	GX5055: Output driver high voltage corresponding to logic high voltage settings and range depends on supply rail voltages see comments for details. GX5295: Output driver high voltage corresponding to logic high. Voltage can be set from -2V to +7V and must be greater than the output driver low voltage.
dLoLevel	DOUBLE	GX5055: Output driver low voltage corresponding to a logic low voltage settings and range depends on supply rail voltages see comments for details. GX5295: Output driver low voltage corresponding to a logic low. Voltage can be set from -2V to +7V and must be lower than the output driver high voltage.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Each channels' output driver has two level driver, low and high level.
Channels output driver voltages can be read back and set dynamically at any time even while the DIO is running mode.

GX5055:
The total output driver voltage swing (Output driver high voltage less Output driver low voltage) is limited to 24V per channel.

	Min	**Max**
dHiLevel	Low Rail Supply +5V	High Rail Supply -3V
dLoLevel	Low Rail Supply +4V	High Rail Supply -7V
dHiLevel - dLoLevel	0.5V	

E.g. if high rail supply = 18V and low rail supply = -14V then min output driver high voltage >=-11V and max output driver high voltage <=12V.

Example

The following example uses an array of channels list sets output driver high voltage to +4.5V and low voltage to -3.25V:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioSetupOutputVoltages (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0, 4.5, -3.25, &nStatus);
```

The following example sets all the board's channels output driver high voltage to +4.5V and low voltage to -3.25V:

```
DioSetupOutputVoltages (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0, 4.5, -3.25, &nStatus);
```

The following example sets all the boards channels output driver high voltage to +4.5V and low voltage to -3.25V:

```
DioSetupOutputVoltages (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0, 4.5, -3.25, &nStatus);
```

The following example sets channels 5 to 10 output driver high voltage to +4.5V and low voltage to -3.25V:

```
DioSetupOutputVoltages (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10, 4.5, -3.25, &nStatus);
```

See Also

DioSetupOutputSlewRate, **DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString**

DioSetupPxiStarTriggerMode

Applies To

GX5280, GX5290, GX5290e, GX5295, File

Purpose

Sets the board PXI Star Trigger mode.

Syntax

DioSetupPxiStarTriggerMode *(nMasterHandle, nMode, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
nMode	SHORT	PXI Star Trigger mode: 0 DIO_PXI_STAR_TRIGGER_DISABLED: the DIO star trigger input is disabled. 1 DIO_PXI_STAR_TRIGGER_TO_PAUSE: the DIO star trigger input will pause the board. 2 DIO_PXI_STAR_TRIGGER_TO_TRIGGER: the DIO star trigger input will trigger the board.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

When the DIO PXI Star Trigger input is set to either trigger or pause the board the input signal can be programmed to respond to low level or rising edge. See for details **DioSetupSignalEdgeOrLevelMode**

Example

The following example sets the PXI Star Trigger input mode to trigger:

```
SHORT nMode;
DioSetupPxiStarTriggerMode (nMasterHandle, DIO_PXI_STAR_TRIGGER_TO_TRIGGER, &nStatus);
```

See Also

DioGetPxiStarTriggerMode, DioSetupSignalEdgeOrLevelMode, **DioSetupPxiTriggerBusLineMode, DioGetErrorString**

DioSetupPxiTriggerBusLineMode

Applies To

GX5280, GX5290, GX5290e, GX5295, File

Purpose

Sets the specified PXI Trigger Bus Line mode.

Syntax

DioSetupPxiTriggerBusLineMode *(nMasterHandle, nLine, nMode, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
nLine	SHORT	PXI Trigger Bus Line: 0 DIO_PXI_TRIGGER_BUS_LINE0 1 DIO_PXI_TRIGGER_BUS_LINE1 2 DIO_PXI_TRIGGER_BUS_LINE2 3 DIO_PXI_TRIGGER_BUS_LINE3 4 DIO_PXI_TRIGGER_BUS_LINE4 5 DIO_PXI_TRIGGER_BUS_LINE5 6 DIO_PXI_TRIGGER_BUS_LINE6 7 DIO_PXI_TRIGGER_BUS_LINE7
nMode	SHORT	PXI Trigger Bus Line mode: 0 DIO_PXI_TRIGGER_BUS_LINE_MODE_DISABLE: the specified PXI Trigger Line is disabled. 1 DIO_PXI_TRIGGER_BUS_LINE_MODE_PAUSE_INPUT: the specified PXI Trigger Line is set to as input pause. 2 DIO_PXI_TRIGGER_BUS_LINE_MODE_PAUSE_OUTPUT: the specified PXI Trigger Line is set to as output pause. 3 DIO_PXI_TRIGGER_BUS_LINE_MODE_TRIGGER_INPUT: the specified PXI Trigger Line is set to as input trigger. 4 DIO_PXI_TRIGGER_BUS_LINE_MODE_TRIGGER_OUTPUT: the specified PXI Trigger Line is set to as output trigger.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function allows you to route trigger and paused signals to a specific PXI Trigger Bus line. The trigger and paused signals can be outputs to other Master DIO's or instruments that share the PXI Trigger Bus and/or inputs signals. The Trigger Bus Line assignments are mutually exclusive and can be set in tandem.

Example

The following example sets the PXI Trigger Bus Line number 1 mode as output trigger:

```
DioSetupPxiTriggerBusLineMode (nMasterHandle, DIO_PXI_TRIGGER_BUS_LINE1,
                  DIO_PXI_TRIGGER_BUS_LINE_MODE_TRIGGER_OUTPUT, &nStatus);
```

See Also

DioGetPxiTriggerBusLineMode, DioGetErrorString

DioSetupSequencerMode

Applies To

GX5150

Purpose

Enables or disables the specified Master's sequencer.

Syntax

DioSetupSequencerMode *(nMasterHandle, nSequencerMode, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
nSequencerMode	SHORT	0 Sequencer disabled. 1 Sequencer enabled (default).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in the HALT or PAUSE state.

Disabling the sequencer causes the Master to disregard any control memory command or external **JumpA**. This is useful for debugging, or if the content of the control memory is unknown.

The sequencer is enabled after reset (by default).

Example

The following example enables the specified board's sequencer:

```
SHORT  nStatus;
DioSetupSequencerMode (nMasterHandle, 1 &nStatus);
```

See Also

DioGetSequencerMode, DioGetErrorString

DioSetupSignalEdgeOrLevelMode

Applies To

GX5280, GX5290, GX5290e, GX5295

Purpose

Sets the specified Signal edge or level active mode.

Syntax

DioSetupSignalEdgeOrLevelMode (*nMasterHandle, nSignal, nMode, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
nSignal	SHORT	Signal can be one of the following: 0 DIO_EXTERNAL_STROBE_ENABLE 1 DIO_EXTERNAL_CLOCK_ENABLE 2 DIO_EXTERNAL_PAUSE 3 DIO_EXTERNAL_TRIGGER 4 DIO_OUTPUT_RUN 5 DIO_OUTPUT_ARM 6 DIO_PXI_STAR_TRIGGER 7 DIO_PXI_TRIGGER_BUS See Comments below for details.
nMode	SHORT	Signal active mode can be one of the following: 0 DIO_SIGNAL_ACTIVE_LOW - signal is active when low level is present (default). 1 DIO_SIGNAL_ACTIVE_HIGH - signal is active when high level is present. 2 DIO_SIGNAL_ACTIVE_RISING_EDGE - signal logic when low to high transient occur. 3 DIO_SIGNAL_ACTIVE_FALLING_EDGE - signal is active when high to low transient occur. See Comments below for details.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

This function returns the current settings of the specified active mode for input control signals and output monitoring signals residing on the Timing connector. Each signal can be set as follows:

DIO_EXTERNAL_STROBE_ENABLE:

The external strobe enable input signal Active mode (Timing connector) can be set as one of the following:

0. DIO_SIGNAL_ACTIVE_LOW – signal is active when low level is present (default).
1. DIO_SIGNAL_ACTIVE_HIGH – signal is active when high level is present.

DIO_EXTERNAL_CLOCK_ENABLE:

The external clock enable input signal Active mode (Timing connector) can be set as one of the following:

0. DIO_SIGNAL_ACTIVE_LOW – signal is active when low level is present (default).
1. DIO_SIGNAL_ACTIVE_HIGH – signal is active when high level is present.

DIO_EXTERNAL_PAUSE:

The external pause enable input signal Active mode (Timing connector) can be set as one of the following:

0. DIO_SIGNAL_ACTIVE_LOW – signal is active when low level is present (default).
1. DIO_SIGNAL_ACTIVE_HIGH – signal is active when high level is present.
2. DIO_SIGNAL_ACTIVE_RISING_EDGE – signal logic when low to high transient occur.
3. DIO_SIGNAL_ACTIVE_FALLING_EDGE – signal is active when high to low transient occur.

DIO_EXTERNAL_TRIGGER:

The external trigger input signal Active mode (Timing connector) can be set as one of the following:

0. DIO_LOGIC_ACTIVE_LOW – signal is active when low level is present (default).
1. DIO_LOGIC_ACTIVE_HIGH – signal is active when high level is present.
2. DIO_LOGIC_ACTIVE_RISING_EDGE – signal logic when low to high transient occur.
3. DIO_LOGIC_ACTIVE_FALLING_EDGE – signal is active when high to low transient occur.

DIO_OUTPUT_RUN:

The output run signal Active mode (Timing connector) can be set as one of the following:

0. DIO_LOGIC_ACTIVE_LOW – signal is active when low level is present (default).
1. DIO_LOGIC_ACTIVE_HIGH – signal is active when high level is present.

DIO_OUTPUT_ARM:

The external arm input signal Active mode (Timing connector) can be set as one of the following:

0. DIO_LOGIC_ACTIVE_LOW – signal is active when low level is present (default).
1. DIO_LOGIC_ACTIVE_HIGH – signal is active when high level is present.

DIO_PXI_STAR_TRIGGER:

The external star trigger input signal Active mode (Timing connector) can be set as one of the following:

1. DIO_LOGIC_ACTIVE_LOW – signal is active when low level is present (default).
2. DIO_LOGIC_ACTIVE_RISING_EDGE – signal logic when low to high transient occur.

DIO_PXI_TRIGGER_BUS:

The external PXI trigger bus input signal Active mode (Timing connector) can be set as one of the following:

1. DIO_LOGIC_ACTIVE_LOW – signal is active when low level is present (default).
2. DIO_LOGIC_ACTIVE_RISING_EDGE – signal logic when low to high transient occur.

Example

The following example sets the external strobe enable input signal to be active on rising edge:

```
DioSetupSignalEdgeOrLevelMode (nMasterHandle, DIO_EXTERNAL_STROBE_ENABLE,
                       DIO_SIGNAL_ACTIVE_RISING_EDGE, &nStatus);
```

See Also

DioGetSignalEdgeOrLevelMode, **DioSetupPxiTriggerBuslineMode, DioGetErrorString**

DioSetupSkewDelay

Applies To

GX5055, GX5295, File.

Purpose

Sets the specified channel skew delay.

Syntax

DioSetupSkewDelay (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, nSource, nEdge, dDelay, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nSource	SHORT	Delay source can be as follows: 0 DIO_SKEW_DELAY_OUTPUT_DATA: add delay to the output driver data in reference to the DIO clock. 1 DIO_SKEW_DELAY_OUTPUT_DATA_ENABLE: add delay to the output driver data enable in reference to the DIO clock. Delaying the output data enable adds an additional delay output driver data in reference to the DIO clock. 2 DIO_SKEW_DELAY_INPUT_LOW_THRESHOLD: apply the delay to the input low voltage threshold comparator. 3 DIO_SKEW_DELAY_INPUT_HIGH_THRESHOLD: apply the delay to the input high voltage threshold comparator.
nEdge	SHORT	Delay edge can be as follows: 0. DIO_RISING_EDGE: signal rising edge. 1. DIO_FALLING_EDGE: signal falling edge.

dDelay	DOUBLE	GX5055 Output Rising Edge: Output skew delay can be set from 0 ns to +9.6875 ns with 312.5 ps resolution. GX5055 Input falling edge: Skew delay can be set from -2.5 ns to +2.1875 ns with 312.5 ps resolution. GX5295 Output Rising Edge: Output skew delay can be set from 0 ns to + 4.6875 ns with 19.53 ps resolution. The propagation delay circuitry adds timing delay to the rising edge and the falling edge in equal amounts. Propagation delay adjustment is typically used for aligning the timing of multiple channels inside a tester. GX5295 Output Falling Edge: Output falling edge skew delay can be set from -2.5 ns to +2.1875 ns with 19.53 ps resolution. The falling edge delay circuitry adds or subtracts timing delay to or from the falling edge while having no effect on the rising edge. Propagation delay adjustment is typically used for removing any pulse width distortion inside a tester. GX5295 Input Rising Edge: Input skew delay can be set from 0 ns to + 4.6875 ns with 19.53 ps resolution. The propagation delay circuitry adds timing delay to the rising edge and the falling edge in equal amounts. Propagation delay adjustment is typically used for aligning the timing of multiple channels inside a tester. GX5295 Input Falling Edge: Input falling edge skew delay can be set from -156.25 ps to +146.5 ps with 9.76 ps resolution. The falling edge delay circuitry adds or subtracts timing delay to or from the falling edge while having no effect on the rising edge. Propagation delay adjustment is typically used for removing any pulse width distortion inside a tester.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Output skew delays:

Each channel output has timing capability with the following characteristics:

- Separate and independent delay circuitry for the driver output data and the driver output data enable.
- Separate and independent delay circuitry for each channel.
- Propagation delay adjust (both rising and edge Tpd are delayed equally).

Input skew delays:

Each channel's input high and low voltage threshold comparators have timing capability with the following characteristics:

- Separate and independent delay circuitry for the high and low voltage threshold comparators.
- Separate and independent delay circuitry each channel.
- Propagation delay adjust (both rising and edge Tpd are delayed equally).

Channels output rising and falling edge skew delays can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example uses an array of channels list set delay edge skew of the output data to 2.0ns:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioSetupSkewDelay (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList, 0,
DIO_OUTPUT_SKEW_DELAY_DATA, DIO_RISING_EDGE, 2.0, &nStatus);
```

The following example sets all the board's channels delay edge skew of the output data to 2.0ns:

```
DioSetupSkewDelay (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL, 0,
DIO_OUTPUT_SKEW_DELAY_DATA, DIO_RISING_EDGE, 2.0, &nStatus);
```

The following example sets all the boards channels delay edge skew of the output data to 2.0ns:

```
DioSetupSkewDelay (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL, 0,
DIO_OUTPUT_SKEW_DELAY_DATA, DIO_RISING_EDGE, 2.0, &nStatus);
```

The following example sets channels 5 to 10 delay edge skew of the output data to 2.0ns:

```
DioSetupSkewDelay (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10,
DIO_OUTPUT_SKEW_DELAY_DATA, DIO_RISING_EDGE, 2.0, &nStatus);
```

See Also

DioGetSkewDelay, **DioSetupInputLoadCommutatingVoltage, DioSetupInputLoadResistance, DioSetupInputThresholdVoltages, DioGetErrorString**

DioSetupTermination

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e

Purpose

Enables or disables the specified I/O board module termination.

Syntax

DioSetupTermination (*nHandle, nTermination, pnStatus*).

Parameters

Name	Type	Comments
nHandle	SHORT	Board handle.
nTermination	SHORT	GC5050/GX5050/GX5150 Termination mode values are as follows: 0 Disable. 1 Enable. GX5280/GX5290/GX5290e: I/O Group Terminators bits 0-3. Each bit represents Group of 8 channels terminators state. Bit Low: Terminators Off, Bit High: Terminators Off.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

If the current I/O module does not support termination option, the function returns an error. See the specific I/O module manual to check if it supports this option.

Example

The following example enables the termination for a board specified by the board handle:

```
SHORT   nStatus;
DioSetupTermination (nHandle, 1, &nStatus);
```

See Also

DioGetTermination, DioGetIOModuleType, DioGetErrorString

DioSetupTriggerDEvent

DioSetupTriggerPEvent

DioSetupTriggerTEvent

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Sets up the specified Master board trigger register (D, P or T) mask and event values.

Syntax

DioSetupTriggerDEvent *(nMasterHandle, wEvent, wMask, pnStatus)*

DioSetupTriggerPEvent *(nMasterHandle, wEvent, wMask, pnStatus)*

DioSetupTriggerTEvent *(nMasterHandle, wEvent, wMask, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
wEvent	WORD	Event register values (0 default).
wMask	WORD	Mask register values (0xFFFF default).
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The **DioSetupTriggerMode** must be called to enable conditional triggering using external event lines.

The mask register value (*wMask*) is ANDed with the external event input lines. A result equal to the event register value (*wEvent*), creates a trigger event. A mask register value of 0 always creates a trigger event.

Example

The following example demonstrates two levels trigger (DT Trigger Mode) and sets the P Events and P Mask registers. An external event input line 1 value of '1' and external event input line 4 value of '0' creates the D Trigger event. An external event input lines 3, 4 and 12 value of '1' creates the T Trigger event. These two trigger events in sequence create a sequencer trigger event.

```
DioSetupTriggerDEvent (nMasterHandle, 0x0002, 0x0012, &nStatus);
DioSetupTriggerTEvent (nMasterHandle, 0x1018, 0x1018, &nStatus);
DioSetupTriggerMode (nMasterHandle, 3, &nStatus);
```

See Also

DioGetTriggerDEvent, DioGetTriggerPEvent, DioGetTriggerTEvent, DioSetupTriggerMode, DioArm, DioTrig, DioSetupTriggerXEventSource

DioSetupTriggerMode

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Sets the external event lines trigger mode.

Syntax

DioSetupTriggerMode *(nMasterHandle, nMode, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
nMode	SHORT	Specifies one of the following trigger modes: 0 Conditional Trigger and Pause Disabled (default). 1 Triggers on D events. 2 Triggers on T events. 3 Triggers first on D events, then on T events. 4 Triggers first on T events, then on D events.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The **DioTrig** function or the external trigger control line will force a trigger event overriding the conditional trigger mode set by this function.

Available Trigger modes are:

0 Conditional trigger and pause events disabled. Sequencer commands using D Register disabled.

1 Enables conditional trigger and pause events. DIO creates trigger events when external event input lines ANDed with D mask register equals the D event register.

2 Enables conditional trigger and pause events. DIO creates trigger events when external event input lines ANDed with T mask register equals the T event register.

3 Enables conditional trigger and pause events. This is a two-condition trigger. The first condition is the external event input lines ANDed with the D mask register equals the D event register The second condition is the external event input lines ANDed with T mask register equal the T event register. These are sequential conditions; the D trigger event must occur before the T Trigger event will cause a sequencer trigger event.

4 Enables conditional trigger and pause events. This is a two-condition trigger. The first condition is that external event input lines ANDed with the T mask register equals the T event register. The second condition is the external event input lines ANDed with D mask register equal the D event register. Those are sequential conditions. The T trigger event must take place before the D Trigger event will cause a sequencer trigger event.

Pause event control:

The Pause event control is enabled whenever the trigger mode is not in disabler mode the user need to follow the following guide lines in order to prevent conflicting conditions of trigger and pause. The PEvent value needs to be different then <>X Register (or the actual External Events lines if they are used) **AND** PEvent value needs to be different then the trigger event itself to prevent conflicting conditions. E.g. if the Trigger Mode was set to D, then the DEvent value need to be different then <> PEvent value.

Note: The board must be in the PAUSE state to enable triggering.

Example

The following example sets the trigger mode to trigger when external event lines ANDed with D mask register equals the D event register:

```
SHORT nStatus;
DioSetupTriggerMode (nMasterHandle, 1, &nStatus);
```

See Also

DioGetTriggerMode, DioSetupTriggerDEvent, DioSetupTriggerPEvent, DioSetupTriggerTEvent, DioArm, DioTrig

DioSetupTriggerXEventSource

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290, GX5290e, GX5295, File

Purpose

Sets the specified Master board external event lines source.

Syntax

DioSetupTriggerXEventSource (*nMasterHandle, nSource, pnStatus*)

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
nSource	SHORT	Values are: 0 The source is external event lines (default). 1 The source is the X register.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The function is used to simulate external event lines. When *nSource* is 1, the *X* register value is used instead of external event lines on trigger, pause and conditional commands in the running vector. **DioWriteXRegister** can be used to set the value of the *X* register.

Example

The following example demonstrates how to simulate external event lines to change the board state to PAUSE:

```
DioWriteXRegister (nMasterHandle, 0, &nStatus);
DioSetupTriggerXEventSource (nMasterHandle, 1, &nStatus);
DioSetupTriggerPEvent (nMasterHandle, 0xFFFF, 0x000A, &nStatus);
DioWriteXRegister (nMasterHandle, 0x000A, &nStatus);
```

See Also

DioWriteXRegister, DioSetupTriggerMode

DioSetupTriStateTerminationMode

Applies To

GX5295, File.

Purpose

Setup the specified channel Tri-State termination mode.

Syntax

DioSetupTriStateTerminationMode (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, nMode, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nMode	SHORT	Returned Tri-State termination mode: 0 DIO_TRI_STATE_TERMINATION_MODE_DEFAULT: Default termination mode, in this mode the channel will be set to Hi-Z whenever the channel direction is set to input. 1 DIO_TRI_STATE_TERMINATION_MODE_LEVEL: in this mode the channel will be set to the voltage as specified in **DioSetupTriStateTerminationVoltage** API whenever the channel direction is set to input.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5295:

The spcifed termination voltage is the same voltage that is used by the input active load commutating voltage. Channels termination voltage can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example returns the specified channel Tri-State termination mode:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioSetupTriStateTerminationMode (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8, anChannelList,
0, DIO_TRI_STATE_TERMINATION_MODE_LEVEL, &nStatus);
```

The following example sets all the board's channels Tri-State termination mode:

```
DioSetupTriStateTerminationMode (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0,
DIO_TRI_STATE_TERMINATION_MODE_LEVEL, &nStatus);
```

The following example sets all the boards' channels Tri-State termination mode:

```
DioSetupTriStateTerminationMode (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0,
DIO_TRI_STATE_TERMINATION_MODE_LEVEL, &nStatus);
```

The following example sets channels 5 to 10 Tri-State termination mode:

```
DioSetupTriStateTerminationMode (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10,
DIO_TRI_STATE_TERMINATION_MODE_LEVEL, &nStatus);
```

See Also

DioGetTriStateTerminationMode, DioSetupTriStateTerminationVoltage, **DioSetupInputLoadCurrent, DioSetupInputLoadCommutatingVoltage, DioGetErrorString**

DioSetupTriStateTerminationVoltage

Applies To

GX5295, File.

Purpose

Setup the specified channel Tri-State termination voltage.

Syntax

DioSetupTriStateTerminationVoltage (*nHandle, nChannelListMode, nCountOrFirstChannel, panChannelList, nLastChannel, dVoltage, pnStatus*)

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
nChannelListMode	SHORT	Channel list mode dictates how *nCountOrFirstChannel* and *panChannelList* parameters are used. Channel list mode options are as follows: 0 DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS: The user specifies an array of channels where *nCountOrFirstChannel* is the number of elements in the list array and *panChannelList* is an array contains the channels numbers. In this mode *nLastChanne*l is ignored. 1 DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS: Apply the settings to all the board's channels associated with this board handle. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 2 DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS: Apply the settings to all the channels in the domain, e.g. if there are two boards in the domain (Master and a Slave), then all 64 channels will be set. In this mode *nCountOrFirstChannel, panChannelList* and *nLastChannel* variavles are ignored. 3 DIO_CH_LIST_MODE_RANGE_OF_CHANNELS: Apply settings to a range of channels where *nCountOrFirstChannel* is the first channel number and *nLastChannel* is the last channel number. In this mode *panChannelList* variavle is ignored.
nCountOrFirstChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, then it is specifying the number of elements in the panChannelList array parameter. If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the first channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.

panChannelList	PSHORT	Array of channels numbers. Channels numbers can be from 0 to the last channel in the domain. E.g., if the domain has two boards then the last channel is 63. This parameter is only used if *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, otherwise is should be passed as NULL. Gx5295: Auxiliary channels numbers are 1000 to 1003.
nLastChannel	SHORT	If *nChannelListMode* parameter is set to DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, then it is specifying the last channel number to apply the settings to. Otherwise it is ignored and should set to zero. Gx5295: Auxiliary channels numbers are 1000 to 1003.
dVoltage	DOUBLE	Sets Tri-State termination voltage. The specified termination voltage is the same voltage that is used by the input active load commutating voltage Voltage range is -2V to +7V.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5295:

The specified termination voltage is the same voltage that is used by the input active load commutating voltage. Channels termination voltage can be read back and set dynamically at any time even while the DIO is running mode.

Example

The following example returns the specified channel Tri-State termination voltage to 2.5V:

```
SHORT anChannelList[]={0, 2, 5, 7, 9, 13, 14, 27};
DioSetupTriStateTerminationVoltage (nHandle, DIO_CH_LIST_MODE_ARRAY_OF_CHANNELS, 8,
anChannelList, 0, 2.5, &nStatus);
```

The following example sets all the board's channels Tri-State termination voltage to 2.5V:

```
DioSetupTriStateTerminationVoltage (nHandle, DIO_CH_LIST_MODE_ALL_BOARD_CHANNELS, 0, NULL , 0,
2.5, &nStatus);
```

The following example sets all the boards' channels Tri-State termination voltage to 2.5V:

```
DioSetupTriStateTerminationVoltage (nHandle, DIO_CH_LIST_MODE_ALL_DOMAIN_CHANNELS, 0, NULL , 0,
2.5, &nStatus);
```

The following example sets channels 5 to 10 Tri-State termination voltage to 2.5V:

```
DioSetupTriStateTerminationVoltage (nHandle, DIO_CH_LIST_MODE_RANGE_OF_CHANNELS, 5, NULL, 10,
2.5, &nStatus);
```

See Also

DioSetupTriStateTerminationVoltage, **DioSetupTriStateTerminationMode, DioSetupInputLoadCurrent, DioSetupInputLoadCommutatingVoltage, DioGetErrorString**

DioStep

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Executes the specified Master board state for given number of steps.

Syntax

DioStep *(nMasterHandle, dwSteps, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
dwSteps	DWORD	Number of steps to execute.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

STEP mode is similar to having a breakpoint in a program: when the program breaks at that line, the step is not yet executed. While issuing a STEP command, the board will pause after *dwSteps* are executed. However, at that point the current step number is not yet executed. Only after issuing a STEP, TRIG or RUN commands will the current step be executed.

The board must be in a PAUSE state or the function returns an error.

Each step is executed by changing the board state from PAUSE to RUN, then back to PAUSE.

This function is useful for debugging purposes.

Example

The following example will step the board through 10 steps and stop:

```
SHORT nStatus;
DioArm (nMasterHandle, &nStatus);
DioStep (nMasterHandle, 10, &nStatus);
```

See Also

DioPause, DioTrig, DioReadProgramCounter, DioSetupTriggerXEventSource

DioTrig

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Triggers the specified Master board and change its state from PAUSE to RUN.

Syntax

DioTrig *(nMasterHandle, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master board handle.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

DioTrig overrides all other trigger conditions previously set. The external trigger control line or the external event lines can also change the board state to RUN.

When the board is in the RUN state, the external pause control line or the P register can be used to switch the state to PAUSE. If a trigger mode is set and the trigger condition is met, calling this function will immediately cause the board to switch to the RUN state.

The board must be in PAUSE mode prior calling this function; otherwise, the function returns an error.

Example

The following example, for a GX5050 board, arms and triggers the board, and waits until the board is in a HALT state:

```
SHORT nStatus;
WORD wData;
DioArm (nMasterHandle, &nStatus);
DioTrig (nMasterHandle, &nStatus);
do
DioReadStatusRegister (nMasterHandle, &wData, &nStatus);
while (wData & 0x1C ==0); // Mask the STATE bits
```

See Also

DioArm, DioHalt, DioReadStatusRegister

DioWriteCtrlCommand

Applies To

GX5280, GX5290, GX5290e, GX5295, File

Purpose

Write the control command parameters from to the specified step.

Syntax

DioWriteCtrlCommand *(nHandle, dwStep, nOpCode, dwParam1, dwParam2, dwParam3, dwParam4, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	DIO or File board handle.
dwStep	DWORD	Vector Step number
nOpCode	SHORT	Operation code
dwParam1	DWORD	Parameter 1 (see comments for details)
dwParam2	DWORD	Parameter 2 (see comments for details)
dwParam3	DWORD	Parameter 3 (see comments for details)
dwParam4	DWORD	Parameter 4 (see comments for details)
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The board must be in a HALT state before calling this function.

GX5281/GX5282:

The Operation code (*nOpCode*) can be one of the following:

Op Code #	Op Code constant	Description
0	DIO_COMMAND_NONE	No Operation code
7	DIO_COMMAND_PAUSE	Pause
8	DIO_COMMAND_HALT	Halt
1000	DIO_COMMAND_CLEAR_ALL	Clear all the control commands.

GX5283/GX5292/GX5293/GX5292e/GX5293e/GX5295:

The board must be in HALT state if the Op Code constant is either DIO_COMMAND_LOOP or DIO_COMMAND_CONTINUOUS_LOOP and the number of steps is less than 256.

The Operation code (*nOpCode*) can be one of the following:

Op Code #	Op Code constant	Description
0	DIO_COMMAND_NONE	No Operation code
3	DIO_COMMAND_LOOP	Loop specified number of times between two predefined steps. dwParam1: Last step dwParam2: Number of loops, min 2 max 4294967295)
7	DIO_COMMAND_PAUSE	Pause
8	DIO_COMMAND_HALT	Halt
11	DIO_COMMAND_CONTINUOUS_LOOP	Loops Continuously between two predefined steps. *dwParam1*: Last step
1000	DIO_COMMAND_CLEAR_ALL	Clear all the control commands.

NOTE: There must be at least 4 steps between PAUSE/HALT and LOOP/Continuous Loop commands.

E.g. having a loop command between steps 10 and 1024, a Pause/Halt command can only be inserted at steps 1026 and above.

When using DIO_COMMAND_LOOP or DIO_COMMAND_CONTINUOUS_LOOP commands the first and last steps are set on 32-bit boundaries, i.e. channels I/O vector data is always on boundaries of 32 bits. As a result when setting the I/O Channels configuration (width) sing **DioSetupIOConfiguration** function the loop command start and last steps need to be set according to the following guidelines:

*I/O channels (width)	First Step	Last Step
32	Any Even step number	Any Odd step number higher than First Ste
16	Any step number that when divided by 2 the integer part of the result is an even number. E.g. 5 is a valid first step number since when dividing by 2 the integer part is even number of 2, but 3 is invalid value since when dividing by 2 the integer part is 1 (not an even number).	Any step number that when divided by 2 the integer part of the result is an odd number. E.g. 1022 is a valid last step number since when dividing by 2 the integer part is an even number of 511, however 1024 is invalid value since when dividing by 2 the integer part is 512 (not an odd number).
8	Any step number that when divided by 4 the integer part of the result is an even number. E.g. 10 is a valid first step number since when dividing by 4 the integer part is even number of 2, but 6 is invalid value since when dividing by 4 the integer part is 1 (not an even number).	Any step number that when divided by 4 the integer part of the result is an odd number. E.g. 2044 is a valid last step number since when dividing by 4 the integer part is an even number of 511, however 2048 is invalid value since when dividing by 4 the integer part is 512 (not an odd number).
4	Any step number that when divided by 8 the integer part of the result is an even number.	Any step number that when divided by 8 the integer part of the result is an odd number.
2	Any step number that when divided by 16 the integer part of the result is an even number.	Any step number that when divided by 16 the integer part of the result is an odd number.
1	Any step number that when divided by 32 the integer part of the result is an even number.	Any step number that when divided by 32 the integer part of the result is an odd number.

Example

The following example inserts PAUSE command to step 100:

```
DioWriteCtrlCommand (nHandle, 100, DIO_COMMAND_PAUSE, NULL, NULL, NULL, NULL, &nStatus);
```

See Also

DioReadCtrlCommand, DioGetNextCtrlCommandStep, DioGetErrorString

DioWriteCtrlMemory

Applies To

GC5050, GX5050, GX5150, File

Purpose

Writes a block of data to the Control memory of the specified Master board.

Syntax

DioWriteCtrlMemory *(nHandle, pvVector, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
pvVector	PVOID	Pointer to an array of type void. The array type depends on the board type as follows: **GX5055:** Array data type is Double Word and maximum array size is 512K of double words. **GC5050, GX5050:** Array data type is Double Word and maximum array size is 1M of double words. **GX5150:** Array type must be BYTE.
dwStart	DWORD	Starting address to write.
dwSize	DWORD	Number of steps to write.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GC5050/GX5050/GX5055: Each step is controlled by one control (DWORD).

GX5150: Each control byte controls eight data bytes in the I/O memory as follows:

- I/O memory width was set to 32-bit - each control byte controls two steps.
- I/O memory width was set to 16-bit - each control byte controls four steps.
- I/O memory width was set to 8-bit - each control byte controls eight steps.

See *Appendix B* for control memory content.

Note: The board must be in HALT state before calling this function.

The program counter will is set to zero upon return.

Example

The following example writes 64 steps to the Master board's Control memory beginning at step 128:

```
DWORD adwBuffer[64];
DioWriteCtrlMemory (nMasterHandle, adwBuffer, 128, 64, &nStatus);
```

See Also

DioReadCtrlMemory, DioWriteInMemory, DioWriteOutMemory, DioWriteIOMemory

DioWriteDirectionMemory

Applies To

GX5055, GX5290, GX5290e, GX5295, File

Purpose

Writes block of direction memory data.

Syntax

DioWriteDirectionMemory *(nHandle, pvVector, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	DIO or File board handle.
pvVector	PVOID	Pointer to an array, array data type needs to comply with the I/O width settings and board type, see comments for details. Each index in the control memory array sets the direction for all the active channels (as was set by the I/O width settings) for the specified step. Each bit corresponds to a channel, i.e. bit 0 sets channel 0 and so on. Direction bits values are as follow: GX5290/GX5290e/GX5295: • Bit low – output. • Bit high – input. GX5055: • Bit low – input. • Bit high – output.
dwStart	DWORD	Starting address.
dwSize	DWORD	Number of steps to write.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5055:

Array data type is Double Word and maximum array size is 512K of double words.

GX5290/GX5290e/GX5295:

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291/GX5291e: Maximum array size is 32M of double words. GX5292/GX5292e: Maximum array size is 64M of double words. GX5293/GX5293e: Maximum array size is 64M of double words. GX5295: Maximum array size is 64M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). GX5291/GX5291e: Maximum array size is 64M words. GX5292/GX5292e: Maximum array size is 128M words. GX5293/GX5293e: Maximum array size is 128M words. GX5295: Maximum array size is 128M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). GX5291/GX5291e: Maximum array size is 128MB. GX5292/GX5292e: Maximum array size is 256MB. GX5293/GX5293e: Maximum array size is 256MB. GX5295: Maximum array size is 256MB.
4, 2, 1	When configuring the Number of channels to be less than 8 bits (4, 2 or 1) the data can be unpacked after reading it using a Double Word array by using DioDataUnpack function. Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291/GX5291e: Maximum array size is 32M of double words. GX5292/GX5292e: Maximum array size is 64M of double words. GX5293/GX5293e: Maximum array size is 64M of double words. GX5295: Maximum array size is 64M of double words.

Note: The board must be in HALT state before calling this function.

Number of channels	Input memory Max Number of steps	Output memory Max Number of steps	Control memory Max Number of steps
32	67108608	67108864	67108864
16	134217216	134217728	134217728
8	268434432	268435456	268435456
4	536868864	536870912	536870912
2	1073737728	1073741824	1073741824
1	2147475456	2147483648	2147483648

Example

The following example write 64 steps from the Master board direction memory beginning at step 128:

```
DWORD adwControl[64];
DioWriteCtrlMemory (nMasterHandle, adwControl, 128, 64, &nStatus);
```

See Also

DioReadDirectionMemory, DioWriteCtrlMemory, DioGetErrorString

DioWriteInMemory

Applies To

GC5050, GX5050, GX5055, GX5290, GX5290e, GX5295, File

Purpose

Writes a block of data to the input memory of the specified board (Master or Slave).

Syntax

DioWriteInMemory *(nHandle, pvVector, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or File board handle.
pvVector	PVOID	Pointer to an array, array data type needs to comply with the I/O width settings and board type, see comments for details.
dwStart	DWORD	Starting address to write.
dwSize	DWORD	Number of steps to write.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The program counter will is set to zero upon return.

GX5050/GX5050:

Array data type is Double Word (DWORD) and maximum array size is 1M of double words.

GX5055:

Array data type is Double Word (DWORD) and maximum array size is 512K of double words.

GX5290/GX5290e:

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291/GX5291e: Maximum array size is 32M of double words. GX5292/GX5292e: Maximum array size is 64M of double words. GX5293/GX5293e: Maximum array size is 64M of double words. GX5295: Maximum array size is 64M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). GX5291/GX5291e: Maximum array size is 64M words. GX5292/GX5292e: Maximum array size is 128M words. GX5293/GX5293e: Maximum array size is 128M words. GX5295: Maximum array size is 128M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). GX5291/GX5291e: Maximum array size is 128MB. GX5292/GX5292e: Maximum array size is 256MB. GX5293/GX5293e: Maximum array size is 256MB. GX5295: Maximum array size is 256MB.
4, 2, 1	When configuring the Number of channels to be less than 8 bits (4, 2 or 1) the data can be unpacked after reading it using a Double Word array by using DioDataUnpack function. Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291/GX5291e: Maximum array size is 32M of double words. GX5292/GX5292e: Maximum array size is 64M of double words. GX5293/GX5293e: Maximum array size is 64M of double words. GX5295: Maximum array size is 64M of double words.

Note: The board must be in HALT state before calling this function.

Number of channels	Input memory Max Number of steps	Output memory Max Number of steps	Control memory Max Number of steps
32	67108608	67108864	67108864
16	134217216	134217728	134217728
8	268434432	268435456	268435456
4	536868864	536870912	536870912
2	1073737728	1073741824	1073741824
1	2147475456	2147483648	2147483648

Example

The following example writes 64 steps to the Master board's output memory beginning at step 128:

```
DWORD adwOutput[64];
DioWriteInMemory (nMasterHandle, adwOutput, 128, 64, &nStatus);
```

See Also

DioWriteCtrlMemory, DioReadCtrlMemory, DioReadInMemory, DioReadOutMemory

DioWriteIOMemory

Applies To

GX5150, GX5280, File

Purpose

Writes a block of data to the specified board (Master or Slave).

Syntax

DioWriteIOMemory *(nHandle, pvMemory, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Master or File board handle.
pvMemory	PVOID	Pointer to an array, array data type needs to comply with the I/O width settings and board type, see comments for details.
dwStart	DWORD	Starting step to write from.
dwSize	DWORD	Number of steps to write.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

The program counter will is set to zero upon return.

GX5150:

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). Maximum array size is 32M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). Maximum array size is 64M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). Maximum array size is 128MB.

GX5280:

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5281: Maximum array size is 32M of double words. GX5282: Maximum array size is 64M of double words. GX5283: Maximum array size is 128M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). GX5281: Maximum array size is 64M words. GX5282: Maximum array size is 128M words. GX5283: Maximum array size is 256M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). GX5281: Maximum array size is 128MB. GX5282: Maximum array size is 256MB. GX5283: Maximum array size is 512MB.
4, 2, 1	When configuring the Number of channels to be less than 8 bits (4, 2 or 1) the data can be unpacked after reading it using a Double Word array by using DioDataUnpack function. Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5281: Maximum array size is 32M of double words. GX5282: Maximum array size is 64M of double words. GX5283: Maximum array size is 128M of double words.

When configuring the Number of channels to be less than 8 bits (4, 2 or 1) the data needs to be packed to 32-bit wide using DioDataPack function.

Note: The board must be in HALT state before calling this function.

Example

The following example reads a block of 64 DWORDs stating at step 128 (width is set to 32-bit):

```
SHORT   nStatus;
DWORD   dwSteps[64];
DioWriteIOMemory(nHandle, dwSteps, 128, 64, &nStatus);
```

See Also

DioReadIOMemory, DioSetupIOConfiguration, DioGetIOConfiguration, DioGetErrorString

DioWriteIOPinsValue

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Write new value to output pin values.

Syntax

DioWriteIOPinsValue *(nHandle, dwValue, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	DIO board handle.
dwValue	DWORD	Data to write to the outputs.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GC5050, GX5050, GX5150, GX5280:

- This function only sets the outputs lines and does not change the loaded vector. In case the direction was changed from output to input while in HALT or PAUSE, output lines are set to Tri-state and writing is disabled.
- A subsequent RUN state will overwrite the I/O value set by this function.
- Since the output values are written only to the output driver, there is no way to read those values back. Calling **DioReadIOPinsValue** following this function will return an error since the board needs to be in input mode. However, input mode bypasses the output drivers.
- The board direction must be set to output prior calling this function.
- The board must be in the HALT or PAUSE state for the user to write to the I/O lines.

GX5055:

- This function only sets the outputs lines and does not change the loaded vector. In case the direction was changed from output to input while in HALT or PAUSE, output lines are set to Tri-state and writing is disabled.
- A subsequent RUN state will overwrite the I/O value set by this function.
- Since the output values are written only to the output driver, there is no way to read those values back. Calling **DioReadIOPinsValue** following this function will return an error since the board needs to be in input mode. However, input mode bypasses the output drivers.
- Each channel's direction can be set independently by calling **DioSetupIOPinsStaticDirection** prior to calling this function.
- The board must be in the HALT or PAUSE state for the user to write to the I/O lines.

GX5290/GX5290e:

- The board may be in HALT or PAUSE state.
- When in this Static I/O mode of writing to the I/O pins data, the channels direction are set according to the **DioSetupChannelsOutputStates**, i.e. bit high is direction is output, bit low direction is input.

GX5295:

- The board may be in a HALT state.
- When in this Static I/O mode of writing to the I/O pins data, the channels direction are set according to the direction memory settings for that specified step number. I.e. bit low, direction is output, bit high direction is input. The user can change the each channel direction in static mode by calling **DioSetupIOPinsStaticDirection** API.
- This function only sets the outputs lines and does not change the loaded vector content. In case the direction was changed from output to input while in HALT or PAUSE, output lines are set to Tri-state and writing is disabled.
- A subsequent ARM command will restore the I/O settings (directions and levels).
- Each channel's direction can be set independently by calling **DioSetupIOPinsStaticDirection** prior to calling this function.

Example

The following example writes 0xAA55AA55 to the I/O pins:

```
SHORT  nStatus;
DioWriteIOPinsValue (nHandle, 0x55AA, &nStatus);
```

See Also

DioReadIOPinsValue, DioSetupIOPinsStaticDirection, DioReadIOPinsValidity, DioGetErrorString

DioWriteOutMemory

Applies To

GC5050, GX5050, GX5055, GX5290, GX5290e, GX5295, File

Purpose

Writes a block of data to the output memory of the specified board (Master or Slave).

Syntax

DioWriteOutMemory *(nHandle, pvMemory, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
pvMemory	PVOID	Pointer to an array, array data type needs to comply with the I/O width settings and board type, see comments for details.
dwStart	DWORD	Starting address to write.
dwSize	DWORD	Number of steps to write.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GX5050/GX5050:

Array data type is Double Word (DWORD) and maximum array size is 1M of double words.

GX5055:

Array data type is Double Word (DWORD) and maximum array size is 512K of double words.

GX5290/GX5290e/GX5295:

Number of channels	Array type
32	Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291/GX5291e: Maximum array size is 32M of double words. GX5292/GX5292e: Maximum array size is 64M of double words. GX5293/GX5293e: Maximum array size is 64M of double words. GX5295: Maximum array size is 64M of double words.
16	Array data type is Word (WORD) when I/O configuration is set to 16-bits width (channels 0-15). GX5291/GX5291e: Maximum array size is 64M words. GX5292/GX5292e: Maximum array size is 128M words. GX5293/GX5293e: Maximum array size is 128M words. GX5295: Maximum array size is 128M words.
8	Array data type is Byte (BYTE) when I/O configuration is set to 8-bits width (channels 0-7). GX5291/GX5291e: Maximum array size is 128MB. GX5292/GX5292e: Maximum array size is 256MB. GX5293/GX5293e: Maximum array size is 256MB. GX5296: Maximum array size is 256MB.
4, 2, 1	When configuring the Number of channels to be less than 8 bits (4, 2 or 1) the data can be unpacked after reading it using a Double Word array by using DioDataUnpack function. Array data type is Double Word (DWORD) when I/O configuration is set to 32-bits width (default). GX5291/GX5291e: Maximum array size is 32M of double words. GX5292/GX5292e: Maximum array size is 64M of double words. GX5293/GX5293e: Maximum array size is 64M of double words. GX5295: Maximum array size is 64M of double words.

Note: The board must be in HALT state before calling this function.

Number of channels	Input memory Max Number of steps	Output memory Max Number of steps	Control memory Max Number of steps
32	67108608	67108864	67108864
16	134217216	134217728	134217728
8	268434432	268435456	268435456
4	536868864	536870912	536870912
2	1073737728	1073741824	1073741824
1	2147475456	2147483648	2147483648

Example

The following example writes 64 steps to the Master board's output memory beginning at step 128:

```
DWORD adwOutput[64];
DioWriteOutMemory (nMasterHandle, adwOutput, 128, 64, &nStatus);
```

See Also

DioWriteCtrlMemory, DioReadCtrlMemory, DioReadInMemory, DioReadOutMemory

DioWriteProgramCounter

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290, GX5290e, GX5295

Purpose

Writes a new value to the program counter.

Syntax

DioWriteProgramCounter *(nMasterHandle, dwCounter, pnStatus)*

Parameters

Name	Type	Comments
nMasterHandle	SHORT	Master or File board handle.
dwCounter	DWORD	Program counter value.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

GC5050/GX5050:

Sets the address of the input, output and control memories. The program counter may be changed only by a control memory command in the RUN state, or by calling a function that resets the program counter to 0, such as **DioReset**, **DioSaveFile**, etc. The control memory commands that change the value of the program counter are JUMP, LOOP, CALL and RETURN.

GX5055:

Sets the address of the input, output, direction, valid data and control memories. The program counter may be changed only by a control memory command in the RUN state, or by calling a function that resets the program counter to 0, such as **DioReset**, **DioSaveFile**, etc. The control memory commands that change the value of the program counter are JUMP, LOOP, CALL and RETURN.

GX5150:

Sets the address of the I/O and control memories. The program counter may be changed only by a control memory command in the RUN state, or by calling a function that resets the program counter to 0, such as **DioReset**, **DioSaveFile**, etc. The control memory commands that change the value of the program counter are JUMP, LOOP, CALL and RETURN.

GX5280/GX5290/GX5290e/GX5295:

GX5280: Sets the address of the I/O memory.

GX5290/GX5290e/GX5295: Sets the address of the I/O and direction memories. The program counter may be changed only by a control memory command in the RUN state, or by calling a function that resets the program counter to 0, such as **DioReset**, **DioSaveFile**, etc.

When the number of active channels is less than 32, the program counter value will be rounded to be on a 32 channels boundaries. E.g. if the I/O configuration was set to 16 channels and the program counter value was set to 1025 the value will be rounded to 1024.

Note: when reading back the program counter value it will correspond to the actual active number of channels as follows:

Number of active channels	Program Counter range
32	GX5281: 0 to 32M GX5282: 0 to 64M GX5283: 0 to 128M GX5291/GX5291e: 0 to 32M GX5292/GX5292e: 0 to 64M GX5293/GX5293e: 0 to 64M GX5295: 0 to 64M
16	GX5281: 0 to 64M GX5282: 0 to 128M GX5283: 0 to 256M GX5291/GX5291e: 0 to 64M GX5292/GX5292e: 0 to 128M GX5293/GX5293e: 0 to 128M GX5295: 0 to 128M
8	GX5281: 0 to 128M GX5282: 0 to 256M GX5283: 0 to 512M GX5291/GX5291e: 0 to 128M GX5292/GX5292e: 0 to 256M GX5293/GX5293e: 0 to 256M GX5295: 0 to 256M
4	GX5281: 0 to 256M GX5282: 0 to 512M GX5283: 0 to 1G GX5291/GX5291e: 0 to 256M GX5292/GX5292e: 0 to 512M GX5293/GX5293e: 0 to 512M GX5295: 0 to 512M
2	GX5281: 0 to 512M GX5282: 0 to 1G GX5283: 0 to 2G GX5291/GX5291e: 0 to 512M GX5292/GX5292e: 0 to 1G GX5293/GX5293e: 0 to 1G GX5295: 0 to 1G
1	GX5281: 0 to 1G GX5282: 0 to 2G

GX5283: 0 to 4G
GX5291/GX5291e: 0 to 1G
GX5292/GX5292e: 0 to 2G
GX5293/GX5293e: 0 to 2G
GX5295: 0 to 2G

Example

The following example writes the value 1024 to the program counter and then verifies the setting by calling **DioReadProgramCounter**:

```
SHORT   nStatus;
DWORD   dwAddress;
DioWriteProgramCounter (nHandle, 1024, &nStatus);
DioReadProgramCounter (nHandle, &dwAddress, &nStatus);
```

See Also

DioReadProgramCounter, DioGetErrorString

DioWriteValidDataMemory

Applies To

GX5055, File

Purpose

Writes a block of data to the valid data memory of the specified board (Master or Slave).

Syntax

DioWriteValidDataMemory *(nHandle, pvMemory, dwStart, dwSize, pnStatus)*

Parameters

Name	Type	Comments
nHandle	SHORT	Board or File board handle.
pvMemory	PVOID	Pointer to an array of double words. Array data type is Double Word and maximum array size is 512K of double words.
dwStart	DWORD	Starting address to write.
dwSize	DWORD	Number of steps to write.
pnStatus	PSHORT	Returned status: 0 on success, negative number on failure.

Comments

Program counter will be set to zero and I/O pin directions are set to input (the output driver is disabled) after calling this function.

The input voltage can be inside one of the following ranges:

- Input voltage is higher than high voltage threshold, input is logic high. Data will be logged as logic high to the input memory and 0 to the invalid logic level input memory.
- Input voltage is lower than low voltage threshold, input is logic low. Data will be logged as logic low to the input memory and 0 to the invalid logic level input memory.
- Input voltage is higher than low voltage threshold and lower then high voltage threshold, input is invalid. Data will be logged as logic low to the input memory and 1 to the invalid logic level input memory.

Note: The board must be in HALT state before calling this function.

Example

The following example writes 64 steps to the Master board's valid data memory beginning at step 128:

```
DWORD adwData[64];
DioWriteValidDataMemory (nHandle, adwData, 128, 64, &nStatus);
```

See Also

DioWriteCtrlMemory, DioReadCtrlMemory, DioReadInMemory, DioReadOutMemory

Chapter 2 - Objects Function Reference

Introduction

GTDIO is supplied with a COM (Component Object Module) object library in addition to the function described in the prior section. The library is used to create and modify DIO files and interface with the DIOEasy application. COM is a Microsoft standard for describing, interfacing and using programmable classes and objects. COM provides a more object oriented programming methodology towards programming then procedural by combining the object code and data to one entity. The COM library describe the classes and their methods, properties and events and additional data types such as enumerated types and structures are defined. COM object classes have methods and properties. Methods are similar to functions calls however they operate on the object which they are called on. Properties are similar to data members and are used to set the object characteristic. Some properties are read only (return only) and cannot be changed, some others can be read and write (returns or set).

The examples provided in this chapter are written in Microsoft Visual Basic version 6.0. Similar code can be used in other languages that support COM including ATEasy, C/C++ and .NET based languages.

Object Classes

The following list the classes available in the GTDIO COM library. The classes are divided to DIO File classes and DIOEasy classes. GTDIO classes are used to create and modify DIO files while the DIOEasy classes are used to control and display documents in DIOEasy.

DIO File Classes:

Class	Description
DioFile	Use to create and modify DIO files.
DioFileBoard	Portion of a DIO File that represents data for a single DIO board in the DIO domain. Each DIO file can support multiple boards.
DioBlock	Collection of channels and their steps data.
DioCommand	Sequencer command; contains operation and condition for the command and extra parameters if required.
DioGroup	Collection of channels grouped under a name.

DIOEasy Classes:

Class	Description
DioApplication	Represents the DIOEasy application.
DioWindows	Collections of DIOEasy opened windows.
DioWindow	A Window inside the DIOEasy main window displaying a document.
DioDocuments	Collection of DIOEasy opened documents.
DioDocument	A DIOEasy opened document.

Properties

Properties are similar to data members of the object. Properties usually do not take any arguments and are used to describe or set the state of an object. All properties return a value, however some properties are read-only, and some are read/write. Here is an example of Visual Basic syntax for getting (property1) and setting a property (property2) of an object:

```
If ob.property1=True then object.property2 = value
```

Methods

Methods are procedures; they can return values and take arguments. They are most often used to execute code on the object. Methods can be used to set values, but only when passing the value through the argument list. Example for calling a method is:

```
Object.Open(“c:\a.txt”)
```

Arguments

Methods and sometimes properties can take one or more arguments, or take none at all. However, arguments might be optional. If they are, you do not have to enter anything for an argument. Once one argument is optional, all arguments following it are also optional. For example, if arguments one and two are required, and three is optional, argument four has to be optional.

Instantiating an Object

To create an instance of a **DioFile** object use a statement like the following example in Visual Basic 6:

```
Set diofile = CreateObject ("GTDio.DioFile")
```

Other programming language may use different syntax to create an object of a class such as new statement.

DioFile Object

Description

The **DioFile** class is used to create, edit, set and get properties and methods of a DIO file object.

Properties

Author	FileVersion	TriggerDEvent
BClokFrequency	Frequency	TriggerDMask
Board	JumpATriggerMode	TriggerMode
Boards	Labels	TriggerPEvent
BoardType	Notes	TriggerTEvent
Channels	OutClockDelay	TriggerTMask
ClkStrobeExternalGateMode	PxiStarTriggerMode	XEventSource
Company	PxiTriggerBusLineMode	XRegister
ExtFrequency	Steps	
FileName		

Methods

Close	DeleteSteps	Save
Command	GetLabelName	SaveAs
CreateBlock	GetLabelStep	SetLabel
CreateGroup	InsertBoard	
DefaultBlock	InsertSteps	
DefultGroup	Label	
DeleteBoard	Open	

Constants

enumDioBoardType

enumDioClkStrobe

enumDioClockStrobeSource

enumDioEventSource

enumDioJumpATriggerMode

enumDioPauseCountMode

enumDioPxiStarTriggerBusMode

enumDioPxiTriggerBusLine

enumDioStep

enumDioTriggerMode

See Also

DioFileBoard, DioBlock, DioCommand, DioGroup

DioFile Object Sample Program

The following example demonstrates how to use DioFile Object as follows:

- Create a new DIO file
- Insert new Board
- Rename channels 0-31 as Data0 - Data31
- Set number of steps to 1024
- Set Clock and Strobe to internal
- Set frequency to 5MHz
- Set Trigger D Event and Mask
- Set Trigger Event and Mask
- Set Trigger Pause Event and Mask
- Create Group of channels to fill
- Create Block of data to filled
- Fill with ramp
- Past the block to the Dio file, stating at step 0 channel 0
- Set the board type to match the file
- Set the command to HALT at step 10 with no condition
- Insert the command at step 10
- Save and close the file

```
Dim diofile As New DioFile
Dim diofileboard As DioFileBoard
Dim dioblock As DioBlock
Dim diogroup As DioGroup
Dim diocommand As New DioCommand

dioFile.Open("DioComExampleFile.dio")
diofile.InsertBoard()
diofileboard = diofile.Board(0)

For i As Integer = 1 To 10
diofileboard.ChannelName(i) = "Data" + Str(i)
Next i

diofile.Steps = 1024
diofile.ClockStrobeSource = enumDioClockStrobeSource.dioClockStrobeInternal
diofile.Frequency = 5000000.0
diofile.TriggerDEvent = &HFF55
diofile.TriggerDMask = &HFFFF
diofile.TriggerTEvent = &HFF55
diofile.TriggerTMask = &HAAAA
diofile.TriggerPEvent = &H1234
diofile.TriggerPMask = &HFFFF
```

```
diogroup = diofile.CreateGroup(0, 31)
dioblock = diofile.CreateBlock(0, 1024, diogroup)
dioblock.FillRamp(0, 1023, 1, 2, 0, 1023, 0, 31)
dioblock.Paste(0, 0)
diocommand.BoardType = diofile.BoardType
diocommand.Set(diofile.BoardType, enumDioCommandOpCode.dioCommandOpCodeHalt, 10,
enumDioCommandCondition.dioCommandConditionNone, 0)
dioblock.Command(10) = diocommand
diofile.Save()
```

Author Property (DioFile)

Applies To

All

Purpose

Sets or returns the DIO file author’s name.

Syntax

Object.Author [= s*Author*]

or

[*sAuthor* =] *Object*.Author

The **Author** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
sAuthor	String	The DIO file authors' name. Default is NULL.

See Also

Company, Notes

BClockFrequency Property (DioFile)

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290

Purpose

Sets or returns the B Clock frequency.

Syntax

Object.*BClockFrequency* [= *BClockFrequency*]

or

[*dwBClockFrequency* =] *Object*.*BClockFrequency*

The **Author** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
dwBClockFrequency	DWORD	The B Clock Frequency value is 1Mhz to 120Mhz in hertz. Default is 5Mhz (5000000).

See Also

Frequency, ExtFrequency

Board Method (DioFile)

Applies To

All

Purpose

Return the **DioBoard** object specified by board number.

Syntax

[*obDioBoard* =] *Object*.Board (*nBoardNumber*)

The **Board** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
nBoardNumber	SHORT	The board numbers can be: 0: Master. 1-7: Slaves.
obDioBoard	DioBoard	**DioBoard** object

Comments

The specified board was added to the **DioFile** object by calling **InsertBoard** method prior calling this function.

See Also

InsertBoard, DeleteBoard, Boards

Boards Property (DioFile)

Applies To

All

Purpose

Returns the total number of file boards in the file.

Syntax

[*iBoards* =] *Object*.Boards

The **Boards** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
iBoards	SHORT	Number of boards in the DIO file.

Where

iBoards can be any number between 1 and 8. E.g. *iBoards* = 3, means 1 Master and 2 Salves.

See Also

BoardType, DeleteBoard, InsertBoard

BoardType Property (DioFile)

Applies To

All

Purpose

Returns board type.

Syntax

[*enDioBoardType* =] *Object*.BoardType

The **BoardType** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
enDioBoardType	enumDioBoardType	Integer specifying the board type.

Where

enDioBoardType can be one of the following:

Name	Value	Description
dioBoardTypeGX5150	0x20	Board type is GX5150
dioBoardTypeGC5050	0x30	Board type is GC5050
dioBoardTypeGX5152	0x40	Board type is GX5152
dioBoardTypeGX5050	0x50	Board type is GX5050
dioBoardTypeGX5055	0x55	Board type is GX5055
dioBoardTypeGX5280	0x60	Board type is GX5280
dioBoardTypeGX5290	0x70	Board type is GX5290
dioBoardTypeGX5290E	0x75	Board type is GX5290E

See Also

DeleteBoard, InsertBoard

Channels Property (DioFile)

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290

Purpose

Returns the total number of channels in the file.

Syntax

[*nChannels* =] *Object*.Channels

The **Channels** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
nChannels	SHORT	Total number of channels.

Comments

GX5050/GC5050: file *nChannels* can be between 32 and 256.

GX5150: file *nChannels* can be between 8 and 256. Use **DioBoard** object to change the number of channels in a specified file board.

See Also

BoardType, DeleteBoard, InsertBoard, Label, Labels

ClkStrobeExternalGateMode Property (DioFile)

Applies To

GX5280, GX5290

Purpose

Sets or returns the Master board Clock or Strobe external gate mode.

Syntax

Object.ClkStrobeExternalGateMode (dioClkStrobe, nMode)

or

[nMode =] *Object*.ClkStrobeExternalGateMode(dioClkStrobe)

The **ClkStrobeExternalGateMode** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
enumDioClkStrobe	dioClkStrobe	Integer specifying either the clock or strobe
nMode	INT	Integer specifying external gate mode: 0. Disabled 1. Enabled.

Where

enumDioClkStrobe can be one of the following:

Name	Value	Description
dioClock	0	Clock
dioStrobe	1	Strobe

Comments

DIO_INTERNAL_CLOCK:

Enables or disables gating the internal Clock of the specified DIO board. For this mode to be active the direction of all the board's channels must be set to output. When the mode is enabled the External Clock Enable input (J3 pin 28) will gate the clock of the specified DIO board. When the External Clock Enable input is set to low the specified DIO board's internal Clock will be active and the DIO will run, when the External Clock Enable input will be high (default state since the input has IS pulled up to VCC). The DIO will be in Pause state.

DIO_INTERNAL_STROBE:

Enables or disables gating the internal Strobe of the specified DIO board. For this mode to be active the direction of all the board's channels must be set to input. When the mode is enabled the External Strobe Enable input (J3 pin 29) will gate the strobe of the specified DIO board. When the External Strobe Enable input is set to low the specified DIO board's internal Strobe will be active and the DIO will run, when the External Strobe Enable input will be high (default state since the input has IS pulled up to VCC). The DIO will be in Pause state

See Also

StrobeDelay

Close Method (DioFile)

Applies To

All

Purpose

Closes DIO file.

Syntax

Object.Close()

The **Close** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object

Comments

The **Close** method will save the current changes to the DIO file.

See Also

Open, Save, SaveAs

Command Method (DioFile)

Applies To

All

Purpose

Set or return command for the specified step.

Syntax

Object.Command (*vStep*, *obCommand*)

Or

[*obCommand* =] Object.Command (*vStep*)

The **Channels** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
vStep	Variant	Step number in the file. Step number specified by either the step label or the step number.
obCommand	DioCommand	**DioCommand** Object.

See Also

DioCommand, DioBoard

Company Property (DioFile)

Applies To

All

Purpose

Returns or sets the company name in the DIO file.

Syntax

Object.Company [= *sCompany*]

Or

[*sCompany* =] *Object*.Company

The **Company** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
sCompany	String	The company name in the **DioFile** object. Default is NULL.

See Also

Author, Notes

CreateBlock Method (DioFile)

Applies To

All

Purpose

Creates and returns reference to a **DioBlock** object.

Syntax

[*obDioBlock* =] *Object*.CreateBlock (*vFileStep, dwSteps,* [*obGroup*])

The **Block** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
vFileStep	Val Variant	Starting step number specified by either the step label or the step number in the file.
dwSteps	DWORD	Number of steps.
obGroup	DioGroup	Optional parameter specifying the number of channels in the new block, can be as follow: Using a predefined group - **DioGroup**. • Default (no parameters passed)– The created block will have as many channels as the file has.
obDioBlock	DioBlock	Reference to **DioBlock** object.

Comments

See the **DioBlock** object for details on the objects' methods and properties.

See Also

DefaultBlock, DefaultGroup

CreateGroup Method (DioFile)

Applies To

All

Purpose

Creates and returns reference to a **DioGroup** object.

Syntax

[*obDioGroup* =] *Object*.CreateBlock ([*vFirstChannel*], [*vLastChannel*])

The **Block** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
vFirstChannel	Val Variant	Optional - Starting channel in the group specified by either the channel number or the step name.
vLastChannel	Val Variant	Optional - Last channel in the group specified by either the channel number or the step name.
obDioGroup	DioGroup	Reference to **DioGroup** object.

Comments

When calling the method with no parameters, all channels of all the boards currently in the File object will be in the **DioGroup** object.

See the **DioGroup** object for details on the objects' methods and properties.

See Also

DefaultGroup, DefaultBlock, CreateBlock

DefaultBlock Method (DioFile)

Applies To

All

Purpose

Creates and returns reference to the default **DioBlock** object.

Syntax

[*obDioBlock* =] *Object*.DefaultBlock

The **DefaultBlock** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
ObDioBlock	DioBlock	Reference to the default **DioBlock** object.

Comments

The default **DioBlock** object contains all the channels and all the steps currently in the **DioFile** object. For example, using this object the user can set all channels and steps to a specific value.

See the **DioBlock** object for details on the objects' methods and properties.

See Also

CreateBlock, DefaultGroup

DefaultGroup Method (DioFile)

Applies To

All

Purpose

Creates and returns reference to the default DioGroup object.

Syntax

[*obDioGroup* =] *Object*.DefaultGroup

The **DefaultGroup** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
ObDioGroup	DioGroup	Reference to the default **DioGroup** object.

Comments

The **DefaultGroup** object contains all the channels currently in the **DioFile** object.

See the **DioGroup** object for details on the objects' methods and properties.

See Also

CreateGroup, DefaultBlock, CreateBlock

DeleteBoard Method (DioFile)

Applies To

All

Purpose

Delete the specified board number.

Syntax

Object.DeleteBoard ()

The **DeleteBoard** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object

Comments

The last board in the file object will be deleted. Up to seven boards can be in each **DioFile** object, the master cannot be deleted.

See Also

InsertBoard, Boards, BoardType

DeleteSteps Method (DioFile)

Applies To

All

Purpose

Delete the specified step's Data and/or Control from all the boards in the DIO file.

Syntax

Object.DeleteSteps (*enDioStep, vFirstStep, dwSteps,* [*vFirstChannel*], [*vLastChannel*])

The **DeleteSteps** Method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
enDioStep	enumDioStep	Integer specifying the operation.
vFirstStep	Val Variant	Start step in the file specified by either the step number or the step name.
dwSteps	DWORD	Number of steps to delete.
vFirstChannel	Variant	Optional - First channel number in the block. Channel number specified by either channel label or the channel number. Default is the first channel.
vLastChannel	Variant	Optional - Last channel number in the block. Channel number specified by either channel label or the channel number. Default is the last channel.

Where

enumDioStep can be one of the following:

Name	Value	Description
dioStepData	0	Deletes selected vector data. Remaining data is moved down the number of steps deleted. Zeros replace data at the high end. Commands and labels remain at the same step. Only data is deleted.
dioStepCommand	1	Deletes selected command steps. Remaining commands are moved down the number of commands deleted. Steps at the end are replaced by NO OPS. Jump references to and from deleted steps are voided.
dioStepDataAndCommand	2	Deletes both data and commands at the selected step(s) and channel(s).

See Also

InsertSteps, Label, Labels, Steps

ExtFrequency Property (DioFile)

Applies To

GC5050, GX5050

Purpose

Returns or sets the external frequency.

Syntax

Object.ExtFrequency [= *dwExtFrequency*]

Or

[*dwExtFrequency* =] *Object*.ExtFrequency

The **ExtFrequency** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
dwExtFrequency	DWORD	External frequency value.

See Also

Frequency, ClockStrobeSource

FileName Property (DioFile)

Applies To

All

Purpose

Returns or sets the DIO file name to be saved as when calling the **Save** or **Close** methods.

Syntax

Object.FileName [= sFileName]

Or

[sFileName =] *Object*.FileName

The **FileName** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
FileName	String	The file name to be saved as.

See Also

Close, Save, SaveAs

FileVersion Property (DioFile)

Applies To

All

Purpose

Return the version of the DIO file format.

Syntax

[*dFileVersion* =] *Object*.FileVersion

The **FileVersion** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
dFileVersion	DOUBLE	DIO file format version.

See Also

BoardType, Notes

Frequency Property (DioFile)

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290

Purpose

Returns or sets the frequency.

Syntax

Object.Frequency [= *dwFrequency*]

Or

[*dwFrequency* =] *Object*.Frequency

The **Frequency** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
dwFrequency	DWORD	Main clock frequency value. Default value is 5000000

See Also

ExtFrequency, BClokFrequency, ClockStrobeSource

GetLabelName Method (DioFile)

Applies To

All

Purpose

Return the label for the specified step.

Syntax

[*sLabel* =] *Object*.GetLabelName (dwStep)

The **GetLabelName** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
DwStep	DWORD	Step number.
SLabel	String	String containing the label at the specified step number.

See Also

GetLabelStep, Label, Labels

GetLabelStep Method (DioFile)

Applies To

All

Purpose

Return the step number where the specified label resides.

Syntax

dwStep = *Object*. GetLabelStep (*sLabel*)

The **GetLabelStep** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
sLabel	String	Label name.
dwStep	DWORD	The step number where the label resides. dwStep will be set to 0xFFFFFFFF in case of invalid label.

See Also

GetLabelName, Label, Labels

InsertBoard Method (DioFile)

Applies To

All

Purpose

Insert a DIO board to the DioFile object.

Syntax

Object.InsertBoard ()

The **InsertBoard** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object

Comments

The new DIO board will become the last board in the file object. Up to seven boards can be added for each DioFile object.

See Also

Boards, BoardType, DeleteBoard

InsertSteps Method (DioFile)

Applies To

All

Purpose

Insert the specified number of steps to all the boards currently in the DIO file.

Syntax

Object.InsertSteps (*enDioStep, vFirstStep, dwSteps*)

The **InsertSteps** Method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
enDioStep	enumDioStep	Integer specifying the operation.
vFirstStep	Val Variant	Start step in the file specified by either the step number or the step name.
dwSteps	DWORD	Number of steps to delete.

Where

enumDioStep can be one of the following:

Name	Value	Description
dioStepData	0	Deletes selected vector data. Remaining data is moved down the number of steps deleted. Zeros replace data at the high end. Commands and labels remain at the same step. Only data is deleted.
dioStepCommand	1	Deletes selected command steps. Remaining commands are moved down the number of commands deleted. Steps at the end are replaced by NO OPS. Jump references to and from deleted steps are voided.
dioStepDataAndCommand	2	Deletes both data and commands at the selected step(s) and channel(s).

See Also

DeleteSteps, Open, Steps

JumpATriggerMode Property (DioFile)

Applies To

GX5150

Purpose

Sets or returns the trigger mode for the external Jump A line.

Syntax

Object.JumpATriggerMode [= *enDioJumpATriggerMode*]

Or

[enDioJumpATriggerMode =]*Object* JumpATriggerMode

The **JumpATriggerMode** Property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
enDioJumpATriggerMode	enumDioJumpATriggerMode	Integers specifying the Jump on register A trigger mode. Default value is 0.

Where

enumDioJumpATriggerMode can be one of the following:

Name	Value	Description
dioJumpATriggerDisable	0	Disable external Jump A.
dioJumpATriggerLowLeve	1	Enable external Jump A to trigger when low level is present.
dioJumpATriggerTransient	2	Enable external Jump A to trigger when low to high transient occurs.

Comments

If the selection is “Enable external”, then the rising edge of the external signal will create the event. If the selection is low to high transient, then a low level will enable the rising edge internal clock to create the event.

This function applies only for “External Jump A” line located on the I/O Control connector.

Label Method (DioFile)

Applies To

All

Purpose

Returns or sets the label in the label array.

Syntax

[*sLabel* =]*Object*.Label (*nIndex*)

Or

Object.Label (dw*Index*, *sLabel*)

The **Label** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
dwIndex	DWORD	Index of the label in the array.
SLabel	String	The DIO file authors' name.

Comments

The label array index is zero based; if the array is empty then index 0 will return NULL.

Getting an index that is out of range then *sLabe* will be NULL. Setting a label with an index that is out of range will add the new element to the array.

The **Labels** property returns the total number of labels in the array.

See Also

Label, Labels, SetLabel

Labels Property (DioFile)

Applies To

All

Purpose

Return the total number of labels.

Syntax

[*dwLabels* =] *Object*.Labels

The **Labels** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
dwLabels	DWORD	Number of labels in the file.

See Also

Label, SetLabel

Notes Property (DioFile)

Applies To

All

Purpose

Returns or sets the notes about the file.

Syntax

Object. Notes [= *sNotes*]

Or

[*sNotes* =] *Object*.Notes

The **Notes** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
sNotes	String	Users' notes about the file.

Comments

This property allows the developer/user of the file to insert text describing the purpose of the file and or any other relevant information.

See Also

Author, Company

Open Method (DioFile)

Applies To

All

Purpose

Creates or opens a DIO file for editing.

Syntax

Object.Open ([*sFileName*], [*enFileMode*], [*enDioBoardType*], [*dwSteps*], [*nChannels*])

The **Open** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
sFileName	String	Optional - File name, default name is “Dio.dio”.
enFileMode	enumFileMode	Optional - Integer specifying the type of access requested for the file. Default mode is **dioFileModeCreate**
enDioBoardType	enumDioBoardType	Optional - Integer specifying the board type. Default board type is **dioBoardTypeGX5050**.
dwSteps	DWORD	Optional - Number of steps in the new file. Default number of steps is 16384 (16K).
nChannels	SHORT	Optional - Number of channels in the new file. Default number of channels is 32
nBoards	SHORT	Optional - Number of boards in the new file, max number of board is 8 (Master and seven Slaves). Default is one board.

Where

enumFileMode can be one of the following:

Name	Value	Description
dioFileModeRead	0	Read only. If the file does not exist or cannot be found the function call fails.
dioFileModeReadWrite	1	Read and Write. The file must exist
dioFileModeCreate	2	Creates a new file. If the file exists, the function overwrites the file and clears the existing attributes.

enDioBoardType can be one of the following:

Name	Value	Description
dioBoardTypeGX5150	0x20	Board type is GX5150
dioBoardTypeGC5050	0x30	Board type is GC5050
dioBoardTypeGX5152	0x40	Board type is GX5152
dioBoardTypeGX5050	0x50	Board type is GX5050
dioBoardTypeGX5055	0x55	Board type is GX5055

dioBoardTypeGX5280	0x60	Board type is GX5280
dioBoardTypeGX5290	0x70	Board type is GX5290
dioBoardTypeGX5290E	0x75	Board type is GX5290E

Comments

The total number of channels and steps follow these guidelines:

GX5050/GC5050:

Number of steps can be between 1024 and 1048576. Number of channels is constant 32.

GX5150:

Number of steps can be between 1024 and 33,554,432 (32M) with number of channels must be 32.
Number of steps can be between 1024 and 67,108,864 (64M) with number of channels must be 16 or 8.
Number of steps can be between 1024 and 134,217,728 (131M) with number of channels must be 8.

Creating a new file creates a DIO file with a single master board. The file will have the default settings as follows (All Boards):

- Frequency: 5MHz
- B Clcck: 5 MHz
- Clock: Internal
- Strobe source: Internal
- StrobeTiming: 10 nSec
- TriggerMode: Disabled
- TriggerMask: 0xFFFF
- DEvent: 0
- DMask: 0xFFFF
- PauseEvent: 0
- PauseMask: 0xFFFF
- XEvent: 0xFFFF
- TriggerEvent:0
- XSource: External

GX5150:

- Direction: Input
- RegisterA: 0
- RegisterB: 0

See Also

InsertBoard, InsertSteps

OutClockDelay Property (DioFile)

Applies To

GX5150

Purpose

Returns or sets the Out Clock delay value.

Syntax

[*dDelay* =] *Object*.OutClockDelay

Or

Object.OutClockDelay [= *dDelay*]

The **Channels** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
dDelay	DOUBLE	Delay values are in nSec, ranges between 0.0 to 64 nSec with increments of 0.25.

See Also

BoardType, StrobeDelay, ClockStrobeSource

PxiStarTriggerMode Property (DioFile)

Applies To

GX5280, GX5290

Purpose

Sets or returns the master board PXI Star Trigger input mode.

Syntax

*Object.*PxiStarTriggerMode (dioPxiStarTriggerBusMode)

or

[dioPxiStarTriggerBusMode =] *Object.* PxiStarTriggerMode

The **PxiStarTriggerMode** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
dioPxiStarTriggerBusMode	enumDioPxiStarTriggerBusMode	PXI Star trigger bus mode

Where

enumDioPxiStarTriggerBusMode can be one of the following:

Name	Value	Description
dioPxiStarTriggerBusModeDisable	0	The DIO star trigger input is disabled.
dioPxiTriggerBusLineModePuase	1	The star trigger input will pause the board
dioPxiStarTriggerBusModeTrigger	2	The DIO star trigger input will trigger the board.

Comments

When the DIO PXI Star Trigger is set to either trigger or pause the board the input signal can be programmed to respond to low level or rising edge. See for details **DioSetupSignalEdgeOrLevelMode**

See Also

PxiTriggerBusLineMode

PxiTriggerBusLineMode Property (DioFile)

Applies To

GX5280, GX5290

Purpose

Sets or returns the master board PXI trigger bus to group output state.

Syntax

*Object.*PxiTriggerBusLineMode (dioTriggerBusLine, dioMode)

or

[nMode =] *Object.*PxiTriggerBusLineMode (dioClkStrobe)

The **PxiTriggerBusLineMode** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
dioTriggerBusLine	enumDioPxiTriggerBusLine	PXI Trigger Bus Lined
dioMode	enumDioPxiTriggerBusLineMode	PXI Trigger Bus Line mode

Where

enumDioPxiTriggerBusLine can be one of the following:

Name	Value	Description
dioPxiTriggerBusLine0	0	PXI Trigger Bus Line 0
dioPxiTriggerBusLine1	1	PXI Trigger Bus Line 1
dioPxiTriggerBusLine2	2	PXI Trigger Bus Line 2
dioPxiTriggerBusLine3	3	PXI Trigger Bus Line 3
dioPxiTriggerBusLine4	4	PXI Trigger Bus Line 4
dioPxiTriggerBusLine5	5	PXI Trigger Bus Line 5
dioPxiTriggerBusLine6	6	PXI Trigger Bus Line 6
dioPxiTriggerBusLine7	7	PXI Trigger Bus Line 7

enumDioPxiTriggerBusLineMode can be one of the following:

Name	Value	Description
dioPxiTriggerBusLineModeDisable	0	The specified PXI Trigger Line is disabled
dioPxiTriggerBusLineModePuaseInput	1	The specified PXI Trigger Line is set to an input pause
dioPxiTriggerBusLineModePuaseOutput	2	The specified PXI Trigger Line is set to as output pause
dioPxiTriggerBusLineModeTriggerInput	3	The specified PXI Trigger Line is set to as input trigger
dioPxiTriggerBusLineModeTriggerOutput	4	The specified PXI Trigger Line is set to as output trigger

Comments

This function allows you to route trigger and paused signals to a specific PXI Trigger Bus line. The trigger and paused signals can be outputs to other Master DIO's or instruments that share the PXI Trigger Bus and/or inputs signals. The Trigger Bus Line assignments are mutually exclusive and can be set in tandem.

See Also

PxiStarTriggerMode

Save Method (DioFile)

Applies To

All

Purpose

Saves the current DIO file object to a file.

Syntax

Object.Save

The **Save** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object

Comments

This method will only work if a valid file name was given to the DIO file by either opening existing file or calling **FileName** property prior to **Save**.

See Also

Open, SaveAs, FileName

SaveAs Method (DioFile)

Applies To

All

Purpose

Save the DIO file object to file.

Syntax

Object.SaveAs (*sFileName*)

The **SaveAs** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
sFileName	String	DIO file name.

See Also

Open, Save

SetLabel Method (DioFile)

Applies To

All

Purpose

Inserts label at the specified step number.

Syntax

Object.SetLabel (*dwStep*, *sLabel)*]

The **SetLabel** method syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
dwStep	DWORD	Step number to insert the label in.
sLabel	String	Label.

See Also

Label, Labels

Steps Property (DioFile)

Applies To

All

Purpose

Return or set the number of steps for all the boards in the **DioFile** object.

Syntax

Object.Steps [= *dwSteps*]

Or

[*dwSteps* =]*Object*.Steps

The Steps property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
dwSteps	DWORD	Number of steps for all the boards in the DioFile object.

Comments

The limit on the number of steps follow these guidelines:

GX5050/GC5050:

- Number of steps can be between 1024 and 1048576. Number of channels is constant 32.

GX5150:

- Number of steps can be between 1024 and 33,554,432 (32M) with number of channels must be 32.
- Number of steps can be between 1024 and 67,108,864 (64M) with number of channels must be 16 or 8.
- Number of steps can be between 1024 and 134,217,728 (131M) with number of channels must be 8.

See Also

BoardType, Channels, StepData

TriggerDEvent Property (DioFile)

TriggerPEvent Property (DioFile)

TriggerTEvent Property (DioFile)

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290

Purpose

Returns or sets the Master board trigger register D/P/T event.

Syntax

Object.DEvent [= *nEvent*]

Object.PEvent [= *nEvent*]

Object.TEvent [= *nEvent*]

Or

[*nEvent* =] *Object*.DEvent

[*nEvent* =] *Object*.PEvent

[*nEvent* =] *Object*.TEvent

The **DEvent/PEvent/TEvent** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
nEvent	SHORT	The Master board trigger register D/P/T event in the **DioFile** object. Default value is 0xFFFF.

See Also

DMask, PMask, TMask, TriggerMode

TriggerDMask Property (DioFile)

TriggerPMask Property (DioFile)

TriggerTMask Property (DioFile)

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290

Purpose

Returns or sets the Master board trigger register D/P/T mask.

Syntax

Object.DMaskt [= *nMask*]

Object.PMask [= *nMask*]

Object.TMask [= *nMask*]

Or

[*nMask* =] *Object*.Mask

[*nMask* =] *Object*.Mask

[*nMask* =] *Object*.Mask

The **DMask/PMask/TMask** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
nMask	SHORT	The Master board trigger register D/P/T mask in the DioFile object. Default value is 0.

See Also

DEvent, PEvent, TEvent, TriggerMode

TriggerMode Property (DioFile)

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290

Purpose

Returns or sets the trigger mode.

Syntax

Object.TriggerMode [= *enDioTriggerMode*]

Or
[*enDioTriggerMode* =] *Object*.TriggerMode

The **TriggerMode** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
enDioTriggerMode	enumDioTriggerMode	Integer specifying the trigger mode. Default value is 0.

Where

enumDioTriggerMode can be one of the following:

Name	Value	Description
dioTriggerDisable	0	Trigger is disabled
dioTriggerDLevel	1	Trigger is waiting for D event.
dioTriggerTLevel	2	Trigger is waiting for T event.
dioTriggerDTLevel	3	Trigger is waiting for D event and then for T event.
dioTriggerTDLevel	4	Trigger is waiting for T event and then for D event.

See Also

ClockStrobeSource, DEvent, DMask, PEvent, PMask, StrobeDelay, Tevent, Mask

XEventSource Property (DioFile)

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290

Purpose

Returns or sets the external event source.

Syntax

Object.ExtEventSource [= *enDioEventSource*]
Or
[*enDioEventSource* =] *Object*.ExtEventSource

The **ExtEventSource** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
enDioEventSource	enumDioEventSource	Integer specifying the external event source value. Default value is 0.

Where

enumDioEventSource can be one of the following:

Name	Value	Description
dioExtEventSource	0	External lines are the source.
dioXRegEventSource	1	X register is external events source

See Also

ClockStrobeSource, XRegister, StrobeDelay, TriggerMode

XRegister Property (DioFile)

Applies To

GC5050, GX5050, GX5150, GX5280, GX5290

Purpose

Returns or sets the external event source.

Syntax

Object.ExtEventSource [= *enDioEventSource*]
Or
[*enDioEventSource* =] *Object*.ExtEventSource

The **ExtEventSource** property syntax has the following parts:

Name	Type	Description
Object	DioFile	**DioFile** object
enDioEventSource	enumDioEventSource	Integer specifying the external event source value. Default value is 0.

Where

enumDioEventSource can be one of the following:

Name	Value	Description
dioExtEventSource	0	External lines are the source.
dioXRegEventSource	1	X register is external events source.

See Also

ClockStrobeSource, XEventSource, StrobeDelay, TriggerMode

DioFileBoard Object

Description

The **DioFile** class is used to create, edit, set and get properties and methods of a DIO file object.

Properties

BoardNumber	InputInterface	InputThresholdVoltageLow
BoardType	InputLoadCurrentSink	OutputDataFormat Property
Channels	InputLoadCurrentSource	OutputSlewRateBias
ChannelsOutputStates	InputLoadCommutatingVoltageHigh	OutputSlewRateFallingEdge
ChannelsVoltageLevel	InputLoadCommutatingVoltageLow	OutputSlewRateRisingEdge
ClkStrobeDelay	InputLoadResistancePulldown	OutputVoltageHigh
Direction	InputLoadResistancePullup	OutputVoltageLow
File	InputThresholdVoltageHigh	

Methods

ChannelName	StepBit
OutputState	StepData

See Also

DioFile, DioBlock, DioCommand, DioGroup

DioFileBoard Object Sample Program

The following example demonstrates how to use DioFileBoard Object as follows:

- Create a new DIO file
- Insert new Board
- Rename channels 0-31 as Data0 - Data31
- Set number of steps to 1024
- Create Group of channels to fill
- Create Block of data to filled
- Fill with ramp
- Past the block to the Dio file, stating at step 0 channel 0
- Set the board type to match the file
- Set the command to HALT at step 10 with no condition
- Insert the command at step 10
- Save and close the file

