MI125

Programmer's Reference Guide

October 2015

Revision history

Revision	Date	Comments
0.1	November 2012	First draft.
1.0	December 2012	First release
1.1	February 2013	Ready for revision
1.2	February 2013	Linguistic revision
1.3	June 2013	Added additional test functions
1.4	November 2014	Added MI125_mi125_set_datapaths_calibration() Up to date for Software Tools Release 6.6
1.5	July 2015	Add FMC position and clock Idelay registers
1.6	October 2015	New glossary Added example application description Up to date for Software Tools Release 7

Version 1.6

WARNING

Do not use Nutaq products in conjunction with life-monitoring or life-critical equipment. Failure to observe this warning relieves Nutaq of any and all responsibility.

FCC WARNING

This equipment is intended for use in a controlled environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of personal computers and peripherals pursuant to subpart J of part 15 of the FCC rules. These rules are designed to provide reasonable protection against radio frequency interference. Operating this equipment in other environments may cause interference with radio communications, in which case the user must, at his/her expense, take whatever measures are required to correct this interference.

This page was left intentionally blank.

Table of Contents

List of Figures and Tables

This page was left intentionally blank.

1 Introduction

Congratulations on the purchase of the MI125 FMC.

This document is contains all the information necessary to understand and use your product. It should be read carefully before using the card and stored in a handy location for future reference.

1.1 Conventions

In a procedure containing several steps, the operations that the user has to execute are numbered (1, 2, 3…). The diamond (♦) is used to indicate a procedure containing only one step, or secondary steps. Lowercase letters (a, b, c…) can also be used to indicate secondary steps in a complex procedure.

The abbreviation NC is used to indicate no connection.

Capitals are used to identify any term marked as is on an instrument, such as the names of connectors, buttons, indicator lights, etc. Capitals are also used to identify key names of the computer keyboard.

All terms used in software, such as the names of menus, commands, dialog boxes, text boxes, and options, are presented in bold font style.

The abbreviation N/A is used to indicate something that is not applicable or not available at the time of press.

Note:
The screen captures in this document are taken from the software version available at the time of press. For this reason, they may differ slightly from what appears on your screen, depending on the software version that you are using. Furthermore, the screen captures may differ from what appears on your screen if you use different appearance settings.

1.2 Glossary

This section presents a list of terms used throughout this document and their definition.

Term	Definition
Advanced Mezzanine Card (AMC)	AdvancedMC is targeted to requirements for the next generation of "carrier grade" communications equipment. This series of specifications are designed to work on any carrier card (primarily AdvancedTCA) but also to plug into a backplane directly as defined by MicroTCA specification.
Advanced Telecommunications Computing Architecture (or AdvancedTCA, ATCA)	AdvancedTCA is targeted primarily to requirements for "carrier grade" communications equipment, but has recently expanded its reach into more ruggedized applications geared toward the military/aerospace industries as well. This series of specifications incorporates the latest trends in high speed interconnect technologies, next-generation processors, and improved Reliability, Availability and Serviceability (RAS).
Application Programming Interface (API)	An application programming interface is the interface that a computer system, library, or application provides to allow requests for services to be made of it by other computer programs or to allow data to be exchanged between them.
Board Software Development Kit (BSDK)	The board software development kit gives users the possibility to quickly become fully functional developing C/C++ for the host computer and HDL code for the FPGA through an understanding of all Nutaq boards major interfaces.
Boards and Systems (BAS)	Refers to the division part of Nutaq which is responsible for the development and maintenance of the hardware and software products related to the different Perseus carriers and their different FMC daughter cards.
Carrier	Electronic board on which other boards are connected. In the FMC context, the FMC carrier is the board on which FMC connectors allow a connection between an FMC card and an FPGA. Nutaq has two FMC carriers, the Perseus601x (1 FMC site) and the Perseus611x (2 FMC sites).
Central Communication Engine (CCE)	The Central Communication engine (CCE) is an application that executes on a virtual processor called a MicroBlaze in the FPGA of the Perseus products. It handles all the behavior of the Perseus such as module initialization, clock management, as well as other tasks.
Chassis	Refers to the rigid framework onto which the CPU board, Nutaq development platforms, and other equipment are mounted. It also supports the shell-like case—the housing that protects all the vital internal equipment from dust, moisture, and tampering.
Command Line Interface (CLI)	The Command Line Interface (or CLI) is a basic client interface for Nutaq’s FMC carriers. It runs on a host device. It consists of a shell where commands can be typed, interacting with the different computing elements connected to the system.
FPGA Mezzanine Card (FMC)	FPGA Mezzanine Card is an ANSI/VITA standard that defines I/O mezzanine modules with connection to an FPGA or other device with re-configurable I/O capability. It specifies a low profile connector and compact board size for compatibility with several industry standard slot card, blade, low profile motherboard, and mezzanine form factors.
HDL	Stands for hardware description language.
Host	A host is defined as the device that configures and controls a Nutaq board. The host may be a standard computer or an embedded CPU board in the same chassis system where the Nutaq board is installed. You can develop applications on the host for Nutaq boards through the use of an application programming interface (API) that comprises protocols and functions necessary to build software applications. These API are supplied with the Nutaq board.
MicroTCA (or µTCA)	The MicroTCA (µTCA) specification is a PICMG Standard which has been devised to provide the requirements for a platform for telecommunications equipment. It has been created for AMC cards.
Model-Based Design	Refers to all the Nutaq board-specific tools and software used for development with the boards in MATLAB and Simulink and the Nutaq model-based design kits.
Model-Based Development Kit (MBDK)	The model-based development kit gives users the possibility to create FPGA configuration files, or bitstreams, without the need to be fluent in VHDL. By combining Simulink from Matlab, System Generator from Xilinx and Nutaq’s tools, someone can quickly create fully-functional FPGA bitstreams for the Perseus platforms.