```
Dim diofile As New DioFile
Dim diofileboard As DioFileBoard
Dim dioblock As DioBlock
Dim diogroup As DioGroup
Dim diocommand As New DioCommand
dioFile.Open("DioComExampleFile.dio")
diofile.InsertBoard()
diofileboard = diofile.Board(0)
For i As Integer = 1 To 10
diofileboard.ChannelName(i) = "Data" + Str(i)
Next i
diofile.Steps = 1024
diogroup = diofile.CreateGroup(0, 31)
dioblock = diofile.CreateBlock(0, 1024, diogroup)
dioblock.FillRamp(0, 1023, 1, 2, 0, 1023, 0, 31)
dioblock.Paste(0, 0)
diocommand.BoardType = diofile.BoardType
diocommand.Set(diofile.BoardType, enumDioCommandOpCode.dioCommandOpCodeHalt, 10,
enumDioCommandCondition.dioCommandConditionNone, 0)
dioblock.Command(10) = diocommand
diofile.Save()
```

BoardNumber Property (DioFileBoard)

Purpose

Return the board index number.

Syntax

[*nBoardNumber* =] *Object*.BoardNumber

The **BoardNumber**property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object.
nBoardNumber	SHORT	Integer specifying the board number in the file object. The index is zero based.

Comments

Boards numbers are between zero and eight. The first board is always the Master with index number zero.

See Also

DeleteBoard, InsertBoard

BoardType Property (DioFileBoard)

Purpose

Returns board type.

Syntax

[*nDioBoardType* =] Object.BoardType

The **BoardType** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nDioBoardType	SHORT	Integer specifying the board type.

Where

enDioBoardType can be one of the following:

Name	Value	Description
dioBoardTypeGX5150	0x20	Board type is GX5150
dioBoardTypeGC5050	0x30	Board type is GC5050
dioBoardTypeGX5152	0x40	Board type is GX5152
dioBoardTypeGX5050	0x50	Board type is GX5050
dioBoardTypeGX5055	0x55	Board type is GX5055
dioBoardTypeGX5280	0x60	Board type is GX5280
dioBoardTypeGX5290	0x70	Board type is GX5290
dioBoardTypeGX5290E	0x75	Board type is GX5290E

See Also

Open

ChannelName Method (DioFileBoard)

Purpose

Set or return the specified channel name.

Syntax

Object.ChannelName (*nChannel, sName*)

Or

[*sName* =] Object.ChannelName (*nChannel*)

The **ChannelName** method syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Channel number can be between 0 and 31.
sName	String	Channel name.

See Also

Channels Property

Channels Property (DioFileBoard)

Purpose

Sets or returns the number of channels in the board.

Syntax

Object.Channels [= *enDioChannels*]

Or

[*enDioChannels* =] Object.Channels

The **Channels** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
enDioChannels	enumDioChannels	Integer specifying the number of channels in the board.

Where

enumDioChannels can be one of the following:

Name	Value	Description
dioChannels8	8	8 - channels
dioChannels16	16	16 - channels
dioChannels32	32	32 - channels

See Also

BoardType

ChannelsOutputStates Property (DioFileBoard)

Applies To

GX5055, GX5280, GX5290.

Purpose

Sets or returns the state of each output channel to be enabled or disabled.

Syntax

Object.ChannelsOutputStates (dwStates)

Or

[dwStates =]*Object*.ChannelsOutputStates

The **ChannelsOutputStates** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
dwStates	DWORD	Each of the 32 bits represents channel's output state. Bit low channel's output disabled, bit high channel's output enabled.

Comments

The function enabled or disabled each channel output. Disabled channels in output mode are in Tri-State. This is useful for connecting to a user bus

See Also

Open

ChannelsVoltageLevel Property (DioFileBoard)

Applies To

GX5280, GX5290.

Sets or returns all channels voltage level.

Syntax

Object.ChannelsVoltageLevel (dVoltage)

Or

[dVoltage =]*Object*.ChannelsVoltageLevel

The **ChannelsVoltageLevel** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
dVoltage	DOUBLE	Channels voltage level in the range of 1.4V to 3.6V with resolution of 10mV.

Comments

The function program the voltage level, applied for all 32 channels, in the TTL Standard connector.

See Also

Open

ClkStrobeDelay Property (DioFileBoard)

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290.

Purpose

Sets or returns clock strobe delay value.

Syntax

Object.ClkStrobeDelay (dVoltage)

Or

[dVoltage =]*Object*.ClkStrobeDelay

The **ClkStrobeDelay** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
dioClockStrobeDelay	enumDioClockStrobeDelay	Integer specifying the Clock/Strobe.
dClockDelay	DOUBLE	Delay values are in nSec GX5150, GC5050, GX5050 GX5055 Out Clock and Strobe: 0.0 to 64 nSec with increments of 0.25. GX5280/GX5290 Main Clock: and Strobe, any of the following values ranges: 0-3 ns with increments of 0.25 ns. 4-7 ns with increments of 0.25 ns. 8-11 ns with increments of 0.25 ns. 12-15 ns with increments of 0.25 ns. 16-19 ns with increments of 0.25 ns. 20-23 ns with increments of 0.25 ns. 24-27 ns with increments of 0.25 ns. If dioClockStrobeDelay = dioOffsetDelay (Gx528X/Gx529X only): -3.0 to +3.0 ns with increments of 0.25 ns.

Where

enumDioClockStrobeDelay can be one of the following:

Name	Value	Description
dioClockDelay	0	Main clock that stimulate the DIO
dioUserClockDelay	1	Out Clock signal (timing connector).
dioStrobeDelay	2	Out Clock signal (timing connector).
dioOffsetDelay	3	Board offset delay

Comments

The function adds a delay between the timing board clock source (i.e. programmable clock) and the following clocks:

NOTE: For the GX5280/GX5290 family when running at frequencies above 100MHz only 0-3nS delay range is available.

See Also

Open

Direction Property (DioFileBoard)

Purpose

Returns or sets all the channels in the **DioFileBoard** object as input or output.

Syntax

Object.Direction [= *enDioDirection*]

Or

[*enDioDirection* =] *Object*.Direction

The **Direction** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
enDioDirection	enumDioDirection	The DIO file authors' name.

Where

e*numDioDirection* can be one of the following:

Name	Value	Description
dioDirectionInput	0	I/O Memory is input only
dioDirectionOutput	1	I/O Memory is output only.

File Property (DioFileBoard)

Purpose

Return a reference to the DioFile object.

Syntax

[*obDioFile* =] *Object*.File

The **File** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object.
obDioFile	DioFile	Reference to **DioFile** object.

Comments

If the **DioFile** object is not declared, then the function generates an exception.

See Also

BoardType

InputInterface Property (DioFileBoard)

Applies To

GX5280, GX5290.

Purpose

Sets or returns input data active interface.

Syntax

Object.InputInterface (dioInterface)

Or

[dioInterface =]*Object*.InputInterface

The **InputInterface** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
dioInterface	enumDioIoInterface	Integer specifying the Input Active Interface.

Where

enumDioIoInterface can be one of the following:

Name	Value	Description
dioIoInterfaceTTL	0	Input TTL connector is the active interface
dioIoInterfaceLVDS	1	Input LVDS connector is the active interface

Comments

This function selects the active input connector LVDS or TTL Standard (3.3V, 2.5V, 1.8V or 1.5V) for each group of channels when in Input mode.

See Also

Open

InputLoadCurrentSink Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns the specified channel input sink load currents.

Syntax

Object.InputLoadCurrentSink (nChannel, dISink)

Or

[dISink =]*Object*.InputLoadCurrentSink(nChannel)

The **InputLoadCurrentSink** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
dISink	DOUBLE	Integer specifying the board type. Input channel constant current sink value. The input channel current sink force the specified constant current to be active when the input voltage is below the low commutating voltage value. Input channel constant current sink value can be set from 0mA to 24mA with 0.3662 μA resolution.

Comments

Each channel has an independent load with the following capabilities:
1. Channels' Input constant source and sink currents up to 24 mA
2. Maintain high impedance over a wide voltage.
3. Separate high and low commutating voltages – Channels' Input Resistive load options

With independent high and low commutating voltages, the source and sink currents each have their own threshold voltage. If the voltage on the input, when the load is activated, is between the two commutating voltages, the load will remain in a high impedance state.

See Also

InputLoadCurrentSource, InputLoadCommutatingVoltageHigh, InputLoadCommutatingVoltageLow, InputLoadResistancePulldown, InputLoadResistancePullup, InputThresholdVoltageHigh, InputThresholdVoltageLow

InputLoadCurrentSource Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns the specified channel input source load currents.

Syntax

Object.InputLoadCurrentSource (nChannel, dISource)

Or

[dISource =]*Object*.InputLoadCurrentSource(nChannel)

The **InputLoadCurrentSource** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
dISource	DOUBLE	Input channel constant current source value. The input channel current source force the specified constant current to be active when the input voltage is above the high commutating voltage value. Input channel constant current source value can be set from 0mA to 24mA with 0.3662 μA resolution.

Comments

Each channel has an independent load with the following capabilities:
0. Channels' Input constant source and sink currents up to 24 mA.
1. Maintain high impedance over a wide voltage.
2. Separate high and low commutating voltages function.
3. Channels' Input Resistive load options.

With independent high and low commutating voltages, the source and sink currents each have their own threshold voltage. If the voltage on the input, when the load is activated, is between the two commutating voltages, the load will remain in a high impedance state.

See Also

InputLoadCurrentSink, InputLoadCommutatingVoltageHigh, InputLoadCommutatingVoltageLow, InputLoadResistancePulldown, InputLoadResistancePullup, InputThresholdVoltageHigh, InputThresholdVoltageLow

InputLoadCommutatingVoltageHigh Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns the specified channel input constant current commutating voltage.

Syntax

Object.InputLoadCommutatingVoltageHigh (nChannel, dVComHi)

Or

[dVComHi =]*Object*.InputLoadCommutatingVoltageHigh(nChannel)

The **InputLoadCommutatingVoltageHigh** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
dVComHi	DOUBLE	Input channel high commutating voltages value, voltage can be set from -16V to +16V.

See Also

InputLoadCurrentSink, InputLoadCurrentSource, InputLoadCommutatingVoltageLow, InputLoadResistancePulldown, InputLoadResistancePullup, InputThresholdVoltageHigh, InputThresholdVoltageLow

InputLoadCommutatingVoltageLow Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns the specified channel input constant current commutating voltage.

Syntax

Object.InputLoadCommutatingVoltageLow (nChannel, dVComLo)

Or

[dVComLo =]*Object*.InputLoadCommutatingVoltageLow(nChannel)

The **InputLoadCommutatingVoltageLow** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
dVComLo	DOUBLE	Input channel low commutating voltages value, voltage can be set from -16V to +16V.

See Also

InputLoadCurrentSink, InputLoadCurrentSource, InputLoadCommutatingVoltageHigh, InputLoadResistancePulldown, InputLoadResistancePullup, InputThresholdVoltageHigh, InputThresholdVoltageLow

InputLoadResistancePulldown Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns the specified channel input pull down resistive loads.

Syntax

Object.InputLoadResistancePulldown (nChannel , dPulldown)

Or

[dPulldown =]*Object*.InputLoadResistancePulldown(nChanne)

The **InputLoadResistancePulldown** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
dPulldown	DOUBLE	Input pull down resistive loads can be one of the following: • 240: 240 Ohms • 290: 240 Ohms • 1000: 1 KOhms • -1: open circuit

See Also

InputLoadCurrentSink, InputLoadCurrentSource, InputLoadCommutatingVoltageHigh, InputLoadCommutatingVoltageLow, InputThresholdVoltageHigh, InputThresholdVoltageLow

InputLoadResistancePullup Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns the specified channel input pull-up resistive load.

Syntax

Object.InputLoadResistancePulldown (nChannel, dPullup)

Or

[dPullup =]*Object*.InputLoadResistancePulldown(nChannel)

The **InputLoadResistancePullup** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
dPullup	DOUBLE	Input pull up resistive loads can be one of the following: • 240: 240 Ohms • 290: 240 Ohms • 1000: 1 KOhms • -1: open circuit

See Also

InputLoadCurrentSink, InputLoadCurrentSource, InputLoadCommutatingVoltageHigh, InputLoadCommutatingVoltageLow, InputLoadResistancePulldown, InputThresholdVoltageHigh, InputThresholdVoltageLow

InputThresholdVoltageHigh Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns the specified channel input high threshold voltages.

Syntax

Object.InputThresholdVoltageHigh (nChannel, dHiLevel)

Or

[dHiLevel =]*Object*.InputThresholdVoltageHigh(nChannel)

The **InputThresholdVoltageHigh** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
dHiLevel	DOUBLE	Input high voltage threshold, value can be -16.0V to +16.0V and higher then Input low voltage threshold.

Comments

The input voltage can be inside one of the following ranges:

0. Input voltage is higher then high voltage threshold, input is logic high. Data will be logged as logic high to the input memory and 0 to the invalid logic level input memory.
1. Input voltage is lower then low voltage threshold, input is logic low. Data will be logged as logic low to the input memory and 0 to the invalid logic level input memory.
2. Input voltage is higher then low voltage threshold and lower then high voltage threshold, input is invalid. Data will be logged as logic low to the input memory and 1 to the invalid logic level input memory.

See Also

InputLoadCurrentSink, InputLoadCurrentSource, InputLoadCommutatingVoltageHigh, InputLoadCommutatingVoltageLow, InputLoadResistancePulldown, InputLoadResistancePullup, InputThresholdVoltageLow

InputThresholdVoltageLow Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns the specified channel input low threshold voltages..

Syntax

Object.InputThresholdVoltageLowChannel, dLoLevel)

Or

[dLoLevel =]*Object*.InputThresholdVoltageLow(nChannel)

The **InputThresholdVoltageLow** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
dLoLevel	DOUBLE	Input low voltage threshold, value can be -16.0V to +16.0V and lower then Input high voltage threshold.

Comments

The input voltage can be inside one of the following ranges:

0. Input voltage is higher then high voltage threshold, input is logic high. Data will be logged as logic high to the input memory and 0 to the invalid logic level input memory.
1. Input voltage is lower then low voltage threshold, input is logic low. Data will be logged as logic low to the input memory and 0 to the invalid logic level input memory.
2. Input voltage is higher then low voltage threshold and lower then high voltage threshold, input is invalid. Data will be logged as logic low to the input memory and 1 to the invalid logic level input memory.

See Also

InputLoadCurrentSink, InputLoadCurrentSource, InputLoadCommutatingVoltageHigh, InputLoadCommutatingVoltageLow, InputLoadResistancePulldown, InputLoadResistancePullup, InputThresholdVoltageHigh

OutputDataFormat Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns the specified channel output data format.

Syntax

Object.OutputDataFormat (dioDataFormat)

Or

[dioDataFormat =]*Object*.OutputDataFormat

The **OutputDataFormat** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
dioDataFormat	enumDioDataFormat	Integer specifying the Output data forma.

Where

enumDioDataFormat can be one of the following:

Name	Value	Description
dioDataFormatNoReturn	0	No Return, the output logic level stay either high or low for the duration of the clock period.
dioDataFormatReturnToZero	1	Return to Zero. The signal returns to zero between consecutive data clocks at 50% of the specified output frequency clock duty cycle. This takes place even if a number of consecutive 0's or 1's occur in the signal. The signal is self clocking and does not require any additional settings.
dioDataFormatReturnToOne	2	Return to One. The signal returns to one between consecutive data clocks at 50% of the specified output frequency clock duty cycle. This takes place even if a number of consecutive 0's or 1's occur in the signal. The signal is self clocking and does not require any additional settings.
dioDataFormatReturnToHiZ	3	Return to Hi-Z .The signal returns to hi-z between consecutive data clocks at 50% of the specified output frequency clock duty cycle. This takes place even if a number of consecutive 0's or 1's occur in the signal. The signal is self clocking and does not require any additional settings.

dioDataFormatReturnToComplement	4	Return to Complement (RC). In Return to Complement (also called Manchester code) each transmitted data bit has at least one transition at 50% of the specified output frequency clock duty cycle. It is, therefore, self-clocking, which means that a clock signal can be recovered from the encoded data. Return to Complement ensures frequent line voltage transitions, directly proportional to the clock rate, this helps clock recovery. Logic low is expressed by a low-to-high transition. Logic high is expressed by high-to-low transition. The transitions which signify logic high or low occur at the midpoint of a period, the direction of the mid-bit transition indicates the data.

Comments

NOTE: the specified channel data format will be applied to all the channels' steps.

See Also

OutputDataFormat, OutputSlewRateBias, OutputSlewRateFallingEdge, OutputSlewRateRisingEdge, OutputVoltageHigh, OutputVoltageLow

OutputSlewRateBias Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns specified channel output slew rate bias.

Syntax

Object.OutputSlewRateBias (nChannel, dBias)

Or

[dBias =]*Object*.OutputSlewRateBias(nChannel)

The **OutputSlewRateBias** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
dBias	DOUBLE	Channel output slew rate bias

Comments

Each channel output has rising and falling slew rate capability with the following characteristics:

- Each output driver has a separate and independent adjustments for the rising and falling slew rate
- Each output driver output stage has a programmable bias current to allow applications that require slower edge rates to consume less power.
- Separate and independent delay circuitry for each channel.

See Also

OutputSlewRateFallingEdge, OutputSlewRateRisingEdge, OutputVoltageHigh, OutputVoltageLow

OutputSlewRateFallingEdge Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns specified channel output falling slew rate.

Syntax

Object.OutputSlewRateFallingEdge (nChannel, dFallingEdge)
Or
[dFallingEdge =]*Object*.OutputSlewRateFallingEdge(nChannel)

The **OutputSlewRateFallingEdge** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
dFallingEdge	DOUBLE	Channel output falling slew rate.

Comments

Each channel output has rising and falling slew rate capability with the following characteristics:

- Each output driver has a separate and independent adjustments for the rising and falling slew rate
- Each output driver output stage has a programmable bias current to allow applications that require slower edge rates to consume less power.
- Separate and independent delay circuitry for each channel.

See Also

OutputDataFormat, OutputSlewRateBias, OutputSlewRateRisingEdge, OutputVoltageHigh, OutputVoltageLow

OutputSlewRateRisingEdge Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns specified channel output rising slew rate.

Syntax

Object.OutputSlewRateRisingEdge (nChannel, dRisingEdge)

Or

[dRisingEdge =]*Object*.OutputSlewRateRisingEdge(nChannel)

The **OutputSlewRateRisingEdge** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
dRisingEdge	DOUBLE	Channel rising slew rate

Comments

Each channel output has rising and falling slew rate capability with the following characteristics:

- Each output driver has a separate and independent adjustments for the rising and falling slew rate
- Each output driver output stage has a programmable bias current to allow applications that require slower edge rates to consume less power.
- Separate and independent delay circuitry for each channel.

See Also

OutputDataFormat, OutputSlewRateBias, OutputSlewRateFallingEdge, OutputVoltageHigh, OutputVoltageLow

OutputState Property (DioFileBoard)

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290

Purpose

Set or return the output pins state.

Syntax

Object.OutputState [= *enDioOutputState*]

Or

[*enOutputState* =] Object.OutputState

The **OutputState** method syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
enDioOutputState	enumDioOutputState	Integer specifying the output pins state. Default is **dioOutputEnable.**

Where

enumDioOutputState can be one of the following:

Name	Value	Description
dioOutputEnable	0	Outputs retain last value on HALT.
dioOutputDisabledOnHalt	1	Outputs are set to Tri-state only when the board is in HALT.
dioOutputDisable	2	Outputs will be set to Tri-state.
dioOutputZero	3	Outputs are set to zero.

Example

The following disable the outputs:

```
diofileboard.OutputState = dioOutputDisable
```

OutputVoltageHigh Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns specified channel output driver high voltages.

Syntax

Object.OutputVoltageHigh (dVoltage)

Or

[dVoltage =]*Object*.OutputVoltageHigh

The **OutputVoltageHigh** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
dHiLevel	DOUBLE	Output driver high voltage corresponding to logic high. Voltage can be set from -16V to +16V and must be greater then the output driver low voltage.

See Also

OutputDataFormat, OutputSlewRateBias, OutputSlewRateFallingEdge, OutputSlewRateRisingEdge, OutputVoltageLow

OutputVoltageLow Property (DioFileBoard)

Applies To

GX5055

Purpose

Sets or returns specified channel output driver low voltages.

Syntax

Object.OutputVoltageLow (dVoltage)

Or

[dVoltage =]*Object*.OutputVoltageLow

The **OutputVoltageLow** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nChannel	SHORT	Specified channel in the DIO board, channel can be 0-31.
dLoLevel	DOUBLE	Output driver low voltage corresponding to logic low. Voltage can be set from -16V to +16V and must be lower then the output driver high voltage.

See Also

OutputDataFormat, OutputSlewRateBias, OutputSlewRateFallingEdge, OutputSlewRateRisingEdge, OutputVoltageHigh

StepBit Method (DioFileBoard)

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290

Purpose

Set or return the specified channel step bit state.

Syntax

Object.StepBit (*vStep*, *iBit*, *enDioPinState*)

Or

[*enDioPinState* =] Object.StepBit (*vStep*, *iBit*)

The **StepBit** method syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
vStep	Variant	Step number in the file. Step number specified by either the step label or the step number.
iBit	INT	Step bit number, 0-31.
enDioPinState	enumDioIgnoreStepPin	Bit state: 0 dioPinStateLow 1 dioPinStateHigh

See Also

StepData, Command

StepData Method (DioFileBoard)

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290

Purpose

Set or return the data of the specified step in the board.

Syntax

Object.StepData (*vStep, dwSepData*)

Or

[*dwData* =] Object. StepData (*vStep*)

The **StepData** method syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
vStep	Variant	Step number in the file. Step number specified by either the step label or the step number.
dwSepData	DWORD	Step Data.

See Also

StepBit, Command

StrobeDelay Property (DioFileBoard)

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290

Purpose

Returns or sets the Strobe Delay value in the DioFileBoard object.

Syntax

Object.StrobeDelay [= *nStrobeDelay*]

Or

[*nStrobeDelay* =]*Object*.StrobeDelay

The **StrobeDelay** property syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
nStrobeDelay	SHORT	Strobe Delay value can be as follow (values are in nSec). GC5050/GX5150: ranges between 0.0 to 64 nSec with increments of 0.25. Default value: 10 nSec.

See Also

ClockStrobeSource, TriggerMode

DioBlock Class

Description

The **DioBlock** object is a user define block of channels and steps. The Block's channels can be defined in any order in the block, i.e. the user can define a block containing channels 0, 2, 10, 4. Then fill or edit those channels and copy their content to the file. This allows the user to concentrate on specific channels and duplicate patterns in the vector. If using the Default Block, the object will contain all the channels and steps that are currently in the block that is the. Since the Block is part of the file, it define a set of steps and channels, any change to data/command in the Block will also change the file. Use the **FileFirstStep** method in order to set the first step in the Block in respect to the file. The **DioBlock** object can only be created from a **DioFile** object

Properties

File	FileFirstStep	Steps

Methods

Command	FillOutputEnable	FillValue
Copy	FillRamp	Paste
Cut	FillRandom	Group
FillClock	FillShift	StepBit
FillDirection	FillToggle	

Constants

enumDioDirection

enumDioFillShiftDirection

enumDioFillShiftType

enumDioFillValueType

enumDioOutputState

enumDioStep

See Also

DioFile, DioFileBoard, DioCommand, DioGroup

DioBlock Object Sample Program

The following example demonstrates how to use DioBlock Object as follows:

- Create a new DIO file
- Set number of steps to 1024
- Set Clock and Strobe to internal
- Create Group of channels to fill
- Create Block of data to filled
- Fill with ramp
- Past the block to the Dio file, stating at step 0 channel 0
- Set the board type to match the file
- Set the command to HALT at step 10 with no condition
- Insert the command at step 10
- Save and close the file

```
Dim diofile As New DioFile
Dim diocommand As New DioCommand
Dim diofileboard As DioFileBoard
Dim dioblock As DioBlock
Dim diogroup As DioGroup

dioFile.Open("DioComExampleFile.dio")
diofile.Steps = 1024
diogroup = diofile.CreateGroup(0, 31)
dioblock = diofile.CreateBlock(0, 1024, diogroup)
dioblock.FillRamp(0, 1023, 1, 2, 0, 1023, 0, 31)
dioblock.Paste(0, 0)
diocommand.BoardType = diofile.BoardType
diocommand.Set(diofile.BoardType, enumDioCommandOpCode.dioCommandOpCodeHalt, 10,
enumDioCommandCondition.dioCommandConditionNone, 0)
dioblock.Command(10) = diocommand
diofile.Save()
```

Command Method (DioBlock)

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290

Purpose

Sets or returns the command to/from specified step.

Syntax

Object.Command (*vFileStep, ObDioCommand*)

Or

[*ObDioCommand* =] Object.Command (*vFileStep*)

The **Command** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
vFileStep	Variant	Starting step number in the file. Step number specified by either step label or step number.
ObDioCommand	DioCommand	**DioCommand** object.

See Also

StepData

Copy Method (DioBlock)

Applies To

All

Purpose

Copy the specified steps Data and/or Control from the DIO block object to be paste in different step.

Syntax

Object.Copy ([*enDioStep*], [*vFirstStep*], [*dwSteps*])

The **Copy** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
enDioStep	enumDioStep	Optional - Integer specifying data type to be cut. Default data type is **dioStepDataAndCommand.**
vFirstStep	Variant	Optional - Starting step number in the block. Step number specified by either the step label or the step number. Default is all the steps on the **DioBlock.**
dwSteps	DWORD	Optional - Number of steps to delete. Default is all the steps in the **DioBlock** after *vFirstStep***.**

Where

enumDioStep can be one of the following:

Name	Value	Description
dioStepData	0	Deletes selected vector data. Remaining data is moved down the number of steps deleted. Zeros replace data at the high end. Commands and labels remain at the same step. Only data is deleted.
dioStepCommand	1	Deletes selected command steps. Remaining commands are moved down the number of commands deleted. Steps at the end are replaced by NO OPS. Jump references to and from deleted steps are voided.
dioStepDataAndCommand	2	Deletes both data and commands at the selected step(s) and channel(s).

Comments

Copying a block to the file object is done in two steps first copy the block to the clipboard set **FileFirstStep** and copy the block to the file object using the **Paste** method.

See Also

Cut, FileFirstStep, Paste

Cut Method (DioBlock)

Applies To

All

Purpose

Cut the specified steps Data and/or Control from the DIO block object.

Syntax

Object.Cut ([*enDioStep*], [*vFirstStep*], [*dwSteps*])

The **Cut** syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
enDioStep	enumDioStep	Optional - Integer specifying data type to be cut. Default data type is **dioStepDataAndCommand.**
vFirstStep	Val Variant	Optional - Starting step number in the block. Step number specified by either the step label or the step number. Default is all the steps on the **DioBlock.**
dwSteps	DWORD	Optional - Number of steps to delete. Default is all the steps in the **DioBlock** after *vFirstStep***.**

Where

enumDioStep can be one of the following:

Name	Value	Description
dioStepData	0	Deletes selected vector data. Remaining data is moved down the number of steps deleted. Zeros replace data at the high end. Commands and labels remain at the same step. Only data is deleted.
dioStepCommand	1	Deletes selected command steps. Remaining commands are moved down the number of commands deleted. Steps at the end are replaced by NO OPS. Jump references to and from deleted steps are voided.
dioStepDataAndCommand	2	Deletes both data and commands at the selected step(s) and channel(s).

Comments

Calling this method will cut the specified step's data and range from the **DioBlock** as well as from the file and post it to the clipboard. The data can be pasted in different location in the file by using the **Paste** method.

See Also

Cut, FileFirstStep, Paste

File Property (DioBlock)

Applies To

All

Purpose

Return reference to the file object

Syntax

obDioFile = Object.File ()

The **File** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
obDioFile	DioFile	**DioFile** object.

Comments

If the **DioFile** object is not declared, then the function generates an exception.

See Also

FileFirstStep, Group, Steps

FileFirstStep Property (DioBlock)

Applies To

All

Purpose

Sets or returns the first step in the **DioBlock** in respect to the file.

Syntax

Object.FirstStep [= *vFileStep*]

Or

[*vFileStep* =] Object.FirstStep

The **FileFirstStep** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
vFileStep	Variant	Starting step number in the file. Step number specified by either the step label or the step number.

Comment

When setting the file first step value, the first step in the **DioBlock object** will point to *vFileStep* step in the file object. When calling the **Paste** method it will paste the block to the file starting form that step.

See Also

File, Group, Steps

FillClock Method (DioBlock)

Applies To

All

Purpose

Fill the specified number of steps and channels with a clock pattern.

Syntax

Object.FillClock *([bStartValue], [dwCycleWidth], [dwInvertStep], [vFirstStep], [dwSteps],* [*vFirstChannel*], [*vLastChannel*])

The **FillClock** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
bStartValue	BOOL	Optional - First step level can be either zero or one. Default is 0.
dwCycleWidth	DWORD	Optional - Cycle Width (in Steps) is the number of steps to complete a clock cycle (must be greater than the Invert after Step entry). Default is 2.
dwInvertStep	DWORD	Optional - Each clock cycle is inverted after a specified number of steps. The Invert after Step. Default is 1.
vFirstStep	Variant	Optional - Starting step number in the block. Step number specified by either the step label or the step number. Default is the first step.
dwSteps	DWORD	Optional - Number of steps. Default is all steps
vFirstChannel	Variant	Optional - First channel number in the block. Channel number specified by either channel label or the channel number. Default is the first channel.
vLastChannel	Variant	Optional - Last channel number in the block. Channel number specified by either channel label or the channel number. Default is the last channel.

See Also

FileFirstStep

FillDirection Method (DioBlock)

Applies To

GC5050, GX5050, GX5055, GX5290

Purpose

Sets the direction for the specified number of steps and channels.

Syntax

Object.FillDirection ([*enDioDirection*], [*vFirstStep*], [*dwSteps*], [*vFirstChannel*], [*vLastChannel*])

The **FillDirection** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
enDioDirection	enumDioDirection	Optional - Integer specifying the DIO direction value. Default is **dioDirectionInput**.
vFirstStep	Variant	Optional - Starting step number in the block. Step number specified by either the step label or the step number. Default is the first step.
dwSteps	DWORD	Optional - Number of steps. Default is all steps
vFirstChannel	Variant	Optional - First channel number in the block. Channel number specified by either channel label or the channel number. Default is the first channel.
vLastChannel	Variant	Optional - Last channel number in the block. Channel number specified by either channel label or the channel number. Default is the last channel.

Where

enumDioDirection can be one of the following:

Name	Value	Description
dioDirectionInput	0	Input
dioDirectionOutput	1	Output.

See Also

FileFirstStep

FillOutputEnable Method (DioBlock)

Applies To

GX5150

Purpose

Set the output state for a range of specified number of steps and channels.

Syntax

Object.FillOutputEnable ([*enDioOutputState*], [*vFirstStep*], [*dwSteps*], [*vFirstChannel*], [*vLastChannel*])

The **FillOutputEnable** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
enDioOutputState	enumDioOutputState	Optional – Integer specifying the out put state. Default is **dioOutputEnable.**
vFirstStep	Variant	Optional - Starting step number in the block. Step number specified by either the step label or the step number. Default is the first step.
dwSteps	DWORD	Optional - Number of steps. Default is all steps
vFirstChannel	Variant	Optional - First channel number in the block. Channel number specified by either channel label or the channel number. Default is the first channel.
vLastChannel	Variant	Optional - Last channel number in the block. Channel number specified by either channel label or the channel number. Default is the last channel.

Where

enumDioOutputState can be one of the following:

Name	Value	Description
dioOutputDisable	0	Disable outputs.
dioOutputEnable	1	Enable outputs.

Comments

Channels are arranged in-group of eight as follow:

- Channels 0-7 are group 0.
- Channels 8-15 are group 1.
- Channels 16-23 are group 2.
- Channels 24-31 are group 3.

Disable/Enable any channel in any group will affect all the channels in that group. E.g. enabling the outputs of channels 0-10 will enable channels 0-15.

All the board direction needs to be set to Output prior calling this function.

See Also

FileFirstStep

FillRamp Method (DioBlock)

Applies To

All

Purpose

Fill the specified number of steps and channels with a Ramp pattern.

Syntax

Object.FillClock ([*dwFirstValue*], [*dwLastValue*], [*dwIncrement*], [*dwCycleWidth*], [*vFirstStep*], [*dwSteps*], [*vFirstChannel*], [*vLastChannel*])

The **FillRamp** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
dwFirstValue	DWORD	Optional - The ramp's starting value. Default is 0.
dwLastValue	DWORD	Optional - The maximum ramp value. Defaults will be last step number in the block.
dwIncrement	DWORD	Optional - Number of counts for each increment, can be negative for down ramp. Default is 1.
dwCycleWidth	DWORD	Optional - Cycle Width (in Steps) is the number of steps to complete a clock cycle (must be greater than the Invert after Step entry). Default is 2.
vFirstStep	Variant	Optional - Starting step number in the block. Step number specified by either the step label or the step number. Default is the first step.
dwSteps	DWORD	Optional - Number of steps. Default is all steps
vFirstChannel	Variant	Optional - First channel number in the block. Channel number specified by either channel label or the channel number. Default is the first channel.
vLastChannel	Variant	Optional - Last channel number in the block. Channel number specified by either channel label or the channel number. Default is the last channel.

Comments

The Ramp can be a up or down ramp, in either cases the following needs to be consider:

Up Ramp, *dwIncrement* is positive: The *dwLastValue* value should exceed dwFirstValue

Down Ramp, *dwIncrement* is negative: The *dwLastValue* value should be less then dwFirstValue.

The following apply to the *dwLastValue* parameter:

- If *dwLastValue* is reached before the last step, the ramp will remain at this value until the last step is reached.
- If the ramp reaches the Last Step before the *dwLastValue* is reached, the ramp ends.
- Channels are assigned to ramp bits starting with the least significant bit. Therefore, if sufficient channels are not allocated to fully represent the (user-modified) *dwLastValue*, only the least significant ramp bits are assigned to channels. However, the ramp is still fully generated internally. Without more significant bits, the ramp will appear to "repeat" with a period that depends on all values entered. The ramp ends if the internal *dwLastValue* value or Last Step is exceeded.

- If there are more channels in the range than are necessary to represent the *dwLastValue* value, the ramp will stop incrementing or decrementing when the *dwLastValue* is reached, or when the Last Step is exceeded. Therefore, higher order channels will not switch.
- The limiting operation of *dwLastValue* value is such that if the next increment would exceed that limit, the ramp value remains unchanged. Thus, if Increment by is set to 5, *dwLastValue* value to 12 and dwFirstValue value to zero, the ramp sequence 0, 5, 10, 10, 10… would be generated. 12, would never be attained because the next value, 15, would exceed the *dwLastValue* value.
- The ratio of *dwIncrement* to the dwCycleWidth value determines the effective slope of the ramp. For example, if *dwIncrement* set to increment by 3, and dwCycleWidth set to 2, the effective slope is 1.5 units per step and increments can occur every second step. If *dwIncrement* set to 6, and dwCycleWidth set to 4, the effective slope is still 1.5, but increments occur every fourth step and the step size is doubled.

See Also

FileFirstStep

FillRandom Method (DioBlock)

Applies To

All

Purpose

Fills the specified number of steps and channels with Pseudo Random values.

Syntax

Object.FillRandom ([i*Seed*], [*vFirstStep*], [*dwSteps*], [*vFirstChannel*], [*vLastChannel*])

The **FillRandom** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
iSeed	Integer	Optional - The seed or starting value, default is 1.
vFirstStep	Variant	Optional - Starting step either number specified by the step label or the step number. Default is the first step.
dwSteps	DWORD	Optional - Number of steps, default is all steps
vFirstChannel	Variant	Optional - First channel number in the block. Channel number specified by either channel label or the channel number. Default is the first channel.
vLastChannel	Variant	Optional - Last channel number in the block. Channel number specified by either channel label or the channel number. Default is the last channel.

See Also

FileFirstStep

FillShift Method (DioBlock)

Applies To

All

Purpose

Shift / Rotate a range of channels and steps.

Syntax

Object.FillShift ([*enDioFillShiftType*], [*enDioFillShiftDirection*], [*dwFirstValue*], [*dwCount*], [*dwCycleWidth*], [*enDioFillValueType*], [*vFirstStep*], [*dwSteps*], [*vFirstChannel*], [*vLastChannel*])

The **FillShift** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
enDioFillShiftType	enumDioFillShiftType	Optional - Integer specifying the shift method, default is **dioFillShiftTypeShift**.
enDioFillShiftDirection	enumDioFillShiftDirection	Optional - Integer specifying the shift direction. Default is **dioFillShiftDirectionLeft**.
dwFirstValue	DWORD	Optional - Starting value Default is 1
dwCount	DWORD	Optional - The number of shifts per step. Default is 1
dwCycleWidth	DWORD	Optional - Cycle Width (in Steps) is the number of steps to complete a clock cycle (must be greater than the Invert after Step entry). Default is 2.
enDioFillValueType	enumDioFillValueType	Optional - Integer specifying the shift value type. This value is valid only when enumDioFillShiftType = 0 (i.e. Shift). Default is **dioFillValueZero**.
vFirstStep	Variant	Optional - Starting step number in the block. Step number specified by either the step label or the step number. Default is the first step.
dwSteps	DWORD	Optional - Number of steps. Default is all steps
vFirstChannel	Variant	Optional - First channel number in the block. Channel number specified by either channel label or the channel number. Default is the first channel.
vLastChannel	Variant	Optional - Last channel number in the block. Channel number specified by either channel label or the channel number. Default is the last channel.

Where

enumDioFillShiftType can be one of the following:

Name	Value	Description
dioFillShiftTypeShift	0	Shift
dioFillShiftTypeRotate	1	Rotate.

enumDioFillShiftDirection can be one of the following:

Name	Value	Description
dioFillShiftDirectionLeft	0	Shift
dioFillShiftDirectionRight	1	Rotate.

enumDioFillValueType can be one of the following:

Name	Value	Description
dioFillValueZero	0	Fill value is Zero.
dioFillValueOne	1	Fill value is One.
dioFillValueIgnore	2	Fill with ignore.
dioFillValueInvert	3	All specified steps values will be inverted.
dioFillValueFirstStep	4	Fill value will be taken from the first step value
dioFillValueCheckerBoard	5	Fill value will alternate between 0xA to 0x 5 for each specified step.
dioFillValueByUser	6	Fill value is specified by the user.

Comments

If *enDioFillShiftType* selected is DioFillShiftTypeRotate, then the value shifted out of the last channel fills the first channel. Then New Cells Value (Shift) is not available.

This command fills the first step of the specified channels with the binary value entered in *dwFirstValue* value. It then shifts the data in each cell left or right according to enumDioFillShiftDirection. The value of the vacated bit position depends on the enumDioFillShiftType and enumDioFillValueType value (Shift only) selection.

If enumDioFillShiftType is DioFillShiftTypeShift, then the value shifted out of the end channel is lost and the value selected in New Cells Value (Shift) fills the beginning channel.

Shift/Rotate (number) *dwCount* specifies the number of shifts per step, i.e. determines the number of steps between shifts.

See Also

FileFirstStep

FillToggle Method (DioBlock)

Applies To

All

Purpose

Fill the specified channels and steps with a square wave.

Syntax

Object.FillToggle ([*bFirstValue*], [*dwToggleStep*], [*vFirstStep*], [*dwSteps*], [*vFirstChannel*], [*vLastChannel*])

The **FilToggle** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
bFirstValue	BOOL	Optional - The first step level can be: Zero or One. Default is One.
dwToggleStep	DWORD	Optional - Toggle interval can be any number. Default is 2 steps.
vFirstStep	Variant	Optional - Starting step number in the block. Step number specified by either the step label or the step number. Default is the first step.
dwSteps	DWORD	Optional - Number of steps. Default is all steps
vFirstChannel	Variant	Optional - First channel number in the block. Channel number specified by either channel label or the channel number. Default is the first channel.
vLastChannel	Variant	Optional - Last channel number in the block. Channel number specified by either channel label or the channel number. Default is the last channel.

See Also

FileFirstStep

FillValue Method (DioBlock)

Applies To

All

Purpose

Fill the specified number of steps and channels with a specific value.

Syntax

Object.FillValue ([*enumDioFillValueType*], [*dwValue*], [*vFirstStep*], [*dwSteps*], [*vFirstChannel*], [*vLastChannel*])

The **FillValue** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
enDioFillValueType	enumDioFillValueType	Optional - Integer specifying the value type. Default is **dioFillValueZero**.
dwValue	DWORD	Optional - Value to insert. Default is 0.
vFirstStep	Variant	Optional - Starting step number in the block. Step number specified by either the step label or the step number. Default is the first step.
dwSteps	DWORD	Optional - Number of steps. Default is all steps
vFirstChannel	Variant	Optional - First channel number in the block. Channel number specified by either channel label or the channel number. Default is the first channel.
vLastChannel	Variant	Optional - Last channel number in the block. Channel number specified by either channel label or the channel number. Default is the last channel.

Where

enumDioFillValueType can be one of the following:

Name	Value	Description
dioFillValueZero	0	Fill value is Zero.
dioFillValueOne	1	Fill value is One.
dioFillValueIgnore	2	Fill with ignore.
dioFillValueInvert	3	All specified steps values will be inverted.
dioFillValueFirstStep	4	Fill value will be taken from the first step value
dioFillValueCheckerBoard	5	Fill value will alternate between 0xA to 0x 5 for each specified step.
dioFillValueByUser	6	Fill value is specified by the user.

See Also

FileFirstStep

Group Method (DioBlock)

Applies To

All

Purpose

Sets or return a group object to the block object.

Syntax

Object.Group (*obGroup*）

Or

[*obGroup* =] Object.Group ()

The **Group** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
ObGroup	DioGroup	**DioGroup** object.

Comments

This property is used to change the current group setting in the block.

See Also

Channels, Copy, Cut, StepData

Paste Method (DioBlock)

Applies To

All

Purpose

Paste the block data from the clipboard to the specified step and channel.

Syntax

Object.Paste (*vFileStartStep, vFileStartChannel*)

The **Paste** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
v*FileStartStep*	Variant	Starting step number in the file. Step number specified by either the step label or the step number.
vFileStartChannel	Variant	Starting channel number in the file. Channel number specified by either the channel name or the channel number.

Comments

Since a block can be viewed as a rectangular, then when pasting we only need to specify the upper left corner of it.

See Also

Copy, Cut, StepData

StepBit Method (DioBlock)

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290

Purpose

Set or return the specified step bit state.

Syntax

Object.StepBit (*vStep*, *iBit*, *enDioPinState*)

Or

[*enDioPinState* =] Object.StepBit (*vStep*, *iBit*)

The **StepBit** method syntax has the following parts:

Name	Type	Description
Object	DioFileBoard	**DioFileBoard** object
vStep	Variant	Step number in the block as was defined by the Group object. Step number specified by either the step label or the step number.
iBit	INT	Step bit number 0 to max number of bits (channels).
enDioPinState	enumDioIgnoreStepPin	Bit state: 0 dioPinStateLow 1 dioPinStateHigh

See Also

StepData

Steps Property (DioBlock)

Applies To

All

Purpose

Sets or returns number of steps in the block.

Syntax

Object.Steps (*dwSteps*)

Or

[*dwSteps* =] Object.Steps()

The **Steps** method syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioBlock** object
dwSteps	DWORD	Number of steps.

See Also

Copy, Cut, StepData

DioCommand Object

Description

The **DioCommand** object is a stand-alone object representing a complete DIO control command. The object methods and properties give access to all individual components of a DIO control command.

Properties

BoardType	OpCode	Value
BranchAdress	Register	
Condition	XEventBit	

Methods

Get	Set

Constants

enumDioBoardType

enumDioCommandCondition

enumDioCommandOpCode

enumDioCommandRegister

enumDioCommandXEventBit

See Also

DioBlock, DioFileBoard, DioFile, DioGroup

DioCommand Object Sample Program

The following example demonstrates how to use DioCommand Object as follows:

- Create a new DIO file
- Set number of steps to 1024
- Create Block of data to filled
- Fill with ramp
- Past the block to the Dio file, stating at step 0 channel 0
- Set the board type to match the file
- Set the command to HALT at step 10 with no condition
- Insert the command at step 10
- Save and close the file

```
Dim diofile As New DioFile
Dim diofileboard As DioFileBoard
Dim dioblock As DioBlock
Dim diogroup As DioGroup
Dim diocommand As New DioCommand

DioFile.Open("DioComExampleFile.dio")
diofile.Steps = 1024
dioblock = diofile.CreateBlock(0, 1024, diogroup)
dioblock.FillRamp(0, 1023, 1, 2, 0, 1023, 0, 31)
dioblock.Paste(0, 0)
diocommand.BoardType = diofile.BoardType
diocommand.Set(diofile.BoardType, enumDioCommandOpCode.dioCommandOpCodeHalt, 10,
enumDioCommandCondition.dioCommandConditionNone, 0)
dioblock.Command(10) = diocommand
diofile.Save()
```

BoardType Property (DioCommand)

Applies To

GC5050, GX5050, GX5055, GX5150, GX5280, GX5290

Purpose

Returns board type.

Syntax

[*nDioBoardType* =] Object.BoardType

The **BoardType** property syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioCommand** object
nDioBoardType	SHORT	Integer specifying the board type.

Where

enDioBoardType can be one of the following:

Name	Value	Description
dioBoardTypeGX5150	0x20	Board type is GX5150
dioBoardTypeGC5050	0x30	Board type is GC5050
dioBoardTypeGX5152	0x40	Board type is GX5152
dioBoardTypeGX5050	0x50	Board type is GX5050
dioBoardTypeGX5055	0x55	Board type is GX5055
dioBoardTypeGX5280	0x60	Board type is GX5280
dioBoardTypeGX5290	0x70	Board type is GX5290
dioBoardTypeGX5290E	0x75	Board type is GX5290E

See Also

Open

BranchAddress Property (DioCommand)

Applies To

GC5050, GX5050, GX5055, GX5150

Purpose

Set or return the command branch address number.

Syntax

Object.BranchAddress [= dwAddress]

or

[dwAddress =] *Object*.BranchAddress

The **BranchAddress** property syntax has the following parts:

Name	Type	Description
Object	DioCommand	**DioCommand** object.
dwAddress	DWORD	Command branch address number.

Comments

GX5050/GC5050/GX5055:

The branch address range is 0 - 131072 (128K) when the **OpCode** is one o the following:

dioOpCodeJumpNear, dioOpCodeLoop, dioOpCodeCall

The branch address range is 0 - 1048576 (1M) when the **OpCode** is

GX5150:

Use **RegisterA** and **RegisterB** in **DioFile** object to set the branch address.

See Also

Register, Condition, Set, Code

Condition Property (DioCommand)

Applies To

GC5050, GX5050, GX5055

Purpose

Set or return the command condition.

Syntax

Object.Condition (*enDioCommandCondition*)

Or

[*enDioCommandCondition* =] *Object*.Condition

The **Condition** property syntax has the following parts:

Name	Type	Description
Object	DioCommand	**DioCommand** object.
enDioCommandCondition	enumDioCommandCondition	Integer specifying the condition.

Where

enumDioCommandCondition can be one of the following:

Name	Value	Description
dioCommandConditionNop	0	No condition
dioCommandConditionGreater	1	External event lines > Register value (GX5050/GC5050).
dioCommandConditionLess	2	External event lines < Register value (GX5050/GC5050).
dioCommandConditionEqual	3	External event lines = Register value.
dioCommandConditionNotEqual	4	External event lines <> Register value.
dioCommandConditionBitLow	5	The specified External event line is low (GX5050/GC5050).
dioCommandConditionBitHigh	6	The specified External event line is high (GX5050/GC5050).

See Also

Register, BoardType, Set, Code

Get Method (DioCommand)

Applies To

All

Purpose

Returns all the properties of the command object.

Syntax

Object.Get (*nCommandOpCode*, *dwAddressOrValue*, *nCommandCondition*, *nCommandRegister*)

Or
Object.Get (*nCommandOpCode*, *dwAddressOrValue*, *nCommandCondition*, *nCommandXEventBit*)

The **Get** method syntax has the following parts:

Name	Type	Description
Object	DioCommand	**DioCommand** object.
nCommandOpCode	VAR SHORT	Integer specifying the command.
dwAddressOrValue	VAR DWORD	The branch address or register value.
nCommandCondition	VAR SHORT	Integer specifying the command condition.
nCommandRegister	VAR SHORT	Optional - Integer specifying the Register number.
nCommandXEventBit	VAR SHORT	Optional - Integer specifying the XEvenBit number.

Comments

See **Set** method for complete details on each parameter.

See Also

Set, Value

OpCode Property (DioCommand)

Applies To

All

Purpose

Set or return the command opcode.

Syntax

Object.OpCode (*enDioCommandOpCode*)

Or

[*enDioCommandOpCode* =] *Object*.OpCode

The **OpCode** property syntax has the following parts:

Name	Type	Description
Object	DioCommand	**DioCommand** object.
enDioCommandOpCode	enumDioCommandOpCode	Integer specifying the opcode

Where

enumDioCommandOpCode can be one of the following:

Name	Value	Description
dioCommandOpCodeNop	0	No operation
dioCommandOpCodeJumpFar	1	Jump Far (GX5050, GX5050, GX5055)
dioCommandOpCodeJumpA	1	Jump Register A (GX5150)
dioCommandOpCodeJumpNear	2	Jump Near(GX5050, GX5050, GX5055)
dioCommandOpCodeJumpB	2	Jump Register B (GX5150)
dioCommandOpCodeLoop	3	Loop (GX5050, GX5050, GX5055)
dioCommandConditionBitHigh	4	Set (GX5050, GX5050, GX5055)
dioCommandOpCodeCall	5	Call (GX5050, GX5050, GX5055)
dioCommandOpCodeReturn	6	Return (GX5050, GX5050, GX5055)
dioCommandOpCodePause	7	Pause
dioCommandOpCodeHalt	8	Halt

See Also

Register, BoardType, Set, Condition

Register Property (DioCommand)

Applies To

GC5050, GX5050, GX5055

Purpose

Set or return the Register value.

Syntax

Object.Register (*dwRegVal*)

Or

[*dwRegVal* =] *Object*.Register

The **Reagister** property syntax has the following parts:

Name	Type	Description
Object	DioCommand	**DioCommand** object.
dwRegVal	DWORD	Register value 0-65365

See Also

BoardType, Set, Get, Condition

Set Method (DioCommand)

Applies To

All

Purpose

Sets all the properties of the command object.

Syntax

Object.Set (*enDioCommandOpCode*, [*vAddressOrValue*], [*enDioCommandCondition*], [*enDioCommandRegister*])

Or

Object.Set (*enDioCommandOpCode*, [*vAddressOrValue*], [*enDioCommandCondition*], [*enDioCommandXEventBit*])

The **Set** method syntax has the following parts:

Name	Type	Description
Object	DioCommand	**DioCommand** object.
enDioCommandOpCode	enumDioCommandOpCode	Integer specifying the command.
vAddressOrValue	Variant	Optional - Set the branch address or register value. Default is 0.
enDioCommandCondition	enumDioCommandCondition	Optional - Integer specifying the command condition. Default is **dioCommandConditionNone.**
enDioCommandRegister	enumDioCommandRegister	Optional - Integer specifying the Register number. Default is **dioCommandRegisterA**.
nDioCommandXEventBit	enumDioCommandXEventBit	Optional - Integer specifying the XEvenBit number. Default is **dioCommandXEventBit0**.

Where

enumDioCommandOpCode can be one of the following:

Name	Value	Description
dioCommandOpCodeNop	0	No operation
dioCommandOpCodeJumpFar	1	Jump Far (GX5050, GX5050, GX5055)
dioCommandOpCodeJumpA	1	Jump Register A (GX5150)
dioCommandOpCodeJumpNear	2	Jump Near (GX5050, GX5050, GX5055)
dioCommandOpCodeJumpB	2	Jump Register B (GX5150)
dioCommandOpCodeLoop	3	Loop (GX5050, GX5050, GX5055)
dioCommandConditionBitHigh	4	Set (GX5050, GX5050, GX5055)
dioCommandOpCodeCall	5	Call (GX5050, GX5050, GX5055)
dioCommandOpCodeReturn	6	Return (GX5050, GX5050, GX5055)
dioCommandOpCodePause	7	Pause
dioCommandOpCodeHalt	8	Halt

enumDioCommandCondition can be one of the following:

Name	Value	Description
dioCommandConditionNone	0	No condition
dioCommandConditionGreater	1	External event lines > Register value(GX5050, GX5050, GX5055).
dioCommandConditionLess	2	External event lines < Register value(GX5050, GX5050, GX5055).
dioCommandConditionEqual	3	External event lines = Register value.
dioCommandConditionNotEqual	4	External event lines <> Register value.
dioCommandConditionBitLow	5	The specified External event line is low(GX5050, GX5050, GX5055).
dioCommandConditionBitHigh	6	The specified External event line is high(GX5050, GX5050, GX5055).

If *enDioCommandOpCode* is **dioCommandOpSet** then *enDioCommandRegister* can be one of the following:

Name	Value	Description
dioCommandRegisterA	0	Register A
dioCommandRegisterB	1	Register B
dioCommandRegisterC	2	Register C
dioCommandRegisterD	3	Register D

If *enDioCommandOpCode* is d**ioCommandConditionBitLow** or d**ioCommandConditionBitHigh** then *enDioCommandXEventBit* can be one of the following:

Name	Value	Description
dioCommandXEventBit0	0	External event line number 0.
dioCommandXEventBit1	1	External event line number 1.
dioCommandXEventBit2	2	External event line number 2.
dioCommandXEventBit3	3	External event line number 3.

enumDioCommandCondition can be one of the following:

Name	Value	Description
dioCommandConditionNone	0	No condition
dioCommandConditionEqual	3	External event lines = Register value.
dioCommandConditionNotEqual	4	External event lines <> Register value.

Comments

The *vAddressOrValue* value can be as follow:

GX5050/GC5050:

If *enDioCommandOpCode* is **dioCommandOpSet** then the *vAddressOrValue* can only be a value 0-65535.

If *enDioCommandOpCode* is **dioCommandOpCodeJumpNear**, **dioCommandOpCodeLoop** or **dioCommandOpCodeCall**, then *vAddressOrValue* can be a step address value or a string representing a label. The branch address range is 0-131072 (128K).

If *enDioCommandOpCode* is **dioCommandOpCodeJumpFar**, then *vAddressOrValue* can be a step address value or a string representing a label. The branch address range is 0-1048576 (1M)

GX5150:

If *enDioCommandOpCode* is **dioCommandOpCodeJumpA** or **dioCommandOpCodeJumpB** then *vAddressOrValue* can be a step address value or a string representing a label. The branch address range is 0-33554432 (32M).

See Also

Get, Value

XEventBit Property (DioCommand)

Applies To

GC5050, GX5050

Purpose

Set or return the command external event bit number.

Syntax

Object.XEventBit [*enDioCommandXEventBit*]

Or

[*enDioCommandXEventBit* =] *Object*.XEventBit

The **XEventBit** property syntax has the following parts:

Name	Type	Description
Object	DioCommand	**DioCommand** object.
enDioCommandXEventBit	enumDioCommandXEventBit	Integer specifying the external event bit number.

Where

enumDioCommandXEventBit can be one of the following (GX5050/GC5050):

Name	Value	Description
dioCommandXEventBit0	0	External event line number 0.
dioCommandXEventBit1	1	External event line number 1.
dioCommandXEventBit2	2	External event line number 2.
dioCommandXEventBit3	3	External event line number 3.

Comments

GX5050/GC5050:

Apply the condition to the specified external events bit number when the condition is set to dioCommandConditionBitLow or dioCommandConditionBitHigh

See Also

Register, BoardType, Set, Get, Code, Condition

Value Property (DioCommand)

Applies To

GC5050, GX5050, GX5055, GX5150

Purpose

Set or return complete command control value.

Syntax

Object.Value [= *dwValue*]

Or

[*dwValue* =] *Object*.Value

The **Value** property syntax has the following parts:

Name	Type	Description
Object	DioCommand	**DioCommand** object.
dwValue	DWORD	Complete command control value.

Comments

GX5050/GC5050: command control value is 32-bit. See "GX5050 User Guide" and "GC5050 User Guide" for details on the command micro code fields.

GX5150: command control value is 8-bit. See "GX515x User Guide" or "GT515x User Guide" for details on the command micro code fields.

See Also

Register, BoardType, Set, Code

DioGroup Object

Description

The **DioGroup** object defines a group of channels specified by name or their logical number in the DIO domain (starting from 0). The channels can be organized in any odder, i.e. different from the order in the file. The group is used by the **DioBlcok** object in order to edit the groups' data. The **DioGroup** object can only be created from a **DioFile** object.

Properties

Count	File	

Methods

Add	ChannelNumber	Delete
ChannelName	Clear	

See Also

DioBlock, DioFileBoard, DioCommand, DioFile

DioGroup Object Sample Program

The following example demonstrates how to use DioGroup Object as follows:

- Create new DIO file.
- Create group with channels 0-18.

```
Dim diofile As New DioFile
Dim dioGroup As DioGroup

diofile.Open ("NewFile.dio")
dioGroup = diofile.CreateGroup (0, 18)
```

Add Method (DioGroup)

Purpose

Adds channel(s) to a group.

Syntax

Object.Add ([*vFirstChannel*], [*vLastChannel*])

The **Add** method syntax has the following parts:

Name	Type	Description
Object	DioGroup	**DioGroup** object.
vFirstChannel	Variant	Optional - First channel to add specified either by name or number.
vLastChannel	Variant	Optional - Last channel to add specified either by name or number.

Comments

If *vFirstChannel* is not specified, all the channels in the file will be added to the group.

If *vLastChannel* was not specified then only single channel (*vFirstChannel*) will be added.

Channels are always added following the last channel in the group.

See Also

Delete, Count, ChannelName

ChannelName Method (DioGroup)

Purpose

Sets or returns the specified channel number name.

Syntax

Object.Channel (*nChannelNumber*, s*ChanneNamel*)

Or

[s*ChannelName* =] *Object*.Channel (*nChannelNumber*)

The **ChannelName** property syntax has the following parts:

Name	Type	Description
Object	DioGroup	**DioGroup** object.
nChannelNumber	SHORT	A zero based index specifying the channel number in the DIO file. Values for vnChannelNumber is 0-255.
s*ChanneNamel*	String	Channel name.

See Also

Add, Delete, Count, ChannelNumber

ChannelNumber Method (DioGroup)

Purpose

Returns the channel number from the specified channel name.

Syntax

[*nChannelNumber* =] *Object*.Channel (s*ChannelName*)

The **ChannelNumber** property syntax has the following parts:

Name	Type	Description
Object	DioGroup	**DioGroup** object.
sChanneNamel	String	Channel name.
nChannelNumber	SHORT	A zero based index specifying the channel number in the DIO file.

Comments

If *nIndex* is out of range then *nChannel* is 0xFFFF.

See Also

Add, Delete, Count, ChannelName

Clear Method (DioGroup)

Purpose

Empty the group from all channels.

Syntax

Object.Clear()

The **Channel** property syntax has the following parts:

Name	Type	Description
Object	DioGroup	**DioGroup** object.

See Also

Add, Delete, Count

Count Property (DioGroup)

Purpose

Returns the total number of channels in the group.

Syntax

[*nCount* =] *Object*.Count

The **Count** property syntax has the following parts:

Name	Type	Description
Object	DioGroup	**DioGroup** object.
nCount	SHORT	Number of channels in the group.

See Also

Add, Delete, Count, ChannelName

Delete Method (DioGroup)

Purpose

Delete range of channels form the group.

Syntax

Object.Delete([*vFirstChannel*], [*vLastChannel*])

The **Delete** method syntax has the following parts:

Name	Type	Description
Object	DioGroup	**DioGroup** object.
vFirstChannel	Variant	Optional - First channel to add specified either by name or number.
vLastChannel	Variant	Optional - Last channel to add specified either by name or number.

Comments

If *vFirstChannel* is not specified, all the channels in the file will be deleted from the group.

If *vLastChannel* was not specified then only single channel (*vFirstChannel*) will be deleted.

See Also

Clear, Delete, Count, ChannelName, ChannelNumkber

File Property (DioGroup)

Purpose

Return reference to the file object

Syntax

obDioFile = Object.File ()

The File property syntax has the following parts:

Name	Type	Description
Object	DioBlock	**DioGroup** object.
obDioFile	DioFile	**DioFile**DioFile_Object object.

Comments

If the **DioFile** object is not declared, then the function generates an exception.

DioApplication Object

Description

The **DioApplication** object controls the DioEasy application. The object controls the location and size of the DioEasy frame, documents (vectors views) position inside the frame and their style (min/max) as well as arranging the vector views. The **DioApplication** object is responsible to create **DioWindows, DioWindow**, **DioDocuments** and **DioDocument** objects.

Properties

Active	ActiveWindow	WindowStyle
ActiveDocument	Visible	

Methods

DioDocuments	GetCoordinatesAndSize
DioWindows	SetCoordinatesAndSize

Constants

enumDioWindowsArrangeStyle

enumDioWinodwStyle

See Also

DioWindows, DioWindow, **DioDocuments, DioDocument**

DioApplication Object Sample Program

The following example demonstrates how to use DioApplication Object as follows:

- Create new DioApplication object.
- Set the application style to be visible.
- Maximized the application.
- Opened a specified DIO file in the DioApplication.

```
Dim dioapp As New DioApplication
dioapp.Visible = True
dioapp.WindowStyle = dioWindowsStateMax
dioapp.DioDocuments.Open("NewFile.dio")
```

ActiveDocument Property (DioApplication)

Purpose

Returns object reference to the currently active document inside the DIOEasy frame.

Syntax

obDioDocument = *Object*.ActiveDocument

The **ActiveDocument** property syntax has the following parts:

Name	Type	Description
Object	DioApplication	**DioApplication** object.
obDioDocument	DioDocument	**DioDocument** object.

Comments

When more than one document is open in the DIOEasy environment, the **ActiveDocument** is the document with the **Error! Objects cannot be created from editing field codes.**focus.

See Also

Arrange, Visible, WindowStyle, DioDocuments

ActiveWindow Property (DioApplication)

Purpose

Returns object reference to the currently active window inside the DIOEasy frame.

Syntax

obDioWindow = *Object*.ActiveDocument

The **ActiveWindow** property syntax has the following parts:

Name	Type	Description
Object	**DioApplication**	**DioApplication** object.
obDioWindow	DioWindow	**DioDocument** object.

Comments

When more than one window is open in the DIOEasy environment, the **ActiveWindow** property is the window with the focus**Error! Objects cannot be created from editing field codes.**. If the main window has the focus, **ActiveWindow** returns **Nothing**.

See Also

Arrange, Visible, WindowStyle

Arrange Property (DioApplication)

Purpose

Sets or returns a value that determines the arregement of windows in a DIOEasy frame.

Syntax

Object.Arrange [= *enDioWindowsArrangeStyle*]

Or

[*enDioWindowsArrangeStyle* =] *Object*.Arrange

The **Arrange** property syntax has the following parts:

Name	Type	Description
Object	DioApplication	**DioApplication** object.
enDioWindowsArrangeStyle	enumDioWindowsArrangeStyle	Integer specifying the arrangement style.

Where

enumDioWindowsArrangeStyle can be one of the following:

Name	Value	Description
dioWindowsArrangeCascade	0	Cascades all non minimized windows.
dioWindowsArrangeTileHorz	1	Tiles all non minimized windows horizontally.
dioWindowsArrangeTileVert	2	Tiles all non minimized windows vertically.

See Also

ActiveWindow, DioWindows

DioDocuments Method (DioApplication)

Purpose

Return reference to the **DioDocuments** object

Syntax

obDioDocuments = Object.DioDocuments ()

The **DioDocuments** property syntax has the following parts:

Name	Type	Description
Object	DioApplication	**DioApplication** object.
obDioDocuments	DioDocuments	**DioDocuments** object.

Comments

If **DioApplication** object is not declared, then the function generates an exception.

See Also

ActiveDocument, DioWindows

DioWindows Method (DioApplication)

Purpose

Return reference to the **DioWindows** object

Syntax

obDioWindows = Object.DioWindows ()

The **DioWindows** property syntax has the following parts:

Name	Type	Description
Object	DioApplication	**DioApplication** object.
obDioWindows	DioWindows	**DioWindows** object.

Comments

If **DioApplication** object is not declared, then the function generates an exception.

See Also

ActiveDocument, Count

GetCoordinatesAndSize Method (DioApplication)

Purpose

Returns the coordinates of the upper-left corner and the size of the DioApplication frame.

Syntax

Object.GetCoordinatesAndSize (*pdwLeft, pdwTop, pdwWidth, pdwHight*)

The **GetCoordinatesAndSize** method syntax has the following parts:

Name	Type	Description
Object	DioApplication	**DioApplication** object.
pdwLeft	PDWORD	Specifies the x-coordinate of the upper-left corner of the frame.
pdwTop	PDWORD	Specifies the y-coordinate of the upper-left corner of the frame.
pdwWidth	PDWORD	Specifies the width of the frame.
pdwHight	PDWORD	Specifies the height of the frame.

See Also

SetCoordinatesAndSize, ActiveDocument, DioWindows

SetCoordinatesAndSize Method (DioApplication)

Purpose

Sets the coordinates of the upper-left corner and the size of the DioApplication frame.

Syntax

Object.SetCoordinatesAndSize (*dwLeft, dwTop, dwWidth, dwHight*)

The S**etCoordinatesAndSize** method syntax has the following parts:

Name	Type	Description
Object	DioApplication	**DioApplication** object.
dwLeft	DWORD	Specifies the x-coordinate of the upper-left corner of the frame.
dwTop	DWORD	Specifies the y-coordinate of the upper-left corner of the frame.
dwWidth	DWORD	Specifies the width of the frame.
dwHight	DWORD	Specifies the height of the frame.

See Also

GetCoordinatesAndSize, ActiveDocument, DioWindows, WindowStyle

Visible Property (DioApplication)

Purpose

Sets or returns if the DioEasy application is be visible or hidden.

Syntax

Object.Visible [= *bVisible*]

Or

[*bVisible* =] Object.Visible

The **Visible** method syntax has the following parts:

Name	Type	Description
Object	DioApplication	**DioApplication** object.
bVisible	`BOOL`	0 Hidden (FALSE). 1 Visible (TRUE).

See Also

ActiveDocument, DioWindows, WindowStyle

WindowStyle Property (DioApplication)

Purpose

Sets the window show style (max/min/normal) for DioEasy frame window

Syntax

Object.WindowState [= *enDioWinodwStyle*]

Or

[*enDioWinodwStyle* =] Object.WindowState

The **WindowStyle** method syntax has the following parts:

Name	Type	Description
Object	DioApplication	**DioApplication** object.
enDioWinodwStyle	enumDioWinodwStyle	Integer specifying the window show style

Where

enumDioWinodwStyle can be one of the following:

Name	Value	Description
dioWindowsStateMax	0	The **DioApplication** is maximized, fills the screen
dioWindowsStateMin	1	The **DioApplication** is minimized to the same size as an icon
dioWindowsStateNormal	2	The **DioApplication** has the same size as it was last opened.

See Also

ActiveDocument, DioWindows

DioWindows Object

Description

The **DioWindows** object controls all windows inside the DioEasy application frame. The object controls the arrangement of the vector views windows, can add a new window and return the number of windows as well as returning a reference to a **Window** specified by its index or caption.

Properties

Arrange	Count

Methods

Add	Window

Constants

enumDioWindowsArrangeStyle

See Also

DioApplication, DioWindow, **DioDocuments, DioDocument**

DioWindows Object Sample Program

The following example demonstrates how to use DioWindows Object as follows:

- Create new DioApplication object.
- Return reference to a **DioWindows** Object.
- Add two new windows.
- Cascade all windows
- Return reference to a **DioWindow** Object specified by index number one.