Term	Definition
NTP	Network Time Protocol. NTP is a protocol to synchronize the computer time over a network.
Peer	A host peer is an associated host running RTDEx on either Linux or Windows. An FPGA peer is an associated FPGA device.
PicoDigitizer / PicoSDR Systems	Refers to Nutaq products composed of Perseus AMCs and digitizer or SDR FMCs in a table top format.
PPS	Pulse per second. Event to indicate the start of a new second.
Reception (Rx)	Any data received by the referent is a reception.
Reference Design	Blueprint of an FPGA system implemented on Nutaq boards. It is intended for others to copy and contains the essential elements of a working system (in other words, it is capable of data processing), but third parties may enhance or modify the design as necessary.
Transmission (Tx)	Any data transmitted by the referent is a transmission. Abbreviated TX.
µDigitizer / µSDR Systems	Any Nutaq system composed of a combination of µTCA or ATCA chassis, Perseus AMCs and digitizer or SDR FMCs.
VHDL	Stands for VHSIC hardware description language.

Table 1 Glossary

1.3 Technical Support

Nutaq is firmly committed to providing the highest level of customer service and product support. If you experience any difficulties using our products or if it fails to operate as described, first refer to the documentation accompanying the product. If you find yourself still in need of assistance, visit the technical support page in the Support section of our Web site at www.nutaq.com.

2 Product Description

The MI125 FPGA mezzanine card (FMC) is a 16/32-channel, phased-aligned, A/D card designed around the high-performance LTM9012 QUAD ADC from Linear Technology. The MI125 takes full advantage of the LTM9012 integrated low-noise amplifiers which are suitable for single-ended drive and pulse train signals such as required for imaging applications. Combined with multiple clocks and trigger modes, the MI125 is at its best in DSP applications such as medical/industrial imaging (PET/ultrasound systems), multichannel DAQ, nondestructive testing, radar beamformers, phased array antennas, and multichannel pulse detectors (linear accelerators, synchrotron).

The MI125 complies with VITA 57.1, a widely used standard in the digital signal processing industry, making it easier for developers to integrate FPGAs into embedded system designs.

Outstanding features

The MI125 offers the following outstanding features:

Linear Technology LTM9012 quad, 14-bit, 125-MSPS ADC
The LTM9012 is a 4-channel, simultaneous sampling 14-bit module analog-to-digital converter (ADC) with integrated, fixed gain, differential ADC drivers. The low noise amplifiers are suitable for single-ended drive and pulse train signals such as imaging applications. Each channel includes a low-pass filter between the driver output and ADC input.

http://www.linear.com/product/LTM9012

Micrel SY89540U precision clock switch
The MI125 features an outstanding clock distribution circuit. The main component of this circuit is a Micrel SY89540U integrated circuit. The SY89540U is a low-jitter, low-skew, high-speed, 4x4 crosspoint switch. The SY89540U features a patent-pending isolation design that significantly improves on channel-to-channel crosstalk performance.