```
Dim dioapp As New DioApplication
Dim dioWindows As DioWindows
dioWindows.Add
dioWindows.Add
dioWindows.Arrange = dioWindowsArrangeCascade
Set dioWindow = dioWindows.Window (1)
```

Add Method (DioWindows)

Purpose

Adds a new window to the DIOEasy frame.

Syntax

Object.Add ()

The **Add** property syntax has the following parts:

Name	Type	Description
Object	DioWindows	**DioWindows** object.

Comments

The new window will be a copy of the current active window.

See Also

Count

Arrange Property (DioWindows)

Purpose

Sets or returns a value that determines the arrangement in a DIOEasy frame.

Syntax

Object.Arrange [= *enDioWindowsArrangeStyle*]

Or

[*enDioWindowsArrangeStyle* =] *Object*.Arrange

The **Arrange** property syntax has the following parts:

Name	Type	Description
Object	DioWindows	**DioWindows** object.
enDioWindowsArrangeStyle	enumDioWindowsArrangeStyle	Integer specifying the arrangement style.

Where

enumDioWindowsArrangeStyle can be one of the following:

Name	Value	Description
dioWindowsArrangeCascade	0	Cascades all non minimized windows.
dioWindowsArrangeTileHorz	1	Tiles all non minimized windows horizontally.
dioWindowsArrangeTileVert	2	Tiles all non minimized windows vertically.

See Also

Add, Count

Count Property (DioWindows)

Purpose

Returns the number of windows currently in DIOEasy frame.

Syntax

[*nCount* =] *Object*.Count

The **Count** property syntax has the following parts:

Name	Type	Description
Object	DioWindows	**DioWindows** object.
nCount	SHORT	Integer specifying the number of windows currently in DIOEasy frame.

See Also

Add

Window Method (DioWindows)

Purpose

Returns reference to a window object specified by its index or caption.

Syntax

[*obWindow* =] *Object*.Window ([*vCaptionIndex*])

The **Window** property syntax has the following parts:

Name	Type	Description
Object	DioWindows	**DioWindows** object.
vCaptionIndex	Variant	Optional - The Window specified by is index number or caption. Default is the Window with index 0.

See Also

Add, Count, DioWindow

DioWindow Object

Description

The **DioWindow** object controls a window dialog inside the DioEasy application frame. The object can close the window, sets and returns the window style (min/max/normal), setting it to be the active window and return the caption.

Properties

Caption	SetActive	WindowStyle

Methods

Close

Constants

enumDioWindowsArrangeStyle

See Also

DioApplication, DioWindow, **DioDocuments, DioDocument**

DioWindow Object Sample Program

The following example demonstrates how to use DioWindow Object as follows:

- Create new DioApplication object.
- Return reference to a **DioWindows** Object.
- Return reference to a **DioWindow** Object specified by index number one.
- Set the window to be the active window.
- Return the winnow's caption.

```
Dim dioapp As New DioApplication
Dim dioWindows As DioWindows
Dim dioWindow As DioWindow

dioWindows = dioapplication.DioWindows
dioWindow = dioWindows.Window (1)
dioWindow.SetActive
str= dioWindow.Caption
```

Caption Property (DioWindow)

Purpose

Returns the window caption.

Syntax

[*sCaption* =] *Object*.Caption

The **Caption** property syntax has the following parts:

Name	Type	Description
Object	DioWindow	**DioWindow** object.
sCaption	String	The window caption

See Also

Close, SetActive

Close Method (DioWindow)

Purpose

Closes the window.

Syntax

Object.Close

The **Close** property syntax has the following parts:

Name	Type	Description
Object	DioWindow	**DioWindow** object.

See Also

Caption

SetActive Property (DioWindow)

Purpose

Sets the window to be the active window.

Syntax

Object.SetActive

The **SetActive** property syntax has the following parts:

Name	Type	Description
Object	DioWindow	**DioWindow** object.

See Also

Close, Caption

WindowStyle Property (DioWindow)

Purpose

Sets the window show style (max/min/normal) for DioEasy frame window

Syntax

Object.WindowState [= *enDioWinodwStyle*]

Or

[*enDioWinodwStyle* =] Object.WindowState

The **WindowStyle** method syntax has the following parts:

Name	Type	Description
Object	DioWindow	**DioWindow** object.
enDioWinodwStyle	enumDioWinodwStyle	Integer specifying the window show style/

Where

enumDioWinodwStyle can be one of the following:

Name	Value	Description
dioWindowsStateMax	0	The **DioApplication** is maximized, fills the screen
dioWindowsStateMin	1	The **DioApplication** is minimized to the same size as an icon
dioWindowsStateNormal	2	The **DioApplication** has the same size as it was last opened.

See Also

Close, Caption, SetActive

DioDocuments Object

Description

The **DioDocuments** object controls DIO documents in DioEasy application. The object can open new or existing file, close all files, return the number of open files and return reference to a document (file) object.

Properties

Count

Methods

Close Document Open

Constants

enumDioWindowsArrangeStyle

See Also

DioApplication, DioWindows, **DioWindow**, **DioDocument**

DioDocuments Object Sample Program

The following example demonstrates how to use DioDocuments Object as follows:

- Create new DioApplication object.
- Creates two new DIO documents.
- Return a reference to a DioDocument object from the first document.
- Close all documents.

```
Dim dioapp As New DioApplication
Dim dioDocuments As DioDocuments

dioDocuments.Open (dioFileModeCreate, Document1.dio, dioBoardTypeGC5050)
dioDocuments.Open (dioFileModeCreate, Document2.dio, dioBoardTypeGX5150)
dioDocument = dioDocuments.Document ("Document1.dio")
dioDocuments.Close
```

Close Method (DioDocuments)

Purpose

Closes all the documents.

Syntax

Object.Close

The **Close** property syntax has the following parts:

Name	Type	Description
Object	DioDocuments	**DioDocuments** object.

See Also

Count, Open

Count Property (DioDocuments)

Purpose

Returns the number of open documents.

Syntax

[*nCount* =] *Object*.Count

The **Count** property syntax has the following parts:

Name	Type	Description
Object	DioDocuments	**DioDocuments** object.
nCount	SHORT	Integer specifying the number of windows currently in DIOEasy frame.

See Also

Document, Open

Document Method (DioDocuments)

Purpose

Returns a reference to Dio File document object specified by its name.

Syntax

[*obDioDocument* =] *Object*.Document (*sFileName*)

The **Document** property syntax has the following parts:

Name	Type	Description
Object	DioDocuments	**DioDocuments** object.
sFileName	String	Dio file name.
obDioDocument	DioDocument	**DioDocument** object.

See Also

Count, Open

Open Method (DioDocuments)

Purpose

Creates or opens a DIO file for editing in DioEasy application and return a reference to a **DioDocument** object.

Syntax

[*obDocument* =] *Object*.Open ([*enFileMode*], [*sFileName*], [*enDioBoardType*], [*dwSteps*], [*nChannels*])

The **Open** method syntax has the following parts:

Name	Type	Description
Object	DioDocuments	**DioDocuments** object
enFileMode	enumFileMode	Optional - Integer specifying the type of access requested for the file. Default mode is **dioFileModeCreate**
sFileName	String	Optional - File name, default name is “Dio.dio”.
enDioBoardType	enumDioBoardType	Optional - Integer specifying the board type. Default board type is **dioBoardTypeGX5050**.
dwSteps	DWORD	Optional - Number of steps in the new file. Default number of steps is 16384 (16K).
nChannels	SHORT	Optional - Number of channels in the new file. Default number of channels is 32
obDocument	DioDocument	**DioDocument** object

Where

enumFileMode can be one of the following:

Name	Value	Description
dioFileModeRead	0	Read only. If the file does not exist or cannot be found the function call fails.
dioFileModeReadWrite	1	Read and Write. The file must exist
dioFileModeCreate	2	Creates a new file. If the file exists, the function overwrites the file and clears the existing attributes.

enDioBoardType can be one of the following:

Name	Value	Description
dioBoardTypeGX5150	0x20	Board type is GX5150
dioBoardTypeGC5050	0x30	Board type is GC5050
dioBoardTypeGX5152	0x40	Board type is GX5152
dioBoardTypeGX5050	0x50	Board type is GX5050
dioBoardTypeGX5055	0x55	Board type is GX5055
dioBoardTypeGX5280	0x60	Board type is GX5280
dioBoardTypeGX5290	0x70	Board type is GX5290
dioBoardTypeGX5290E	0x75	Board type is GX5290E

Comments

The total number of channels and steps follow these guidelines:

GX5050/GC5050:

Number of steps can be between 1024 and 1048576. Number of channels is constant 32.

GX5150

Number of steps can be between 1024 and 33,554,432 (32M) with number of channels must be 32.

Number of steps can be between 1024 and 67,108,864 (64M) with number of channels must be 16 or 8.

Number of steps can be between 1024 and 134,217,728 (131M) with number of channels must be eight.

See Also

Count, Document

DioDocument Object

Description

The **DioDocument** object controls a DIO document. The object can save or close the document, compare to another document and return the document full path and name.

Properties

Path

Methods

Close	Save
Compare	SaveAs

See Also

DioApplication, DioWindows, **DioWindow**, **DioDocuments**

DioDocument Object Sample Program

The following example demonstrates how to use DioDocument Object as follows:

- Create new DioApplication object.
- Creates new DIO document.
- Saves the document with a different name
- Close the file.

```
Dim dioapp As New DioApplication
Dim dioDocuments As DioDocuments
Dim dioDocument As DioDocument