Compliant with the VITA 57.1 FPGA mezzanine card

The MI125 uses the VITA 57.1 standard, widely used in the digital signal processing industry, which makes it easier for developers to integrate FPGAs into their embedded system designs. The FMC is completely Plug and Play with Nutaq’s µTCA Perseus AMCs, but the MI125 can as easily be used on any FMC carrier.

Note:
When the MI125 is used in a stacked MI125-32 configuration, the FMC mechanical compliance is not respected. The mechanical envelope of the carrier needs to be confirmed. The electrical compliance is not respected in this configuration (power supplies).

3 FPGA Core Description

3.1 Core Level Features

Digital calibration
The MI125 core can calibrate each LVDS data line for ADC data communication between the Perseus and an MI125 FMC card. Once the data lines are calibrated, the ADC data valid will be asserted to indicate that the ADC data ports are valid.

Trigger direction selection
In the MI125 core, both input and output triggers are present. Physically, there is only one trigger port on the MI125 front panel. To allow the output trigger signal to drive the front panel trigger port, the "MI125_mi125_set_trigout_send" function must be called to configure the MI125 FMC card correctly.

Variable input trigger delay
The input trigger can be delayed to match the ADC chip latency. This delay is variable and can be modified by a core register write operation. The delayed trigger signal is synchronous to the ADC clock.

Double-stack MI125 synchronization
When two MI125 FMC cards are used in a double-stack configuration, the core supports data synchronization of all 32 channels on a common clock. The *Primary* module clock is used and is routed to the *Secondary* module through "MI125 module interconnection signals".

3.2 Core Parameters and Ports

This section presents a list of configuration parameters.

Parameter	Type	Allowable values	Default values	Description
C_BUILD_REVISION	std_logic_vector	*DO NOT CHANGE*	*DO NOT CHANGE*	Core build revision number
C_PRIMARY_MODULE	boolean	{false, true}	true	If *true*, the clock from the FMC connector is used to capture ADC data. If *false*, the clock from "i_IPSerialClockIn_p" is used to capture ADC data.
C_FMC_POSITION	boolean	{false, true}	True	0 = MI125 board directly connected to the first FMC connector. 1 = MI125 board connected on top of an another FMC board (double-stack configuration) on the first FMC connector 2 = MI125 board directly connected to the second FMC connector. 3 = MI125 board connected on top of an another FMC board (double-stack configuration) on the second FMC connector Also, this parameter is used the TCL script to select the right constraints to apply. The trigger signal is only available for boards directly connected to the FMC connector.

Table 2 MI125 core configurable parameters

Port	Range	Direction	Description
o_AdcDataClk_p		Out	ADC design clock
ov14_AdcDataCh#_p	[13:0]	Out	ADC data for channel #, where # is [1, 16]
o_AdcCh#Valid_p		Out	ADC data valid signal for channel #, where # is [1, 16]
o_AdcCh#Enabled_p		Out	ADC chip-enabled status, where the chip number describes the data channels of the ADC chip. The chip numbers are: 1 to 4, 5 to 8, 9 to 12, and 13 to 16.
o_DataFormat_p		Out	ADC data format. *0*: *Unsigned*; *1*: *Signed*
o_TriggerToFPGA_p		Out	ADC trigger from the MI125 front panel to the FPGA
i_TriggerFromFPGA_p		In	ADC trigger from the FPGA to the MI125 front panel

Table 3 MI125 core user ports

Port	Range	Direction	Description
ADC interface signals			
idp_DataFromADC_p	[31:0]	In	ADC differential data bus
idn_DataFromADC_p	[31:0]	In	
idp_ClockFromADC_p	–	In	ADC differential clock signal
idn_ClockFromADC_p	–	In	
idp_FrameFromADC_p	–	In	ADC differential frame signal
idn_FrameFromADC_p	–	In	
MI125 module interconnection signals			
o_IPSerialClockOut_p	–	Out	Serial clock signal of the *Primary* module (from the *Primary* module)
o_IPSerialClockOutDiv_p	–	Out	Divided serial clock signal of the *Primary* module (from the *Primary* module)
i_IPSerialClockIn_p	–	In	Serial clock signal of the *Primary* module (for the *Secondary* module)
i_IPSerialClockInDiv_p	–	In	Divided serial clock signal of the *Primary* module (for the *Secondary* module)
o_primTapAlignedDone_p	–	Out	Tap alignment status of the first bit of the *Primary* module (from the *Primary* module)
ov5_primTapPos_p	[4:0]	Out	Tap alignment position of the first bit of *Primary* module (from *Primary* module)
i_primTapAlignedDone_p	–	In	Tap alignment status of the first bit of the *Primary* module (for the *Secondary* module)
iv5_primTapPos_p	[4:0]	In	Tap alignment position of the first bit of the *Primary* module (for the *Secondary* module)
Miscellaneous signals			
i_logicRst_p	–	In	Resets the MI125 parameters.
i_RefClk200MHz_p	–	In	200 MHz reference clock for IODELAY control

Table 4 MI125 core FMC ports

3.3 Core Registers

3.3.1 Registers Map

Register name	Address from offset	Direction	Description
MI125INFO	0x00	R	Core ID and version identifier
MI125CTRL	0x04	R/W	Control of the inner logic
MI125STATUS	0x08	R	General status register
MI125ADCIDELAYVALUE	0x0C	R	ADC Idelay Value
MI125ADCIDELAYWE	0x10	R/W	ADC Idelay mask
MI125BITSLIP	0x14	R/W	Bitslip mask
MI125CALIBERROR	0x18	R	Indicate if the data is stable for each lane
MI125CALIBPATTERNERROR	0x1C	R	Indicate if the data is the same than the calibration pattern for each lane
MI125FREQCNTCLK	0x20	R/W	Control and status of the clock frequency counter

Table 5 Registers map

3.3.2 Registers Description

Offset 0x00 — MI125INFO

This register shows the MI125 IP core identification and version number.

31 to 16	15 to 0
Core ID	Core version
R	R
0xC125	0x0201

Table 6 MI125INFO register

Bit	Description	Configuration
Core ID	This is the MI125 IP core unique ID number.	0xC125
Core Version	This is the MI125 IP core version number.	0x0201 = Version 2.1

Table 7 MI125INFO register field description

Offset 0x04 — MI125CTRL

This register controls the inner logic of the MI125 core.

31 to 16	15	14 to 10	9	8	7 to 6
Reserved	reset_calib_detection	TriggerDelay	ADCClockMMCMRst	DataFormat	ChannelConfig
R	R/W	R/W	R/W	R/W	R/W
0	0	7	0	1	“11”
5	**4**	**3**	**2**	**1**	**0**
IPSoftRst	DigOutRandEn	IserdesRst	IdelayCtrlRst	IdelayRst	UpdateADCStatus
R/W	R/W	R/W	R/W	R/W	P
1	0	0	1	1	0

Table 8 MI125CTRL register

Bit	Description	Configuration
UpdateADCStatus	Update ADC valid and channel config values. Self-clearing value.	1 = Update adc status
IdelayRst	Resets the IDELAYs.	1 = Reset IDELAY taps 0 = Clear IDELAY taps reset
IserdesRst	Reset Iserdes	1 = Reset Iserdes 0 = Clear Iserdes reset
IdelayCtrlRst	Resets the IDELAY controller.	1 = Reset IDELAY controller 0 = Clear IDELAY controller reset
DigOutRandEn	Applies the XOR at the ADC output to undo the ADC data randomizer.	1 = Undo data randomization enable 0 = Undo data randomization disable
IPSoftRst	IP soft reset to reset the MI125 core registers	1 = Reset MI125 core registers 0 = Clear MI125 core registers reset
ChannelConfig	Indicates which channels are enabled.	"11" = Channels 1 to 16 are active "10" = Channels 1 to 12 are active "01" = Channels 1 to 8 are active "00" = Channels 1 to 4 are active
DataFormat	Specifies the data format of ADCs.	1 = *Signed* 0 = *Unsigned*
ADCClockMMCMRst	ADC MMCM reset	1 = Reset ADC clock MMCM 0 = Clear ADC clock MMCM reset
TriggerDelay	Sets the trigger delay length. This delay is used to synchronize the trigger input of the front panel with the ADC data. This 5-bit vector is interpreted like an *Unsigned* value and the trigger delay is 1 clock cycle higher than the TriggerDelay register value. The trigger output is synchronous with ADC clock.	0 = 1 ADC clock cycle delay 1 = 2 ADC clock cycle delay ... 30 = 31 ADC clock cycle delay 31 = 32 ADC clock cycle delay
reset_calib_detection	Reset calibration detection process	1 = Reset detection process 0 = Clear detection process reset

Table 9 MI125CTRL register field description

Offset 0x08 — MI125STATUS

This register shows the MI125 core status.