dioDocuments.Open (dioFileModeCreate, Document1.dio, dioBoardTypeGX5050)
dioDocument = dioDocuments.Documet ("Document1.dio")
dioDocument.Close
```

Close Method (DioDocument)

Purpose

Closes the documents.

Syntax

Object.Close

The **Close** property syntax has the following parts:

Name	Type	Description
Object	DioDocument	**DioDocument** object.

See Also

Path, Save, SaveAs

Compare Method (DioDocument)

Purpose

Compare the document to another documents and display the comparison in DIOEasy.

Syntax

Object.Compare (*sCompareFileName*)

The **Compare** property syntax has the following parts:

Name	Type	Description
Object	DioDocument	**DioDocument** object.
sCompareFileName	String	The document name to compare with.

See Also

Path

Path Property (DioDocument)

Purpose

Returns the full path and file name of the document.

Syntax

[*sFileName* =] *Object*.Path

The **Path** property syntax has the following parts:

Name	Type	Description
Object	DioDocument	**DioDocument** object.
sFileName	String	

See Also

Save, SaveAs

Save Method (DioDocument)

Purpose

Saves the documents.

Syntax

Object.Save

The **Save** property syntax has the following parts:

Name	Type	Description
Object	DioDocument	**DioDocument** object.

See Also

Save, Path

SaveAs Method (DioDocument)

Purpose

Saves the documents with a different name.

Syntax

Object.SaveAs (*sFileName*)

The **SaveAs** property syntax has the following parts:

Name	Type	Description
Object	DioDocument	**DioDocument** object.
sFileName	String	New document name.

See Also

Path

Appendix A - Control Memory Command Microcode

Overview

Information in this section describes control memory command microcode. This is useful when reading and writing from/to the DIO board control memory. See the Commands section in "Theory of Operation" of the respective board user guide for information concerning commands and their parameters.

GC5050, GX5050 Command Microcode

The control memory in a GC5050 and GX5050 contains a 32-bit command for each step. These commands are loaded into the on-board sequencer that controls the flow of the program counter and the direction of I/O channels. Each command is divided into six fields. Table B-1 displays a command layout diagram with the name of each field and bit position. A definition of each command field follows:

Bit #	31 28	27 25	24 22	21 20	19 17	16 0
Field Name	I/O Groups Control	Reserved (0)	Operation Code	Register	Condition	Address

Table A-01: Command Layout

Command Field Descriptions

COMMAND FIELD	DESCRIPTION
I/O Groups Control bits 31-28	These bits control the direction of I/O pin groups, as described in Table A-2.
Reserved, bits 27-25	Reserved for future versions. Must be set to 0.
Operation Code, bits 24-22	These bits provide the command code, as described in Table A-3.
Register, bits 21 and 20	Specifies the internal register used with either the related command or the external event line number (0-3) conditions, as described in Table A-4.
Condition, bits 19-17	This field is the condition code for related commands and tree address MSBs for the JUMP FAR command. Condition codes are described in Table A-5.
Address, bits 16-0	This field contains the address for branch commands: JUMP, LOOP, GOTO and CALL. When the Operations Code is the SET command, this field (bits 0-15) are used as data to be assigned to a register.

Command Code Structure

Table A-2: I/O Groups Control Field

I/O Pins	Control Word Bit	Direction for Bit = 1	Direction for Bit = 0
0-7	28	IN	OUT
8-15	29	IN	OUT
16-23	30	IN	OUT
23-32	31	IN	OUT

Table A-3: Operation Code Field

Mnemonic	Additional fields	Value	Bit 24	Bit 23	Bit 22
NOP	None	0	0	0	0

JUMP FAR	Bits 15 – 0 Destination = Address	1	0	0	1
JUMP NEAR	Condition & Address	2	0	1	0
LOOP	Reg, Address & Condition	3	0	1	1
SET	Reg & Address	4	1	0	0
CALL	Condition & Address*	5	1	0	1
RETURN	Condition	6	1	1	0
PAUSE	Condition	7	1	1	1
HALT	Condition = 7	7	1	1	1

*Address in the CALL is divided by 8.

Table A-4: Register Fields

Register	External Event Line #*B**	Bit 21	Bit 20
A	0	0	0
B	1	0	1
C	2	1	0
D	3	1	1

* See Table B-4

Table A-5: Condition XXX Field

Condition	Value	Bit 19	Bit 18	Bit 17
None	0	0	0	0
External event lines value > Register value	1	0	0	1
External event lines value < Register value	2	0	1	0
External event lines value = Register value	3	0	1	1
External event lines value <> Register value	4	1	0	0
External event line # *B* is Low*	5	1	0	1
External event line # *B* is High*	6	1	1	0
HALT when combined with PAUSE command	7	1	1	1

**B* is determined by the Register field

Coding and Writing Control Commands

When writing commands to a file or to the control memory, the following guidelines should be observed:

1. Commands other than NOP can reside every four steps. Microcode should be the same for all four steps except for I/O Groups Control direction and Operation Code. The Operation Code should be encoded only in the fourth step.
2. The direction can be changed at each step.
3. Control memory and output memory must be written to all boards, Master and Slave.

GX5055 Command Microcode

The control memory in a GX5055 contains a 32-bit command for each step. These commands are loaded into the on-board sequencer that controls the flow of the program counter and the direction of I/O channels. Each command is divided into six fields. Table B-1 displays a command layout diagram with the name of each field and bit position. A definition of each command field follows:

Table A-6: Command Layout

Bit #	32 25	24 22	21 20	19 17	16 0
Field Name	Reserved (0)	Operation Code	Register	Condition	Address

Command Field Descriptions

COMMAND FIELD	DESCRIPTION
Reserved, bits 32-25	Reserved for future versions. Must be set to 0.
Operation Code, bits 24-22	These bits provide the command code, as described in Table B-3.
Register, bits 21 and 20	Specifies the internal register used with either the related command or the external event line number (0-3) conditions, as described in Table B-4.
Condition, bits 19-17	This field is the condition code for related commands and tree address MSBs for the JUMP FAR command. Condition codes are described in Table B-5.
Address, bits 16-0	This field contains the address for branch commands: JUMP, LOOP, GOTO and CALL. When the Operations Code is the SET command, this field (bits 0-15) are used as data to be assigned to a register.

Command Code Structure

Table A-7: Operation Code Field

Mnemonic	Additional fields	Value	Bit 24	Bit 23	Bit 22
NOP	None	0	0	0	0
JUMP FAR	Bits 15 – 0 Destination = Address	1	0	0	1
JUMP NEAR	Condition & Address	2	0	1	0
LOOP	Reg, Address & Condition	3	0	1	1
SET	Reg & Address	4	1	0	0
CALL	Condition & Address*	5	1	0	1
RETURN	Condition	6	1	1	0
PAUSE	Condition	7	1	1	1
HALT	Condition = 7	7	1	1	1

*Address in the CALL is divided by 8.

Table A-0-8: Register Fields

Register	External Event Line #*B**	Bit 21	Bit 20
A	0	0	0
B	1	0	1
C	2	1	0

* See Table B-4

Table A-9: Condition XXX Field

Condition	Value	Bit 19	Bit 18	Bit 17
None	0	0	0	0
HALT when combined with PAUSE command	7	1	1	1

**B* is determined by the Register field

Coding and Writing Control Commands

When writing commands to a file or to the control memory, the following guidelines should be observed:

1. Commands other than NOP can reside every four steps. Microcode should be the same for all four steps except for I/O Groups Control direction and Operation Code. The Operation Code should be encoded only in the fourth step.
2. The direction can be changed at each step.
3. Control memory and output memory must be written to all boards, Master and Slave.

GX5150 Command Microcode

The control memory in a GX515X contains an 8-bit command for each step. These commands are loaded into the on-board sequencer that controls the flow of the program counter and the direction of I/O channels. See the Commands section in "Theory of Operation" of the respective board user guide for information concerning commands and their parameters.

Each location in Control Memory consists of an 8-bit wide 'Command Word'. Figure B-6 shows the bit fields in each Command Word:

Table A-10: GX5150 Command Word

Bit #	7	6	5	4	3 2	1 0
Command Word	OE3	OE2	OE1	OE0	CND	Op Code

Op Code	Bit 0 and Bit 1 define the Operation code field.
CND	Bit 2 and Bit 3 define the Conditions field.
OE*n*	Bits 4 through bit 7 are the output enables for group n (where n = 0-3). These are valid only when the GX5150 is in output mode. This feature is not used when programmed for input mode. A *logic low* level enables the corresponding output driver.

Op Codes

Table A-11: defines bits in the Command Word Op Code field.

Mnemonic	**Code**	**Bit 1**	**Bit 0**
NOP	0	0	0
Jump A	1	0	1
Jump B	2	1	0
Pause/HALT	3	1	1

Condition Codes

Table A-12 defines the bit designations for the condition (CND) field of the Command Word. The conditions define the trigger event that executes the desired jump to a predetermined address location in vector memory.

Mnemonic	**Code**	**Bit 1**	**Bit 0**
None	0	0	0
Equal	1	0	1
Not Equal	2	1	0
Halt	3	1	1

Appendix B - Error Codes

The following list identifies error codes by category. Codes may apply universally or to only a particular board type.

Error Value	Cause/Symptom	Solution
RESOURCE ERRORS:		
0	No error	
-2000	Unable to allocate the necessary memory.	There is not enough memory for the DIO Driver to operate. Close other applications to free up more memory.
-2001	Too many open files.	More than eight DIO files are open. Close one or more as needed.
-2002	The specified file name could not be found in the given path.	Check the file name, path and access rights.
-2004	The file could not be opened for Read-Only since its attributes are not set properly.	Check file properties, one or more of the following file properties are not set correctly: The file attribute should be Read-Only. The file attribute should not be Hidden or System.
-2005	The file could not be opened for read/write since its attributes are not set properly.	Check file properties, one or more of the following file properties are not set correctly: The file attribute should not be Read-Only, Hidden or System.
-2006	Source and destination paths and file names are the same.	Change the destination path or file name.
-2007	One or more of the specified file paths are invalid.	Change the file path.
-2009	Unable to delete temporary DIO file.	The driver was unable to delete the temporary DIO file created using ***DIOEasy*** 1.x. Delete the temporary file, named "~temp.dio", that resides in the same directory as the problem file.
-2020	A board handle is specified, but a DIO File board handle is required.	The driver function can only accept a File board handle. Use another function or use a DIO File board handle.
-2021	Unable to invoke WinHelp.	"Windows Help" could not be invoked. Check for the existence of WinHelp.exe in the Windows or WinNT folder, as appropriate for the operating system.
-2022	The string is too short.	Increase the string length or set it to 255.
-2026	Valid configuration must be set before initializing this master.	Use the front panel to create new configuration file.
-2028	The driver was unable to find Windows directory.	
-2029	Invalid OS version.	

Error Value	Cause/Symptom	Solution
PARAMETER ERRORS:		
-3000	Invalid parameter	A parameter passed to the specified function is invalid. The cause may be one of the following: Wrong type (for example, float instead of long). Out of range (for example, valid values are 0 or 1 and 2 is passed). A NULL pointer is assigned instead of a valid one.
-3001	The Master board number is out of range.	Master indices range from 0 to 15 (16 allowed).
-3002	A domain Slave board number is out of range.	Slave numbers range from 1 to 7. Number 0 is always reserved for the Master board.
-3003	The number of memory module banks is out of range.	Total installed memory banks range from 1 to 4.
-3004	The memory module density is out of range.	Memory density ranges from 0 to 2.
-3005	Invalid base address.	Valid base addresses range from 0x100 to 0xFFFF.
-3006	Invalid board or File board handle.	The handle passed to a function is invalid. Use the “DioGetSlaveHandle” or “DioFileGetBoardHandle” function to retrieve the correct handle.
-3007	This function is not supported for the specified board.	A called function is not supported for the specified board. See the Function Reference section in your Software User Guide and Function Reference manual to determine the supported functions.
-3008	Invalid file type.	The given file could not be identified. Check if the file is corrupted or needs to be converted.
-3009	Invalid board handle.	The given handle is not a valid board handle. Use “DioGetHandle” to a valid handle for he specified board number.
-3010	Parameter is out of range.	
-3011	Invalid Interface Standard Level value.	
-3012	Specified External Frequency is out of the allowed range	
-3014	Unable to detect the 3.0 reference voltage.	
-3015	Unable to detect the 3.3 voltage supply.	
-3016	Unable to detect positive voltage supply.	
-3017	Unable to detect negative voltage supply.	
-3100	Invalid calibration mode	

Error Value	Cause/Symptom	Solution
-3101	Bad or Wrong board EEPROM	
MAIN BOARD ERRORS:		
-4000	A board type could not be detected at the given base address	Check the DIP Switch base address setting for the DIO board. Make certain the board is configured with that base address specified and that this does not conflict with any other board's base address. Change the base address to resolve conflicts. See the appropriate hardware user guide for the specified board DIP Switch setting. If the board is installed in a GTXI chassis, then check that the specified base address is enabled (using GTXI.CFG). Check that the carrier board DIP Switch is set correctly.
-4001	This function requires a Master board handle.	The function was called with a Slave handle instead of a Master handle. See the Function Reference section in the Software User Guide and Function Reference to find out which functions apply to a Master and which to a Slave.
-4002	Invalid attempt to initialize a Master board as a Slave.	A Master board cannot be initialized or used as a Slave. Check the base address and the DIP Switch settings of the DIO board.
-4003	A specified Slave cannot be associated with the specified Master handle.	Check the Slave board number. Call the "DioSetupinitialization" function to add a missing Slave to the DIO domain, or use the Configuration panel in the GTDio Panel in order to add a Slave board.
-4004	The *write/read* command tries to access memory past its configured depth.	Attempts to write/read to DIO Data or Control Memory beyond the available physical memory. This error is trapped before invalid data is stored or retrieved. Check the code. Add memory to DIO boards as needed.
-4005	Incompatible memory size.	The DIO board could not be initialized since it contains different SIMM densities. SIMMs of different capacity (density) cannot be mixed on a board. SIMMs can be different for each board.
-4006	Bank 0 is not installed.	Memory Bank 0 could not be detected on the DIO board. Any DIO board must have bank (0) installed.
-4007	The Program Counter value is out of range.	Make sure the Program Counter specifies an address within the range of physical memory. Add additional memory as required.
-4008	Control Memory is not installed.	Control Memory must be installed for DIO to operate.
-4009	I/O board direction is invalid.	Set the DIO direction to "input" before calling the "GetIOPins" function.
-4010	The Program Counter value does not reside on 32-bit boundaries.	Check the following: If the width is 32 bits, then the Program Counter value modulo 2 must be zero. If the width is 16 bits, then the Program Counter value modulo 4 must be zero. If the width is 8 bits, then the Program Counter value modulo 8 must be zero.
-4011	Mismatching board types are not allowed.	Adding a Slave to a Master in another DIO family is not allowed.

Error Value	Cause/Symptom	Solution
-4012	The memory module(s) was not installed in consecutive slots.	Check that the all memory modules are installed in consecutive slots, for example, bank 0, 1, 2 etc.
-4013	Unrecognized SIMM was detected.	
-4014	Invalid I/O width.	
-4015	Data could not be written correctly to the specified address.	
-4016	Could not find Master board.	
-4017	Unable to find DIO board.	
-4018	In memory not installed.	
-4019	Out memory not installed	
-4020	Illegal timeout or not enough time to finish the operation.	
-4021	Groups directions needs to be all inputs or all outputs.	
-4022	Invalid Signal Active Mode.	
-4100	Unable to set the Control Memory register.	These are hardware problems. The specified register cannot be set properly. Check for a conflict with another installed device in the specified DIO domain. Call Marvin Test Solutions Technical Support for assistance.
-4101	Unable to set the Path register.	
-4102	Unable to set the Control register.	
-4103	Unable to set the Data Memory value.	
-4104	Unable to Reset the DIO Board.	
-4105	Unable to reset Timing board.	
-4106	Invalid command operation code	
-4107	Invalid command register.	
-4108	Invalid command condition.	
-4109	Invalid command address.	
-4110	Invalid command bit number.	
-4111	Invalid Loop command start address, the address must be an even number.	
-4112	Invalid Loop command end address, the address must be an odd number.	
-4113	Invalid Firmware.	These are hardware problems. Check for a conflict with another installed device in the specified DIO domain. Call Marvin Test Solutions Technical Support for assistance.
-4200	GX5150 Data memory failed self-test.	
-4201	GX5150 Control memory of failed self-test.	
-4202	GC50050 Memory In failed self-test.	

Error Value	Cause/Symptom	Solution
-4203	GC5050 Memory Out failed self-test.	
-4204	Direction memory failed self-test.	
-4205	Valid data memory failed self-test.	
TIMING MODULE ERRORS:		
-5000	Arm command fails.	Executing the ARM command failed. One of the following may be the cause: Using an external clock with no signal present. TBD
-5001	This command requires that boards be in the PAUSE state.	Call "DioPause" function before calling the command that failed.
-5002	Halt command failed.	Executing the HALT command failed. One of the following may be the cause: TBD
-5003	The Step command was unable to pause the boards after running the designated steps.	Possible causes are: The External Trigger Event/Mask setting caused the board to trigger. The Sequencer encountered a command that caused it to enter the RUN mode.
-5004	Trigger (TRIG) Command fails.	The PAUSE External Event setting may have set the board in a permanent PAUSE mode.
-5005	PAUSE command fails.	Possible causes are: An External Trigger Event/Mask setting caused the board to trigger. The Trigger source was set to External and the external trigger line is low.
-5006	The board must be in the PAUSE or HALT mode before calling this command.	Call any of the following functions before the command that failed: "DioPause", "DioHalt" or "DioReset".
-5007	The board must be in the HALT mode before calling this command.	Call "DioHalt" or "DioReset" before the command that failed.
-5100	Unable to set the specified frequency.	These are hardware problems. Check for a conflict with another installed device. Call Marvin Test Solutions Technical Support for assistance.
-5101	Unable to set the Trigger External Events register.	
-5102	Unable to set the Trigger External Events Mask register.	
-5103	Unable to set the PAUSE External Events register.	
-5104	Unable to set the PAUSE External Events Mask register.	
-5105	Unable to set the D Events register.	

Error Value	Cause/Symptom	Solution
-5106	Unable to set the D External Events Mask register.	
-5107	Unable to set the External Events register.	
-5108	Unable to set the External Events Source.	
-5109	Unable to set the Program Counter.	
-5110	Unable to set the Counter Overflow mode.	
-5111	Unable to set the I/O pins.	
-5112	Unable to reset the RUN flag.	
-5113	Unable to set the trigger mode.	
-5114	Unable to program the Interface Standard Level.	
I/O MODULE ERRORS:		
-6000	Invalid Group number.	A Group number can only be 0 or 1.
-6001	I/O Module does not exist.	The function could not detect an I/O Module. Check that an I/O module is mounted and properly seated.
-6002	The installed I/O Module does not support termination.	Check the Specification section of your I/O Modules User Guide for modules that support terminations.
-6003	Unrecognized I/O module.	
-6004	In valid Carreir number.	
-6005	A specified Carrier cannot be associated with the specified handle.	
-6006	Maximum number of Carreir boards.	
-6007	Carrier board could not be detected at the given base address.	
-6100	Unable to set the Frequency-Doubler I/O width.	These are hardware problems. The specified register could not be set properly. Check for a conflict with another device installed in the specified DIO. Call Marvin Test Solutions Technical Support for assistance.
-6101	Unable to set the Frequency-Doubler clock source.	
-6102	The Frequency-Doubler cannot be set up correctly.	
-6110	The specified Frequency-Doubler frequency is out of range.	Check the Specification section in your I/O Module User Guide for the correct Frequency-Doubler frequency range.
-6200	The specified level shifter voltage is out of range.	Check the specification section in your I/O Modules and Interface User Guide to verify the allowable voltage range.
-6201	The low-level voltage is out of range.	
-6202	The high level voltage is out of range.	Enter a valid voltage value.

Error Value	Cause/Symptom	Solution
-6203	The threshold level voltage is out of range.	
-6204	The Level Shifter is not calibrated.	
-6205	An invalid calibration value was found in the EEPROM.	
-6206	User does not have permission to set the Level shifter EEPROM data.	
-6207	Writing data to the EEPROM failed.	
-6208	Invalid Level Shifter gain.	
-6209	Invalid Level Shifter offset.	
-6210	Failed to program the specified DAC.	
-6211	The EEPROM CS can not be set.	
-6300	Timeout:Unable to write correctly to the pin electronics memory register.	
-6301	Timeout:Unable to write correctly to the pin electronics settings register.	
-6302	The specified output voltage low is out of range.	
-6303	The specified output voltage low is out of range.	
-6304	The specified output voltage high is out of range.	
-6305	The specified input threshold voltage low is out of range.	
-6306	The specified input threshold voltage low is out of range.	
-6307	The specified input threshold voltage high is out of range.	
-6309	The specified input current load sink is out of range.	
-6310	The specified input current load source is out of range.	
-6311	The specified output current limit is out of range.	
-6312	Invalid low voltage commutating value.	
-6313	Invalid high voltage commutating value.	
-6314	Invalid PMU comparator low voltage.	

Error Value	Cause/Symptom	Solution
-6315	Invalid PMU comparator high voltage.	
-6316	Invalid PMU comparator voltage.	
-6317	Invalid skew delay.	
-6318	The specified forced voltage is out of range.	
-6319	Unable to detect all ICs.	
-6320	Unable to detect the 3.0 reference voltage.	
-6321	Unable to detect the 3.3 voltage supply.	
-6322	Unable to detect positive voltage rail or positive voltage rail is out of range.	
-6323	Unable to detect negative voltage rail or negative voltage rail is out of range.	
-6324	Voltage between positive and negative rails is out of range.	
-6325	Invalid slew rate settings.	
-6326	Invalid channel number.	
-6327	Unable to communicate with all Pin Electronics ICs.	
-6328	Error: channel not in PMU mode.	
-6329	Error: channel not in PMU forced current mode.	
-6330	Error: channel not in PMU forced voltage mode.	
-6331	Error: PMU forced voltage over the specified current limit.	
-6332	Invalid current range value.	
DIO FILE ERRORS:		
-7000	Invalid DIO board type in DIO file.	An invalid board type was found while creating or editing a DIO file.
-7001	Invalid number of steps for this board.	The “DioFileSetNumberOfSteps” function contains an invalid number of steps.
-7002	File type does not match board type.	The DIO file can only be loaded for a specific board type. Mismatched types are not allowed.
-7003	The specified file was opened for reading only.	Reopen the file for read/write.

Error Value	Cause/Symptom	Solution
-7004	The specified DIO files have deferent width.	When comparing two files, both files must have the same width.
-7005	Deferent files types.	When comparing two files, both files must have the same board type.
-7006	Not a ***DIOEasy*** 2.0 file.	File functions needs a ***DIOEasy*** 2.0 file. Convert the file to ***DIOEasy*** 2.0 format before calling those functions.
-7008	The specified file requires GTDIO driver upgrade.	
-7009	The specified file has zero steps.	
-7010	The specified number of steps to be saved is less than the minimum required.	
-7011	Unable to load DIO file. The file has more boards then physical boards.	
-7012	Invalid file name.	
-7013	The specified DIO file is currently been used by another application.	
-7100	The ASCII file has an invalid width specified.	An invalid width was found while loading or editing a DIO ASCII file.
-7101	The ASCII file has an invalid DIO board type specified.	An invalid board type was found while creating or editing a DIO ASCII file. Use any editor to set this value correctly.
-7102	Unable to continue loading ASCII file. Invalid character was found.	Use any editor to set this value correctly.
-7103	Illegal number of steps	
-7104	Illegal number of channels.	
-7105	Ilegal number of boards. (1 to 8 is allowed)	
-7106	Illegal board type.	
-7107	Illegal freqency	
-7108	Illegal ext frequency.	
-7109	Ilegal index of strobe delay.	
-7110	Illegal freqency.	
-7111	Illegal ext frequency.	
-7112	Ilegal index of strobe delay.	
-7114	Illegal index of clock source.	
-7115	Illegal index of strobe source.	
-7116	Illegal DEvent.	
-7117	Illegal DMask.	

Error Value	Cause/Symptom	Solution
-7118	Illegal PEvent.	
-7119	Illegal PMask.	
-7120	Illegal TEvent.	
-7121	Illegal TMask.	
-7122	Illegal index of trigger mode.	
-7123	Illegal ext. event x register.	
-7124	Illegal index of ext. event source.	
-7125	Illegal registerA or registerB.	
-7126	Invalid Clock delay for the specified frequency settings	
-7127	Invalid Strobe delay for the specified frequency settings	
-7200	Unable to open old DIO file.	The driver encountered a problem while reading an old DIO file (created using ***DIOEasy 1.x***). The file may be corrupt. Try to open the file using ***DIOEasy 2.****x* and re-save, or simply create a new DIO file.
-7201	Unable to read old DIO file Control Memory.	
-7202	Unable to read old DIO file data.	
-7203	Unable to read the old DIO file header.	
-7500	Can not open the specified JAM file.	The JAM file is either being used by other application or the file is corrupted.
-7501	Jam file can not be accessed.	
-7502	Can not read JAM file.	
-7503	JAM file failed the CRC check.	The JAM file is corrupted.
-7504	Unexpected end of file in JAM file.	The JAM file is corrupted.
-7505	JAM file has an invalid board type.	
MISCELLANEOUS ERRORS:		
-8xxx	Internal error.	Internal errors range between -8000 and -8999. These errors are internal to the program. They are caused by unhandled exceptions. Record error numbers of this type and report them to Marvin Test Solutions Customer Support for help in resolving them.

Appendix C - ASCII File Formats

Introduction

ASCII file formats are easy to read and edit using any text editor. GTDIO ASCII files are much larger than compressed DIO or DI binary files. Two distinct ASCII formats are available: *Raw* ASCII and ASCII with *Commands.* Both types use *ASC* extensions.

Raw ASCII contains only data and is not board dependent. An *ASCII with Command file (Command ASCII)* contains the same information as a DIO file, such as data, board settings and commands. As a result, the Command ASCII format is board dependent.

DIOEasy and the GTDIO Driver permit 2-way conversion of DIO and DI files to ASCII file types.

Both ASCII file types can contain either DIO (vector) or DI (result) data, depending on the source file. Vector programs can run either Command ASCII or DIO files.

Raw ASCII File Format Overview

Raw ASCII files contain only data. Because they are not board dependent, they can be used to exchange data between any test platforms that use ASCII.

Raw ASCII files save all vector and result data, but not command nor board configuration data. Unsaved data components are lost when the source (DIO or DI) file is converted. Therefore, when a Raw ASCII file is loaded, it cannot run a vector or configure domain boards. Command and configuration data must be added. Raw ASCII files that are read can then be saved as DIO or DI files with added command and configuration data.

Command ASCII File Format Overview

A Command ASCII file contains a header. It also contains channel, direction, label and command data. When a Command ASCII file is loaded, it configures the domain and runs just like a DIO vector file. A Command ASCII file can be freely converted back to native DIO and DI files when saved.

Converting Files to ASCII Format

There are two ways to convert a DIO or DI file to ASCII: use ***DIOEasy*** or use the GTDIO driver.

1. The ***DIOEasy*** Method:

   Open the file in ***DIOEasy***.

   In the file menu select ”Save As...”.

   In the “Save as type” combo box convert as follows:

   For Raw ASCII, select “ASCII Files”.

   For Command ASCII, select “ASCII Files w/Command”.

2. The GTDIO Method:

   Call the “DioFileConvert” command with the conversion mode set to either *Raw ASCII* or *ASCII with Commands*. Details for this method are found in the book "Programmer's Reference User's Guide in the section – Functions Reference".

Comparing Formats

Table A-1 (on the following page) compares RAW and Command type ASCII files side by side. The first column contains header (*Hnn*) data or line numbers. RAW displays in the second column, Command in the last column. Both file formats were derived from the same one-board domain vector source file. The source file was created using ***DIOEasy***. It has a default size of 16,384 steps, but only steps 4 to 21 contain valid channel data. This table is discussed further in following topics.

Note: Header lines in Table A-1 (marked with an H in column 1) appear only in Command ASCII files. Line 0 is the first line in a Raw ASCII file.

Line		**COMMAND ASCII FILE** ↓
H0		[General]
H1		ASCIIVersion=2.0
H2		NumBoards=1
H3		BoardType=GX5050
H4		NumSteps=16384
H5		[Company Information]
H6		Company=
H7		Author=
H8		Notes=
H9		[Setup]
H10	Header for	Frequency=5000000
H11	Command ASCII File	ExtFrequency=5000000
H12	Only	ClockSource=2
H13		StrobeDelay=1
H14		StrobeSource=0
H15		DEvent=0
H16		DMask=0xFFFF
H17		PEvent=0
H18		PMask=0xFFFF
H19		TEvent=0
H20		TMask=0xFFFF
H21		TriggerMode=2
H22		ExtEventSource=0
H23		ExtEventXRegister=0
H24		NumChannelsBoard0=32
H25	**Raw ASCII File** ↓	[Data]
0	0000000000000000XXXXXXX0XXXX000X	0000000000000000XXXXXXX0XXXX000X,OOOO
1	0000000000000000XXXXXXX0XXXX000X	0000000000000000XXXXXXX0XXXX000X

2	00000000000000000XXXXXXX0XXXX000X	00000000000000000XXXXXXX0XXXX000X
3	00000000000000000XXXXXXX0XXXX000X	00000000000000000XXXXXXX0XXXX000X
4	00000000000000000000000000000000	00000000000000000000000000000000,OIOO,Begin
5	00000000000000000000000000000001	00000000000000000000000000000001
6	00000000000000000000000000010000	00000000000000000000000000010000
7	00000000000000000000000000010001	00000000000000000000000000010001
8	00000000000000000000000000100000	00000000000000000000000000100000
9	00000000000000000000000000100001	00000000000000000000000000100001,,ToExitLoop,JGT A 13
10	00000000000000000000000000110000	00000000000000000000000000110000
11	00000000000000000000000000110001	00000000000000000000000000110001
12	00000000000000000000000001000000	00000000000000000000000001000000,OOOO,LoopToBegin,L A 4
13	00000000000000000000000000010001	00000000000000000000000000010001,,NextSeg
14	00000000000000000000000000100000	00000000000000000000000000100000,OIOO
15	00000000000000000000000001000001	00000000000000000000000001000001,OOOO
16	00000000000000000000000000000000	00000000000000000000000000000000
17	00000000000000000000000000010001	00000000000000000000000000010001
18	00000000000000000000000000100000	00000000000000000000000000100000
19	00000000000000000000000001000001	00000000000000000000000001000001
20	00000000000000000000000000000000	00000000000000000000000000000000
21	00000000000000000000000000000001	00000000000000000000000000000001,,End,HLT
22	00000000000000000000000000000000	00000000000000000000000000000000
23	00000000000000000000000000000001	00000000000000000000000000000001

Table C-1: Comparing Raw and Command ASCII Files for One Board

Raw ASCII File Organization

A Raw ASCII example displays in the second column of Table A-1.

Raw ASCII files contain channel data organized as one line per step. The number of file lines *always* equals the number of steps. Each line contains the same number of characters, equal to the number of domain channels configured. Channel Data is represented by one of three ASCII characters: “0”, “1” or “X”. “0” and “1” are logic states. “X” represents either “don’t care” (GC5050/GX5050) or “open” (GX515X) and normally occurs in ASCII files that are converted from DIO (not DI) files.

Channel Data is saved in bit order. The highest order channel is at the left near the beginning of the line. That channel is the highest channel number on the highest board number. The lowest order channel, Board 0 - Channel 0 (Master board), is at the right end of the line. There can be up to 256 characters per line for a fully populated domain. The default file size is 16,384 steps (lines). The file size ranges from 1,024 lines to the largest number of steps supported in the target domain.

Command ASCII File Organization

Command ASCII contains a multi-line header and additional fields that are not present in the Raw ASCII file. Command ASCII is compared with Raw ASCII for a one-board domain in Table A-1. The Command ASCII format has a header and comma delimited fields that extend some Channel Data lines. The Channel Data field is the same in content and organization as in the Raw ASCII file.

Editing a Command ASCII File

The Command ASCII file can be edited with any text editor (for example, Notepad). Interpretation of a Command ASCII vector file is case insensitive, so upper and lower case can be mixed. Data and Command fields are always converted to upper case. All other fields print as edited. Channel Data may only be “1”, “0” or “X”. Direction Field characters may only be “I” or “O”. Letters may be either upper or lower case. Illegal characters generate an error.

Header

The Header (Table A-1, last column) is divided into sections whose tags are [General], [Company Information] and [Setup]. Section tags are contained in brackets and followed by section data. Section data can often be entered in the tabbed ***DIOEasy*** Property function under the File menu, or by calling DIO Driver functions.

The [General] section contains the ASCII file version (2.0), the number of boards programmed (1), type of DIO domain board (GX5050) and the number of steps provided (16384).

The [Company Information] section contains Company, Author and Notes. This information is optional and presented as entered.

The [Setup] section holds parameters that affect domain operation. The last parameter, NumChannelsBoard0, is enumerated on following lines if more boards exist. See “Multi-Board Files” below.

The [Data] section tag follows the last line of the Header field.

Channel Data

The Channel Data field starts after the [Data] tag at the left of the next line, which is line 0. Channel Data is at the beginning of every new line until the end of file.

The Channel Data field is the same for Command ASCII and Raw ASCII. See “Raw ASCII File Organization” above for details.

Other Fields

The Channel Data field can be followed by up to three more comma delimited fields: Direction, Label and Command. These fields are appended to lines when steps are characterized as follows:

It is the first step (step 0). A Direction field is required.

A change in I/O direction occurs.

A label exists.

A command exists.

Data fields are described in Table A-2 below.

Field	Description
Channel Data	Leftmost field. Contains one character per channel: 0, 1 or X. Lowest domain channel appears at line right, the highest at line left.
Direction	Second field. Contains "I" or "O" for each 8-channel group. Ordered left to right, opposite of Channel Data. If a following field exists but there is no direction change, this is a null (empty) field denoted by two commas.
Label	Program label. If there is none and a Command exists, this is a null field.
Command	Program instruction. Last field on the line if it exists, otherwise not written.
Command Parameter List	Part of the command field. Space-delimited list of parameters for the command. The number of parameters depends on the specific command.

Table C-2: Command File Data Fields

Data fields are identified in the order shown in Figure A-1 below.

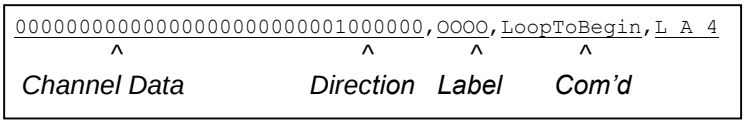

Figure C-1: Example Data Fields in Command ASCII Files

Multi-Board Files

The previous discussion concerned ASCII formats for a single DIO board. For multiple DIO boards, ASCII formats are extended.

The Header adds additional "NumChannelsBoardx = n" lines, where *x* enumerates the board and *n* specifies the number of channels (always 32 for GC5050/GX5050s). There can be up to eight lines for an eight-board domain.

The Channel Data field is augmented to include data for more domain channels. Additional DIO boards can add 8, 16 or 32 new channels. Added board channels are inserted to the left of the next lower board. The augmented Channel Data field appears in both Raw and Command ASCII formats.

The Direction Field (described in the previous section) now includes additional channel groups. As before, one character controls the direction of eight channels.

Table A-3 shows a Command ASCII format for two DIO boards. Bold type designates Master Board Channel Data. Multiple boards do not affect label and Command fields. The format is extensible to larger multi-board domains.

```
[General]

ASCIIVersion=2.0
NumBoards=2
BoardType=GX5050
NumSteps=16384
[Company Infomation]
Company=
Author=
Notes=

[Setup]
Frequency=5000000
ExtFrequency=5000000
ClockSource=2
StrobeDelay=1
StrobeSource=0
DEvent=0
DMask=0xFFFF
PEvent=0
PMask=0xFFFF
TEvent=0
TMask=0xFFFF
TriggerMode=2
ExtEventSource=0
ExtEventXRegister=0xFFFF
NumChannelsBoard0=32
NumChannelsBoard1=32
```

```
[Data]
00000000000000000000000000000000000000000000000xxxxxxx0xxxx000x,OOOOIIIIIIIIIIII
00000000000000000000000000000000000000000000000xxxxxxx0xxxx000x
00000000000000000000000000000000000000000000000xxxxxxx0xxxx000x
00000000000000000000000000000000000000000000000xxxxxxx0xxxx000x
00000000000000000000000000000000000000000000000000000000000000000000000000000000,OIOOIIIIIIIIIIII,Begin
00000000000000000000000000000000000000000000000000000000000000000000000000000001
00000000000000000000000000000000000000000000000000000000000000000000000000010000
00000000000000000000000000000000000000000000000000000000000000000000000000010001
00000000000000000000000000000000000000000000000000000000000000000000000000100000
00000000000000000000000000000000000000000000000000000000000000000000000000100001,,ToExitLoop,JGT A 13
00000000000000000000000000000000000000000000000000000000000000000000000000110000
00000000000000000000000000000000000000000000000000000000000000000000000000110001
00000000000000000000000000000000000000000000000000000000000000000000001000000,OOOOIIIIIIIIIIII,LoopToBegin,L A
4
00000000000000000000000000000000000000000000000000000000000000000000000000010001,,NextSeg
00000000000000000000000000000000000000000000000000000000000000000000000000100000,OIOOIIIIIIIIIIII
00000000000000000000000000000000000000000000000000000000000000000000001000001,OOOOIIIIIIIIIIII
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000010001
00000000000000000000000000000000000000000000000000000000000000000000000000100000
00000000000000000000000000000000000000000000000000000000000000000000001000001
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000001,,End,HLT
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000001
```

Table C-3. Command ASCII File for Two DIO Boards

Index