31 to 8	7 to 4	3	2	1	0
Reserved	FmcPosition	ADCClockMMCMPresent	ADCClockMMCMLocked	calib_detection_done	idelay_ready
R	R	R	R	R	R
0	-	-	-	-	-

Table 10 MI125STATUS register

Bit	Description	Configuration
idelay_ready	IDELAY controller ready status	1 = IDELAYCTRL is ready 0 = IDELAYCTRL is not ready
calib_detection_done	Calibration Detection process status	1 = Calibration detection process is done 0 = Calibration Detection process is in reset or is still running

Bit	Description	Configuration
ADCClockMMCMLocked	ADC MMCM locked status	1 = ADC MMCM is locked 0 = ADC MMCM is not locked
ADCClockMMCMPresent	ADC MMCM utilization status. Only the *Primary* module MMCM is used. The *Secondary* module MMCM is not used because this module is the *Primary* module clock.	1 = ADC MMCM is used 0 = ADC MMCM is not used
FmcPosition	The FMC position of the MI125 board targeted by the FPGA core instance.	0 = MI125 board directly connected to the first FMC connector. 1 = MI125 board connected on top of an another FMC board (double-stack configuration) on the first FMC connector 2 = MI125 board directly connected to the second FMC connector. 3 = MI125 board connected on top of an another FMC board (double-stack configuration) on the second FMC connector

Table 11 MI125STATUS register field description

Offset 0x0C — MI125ADCIDELAYVALUE

This register controls the ADC Idelay values of the MI125 core.

31 to 10	9 to 5	4:0
Reserved	clkIdelay_value	adcIdelay_value
R	R/W	R/W
0	0	-

Table 12 MI125ADCIDELAYVALUE register

Bit	Description	Configuration
adcIdelay_value	ADC Idelay Value (78 ps/tap)	[0:31] = Idelay taps value
clkIdelay_value	Serial clock Idelay Value (78 ps/tap)	[0:31] = Idelay taps value

Table 13 MI125ADCIDELAYVALUE register field description

Offset 0x10 — MI125ADCIDELAYWE

This register controls the ADC Idelay mask of the MI125 core.

31 to 0
adcIdelay_we
P
"00000000000000000000000000000000"

Table 14 MI125ADCIDELAYWE register

Bit	Description	Configuration
adcIdelay_we	Mask for each LVDS lane for updating the ADC Idelay value with the current adcIdelay_value.	For each bit: 1 = assign current adcIdelay_value to the lane corresponding to the bit position

Table 15 MI125ADCIDELAYWE register field description

Offset 0x14 — MI125BITSLIP

This register controls the ADC Iserdes bitslip of the MI125 core.

31 to 0
bitslip
P
“00000000000000000000000000000000”

Table 16 MI125BITSLIP register

Bit	Description	Configuration
bitslip	Mask for each LVDS lane for enabling the bitslip of the ISERDES.	For each bit: 1 = Enable ISERDES bitslip of the lane corresponding to the bit position

Table 17 MI125BITSLIP register field description

Offset 0x18 — MI125CALIBERROR

This register shows the ADC data stability.

31 to 0
calib_error
R
-

Table 18 MI125CALIBERROR register

Bit	Description	Configuration
calib_error	Indicate if the data is stable for each lane	For each bit: 1 = The received value of the lane corresponding to the bit position is stable 0 = The received value of the lane corresponding to the bit position is unstable

Table 19 MI125CALIBERROR register field description

Offset 0x1C — MI125CALIBPATTERNERROR

This register shows the ADC data pattern verification.

31 to 0
calib_pattern_error
R
-

Table 20 MI125CALIBPATTERNERROR register

Bit	Description	Configuration

Bit	Description	Configuration
calib_pattern_error	Indicate if the data is the same than the calibration pattern for each lane	For each bit: 1 = The received value of the lane corresponding to the bit position is stable and corresponds to the pattern 0 = The received value of the lane corresponding to the bit position is unstable or does not correspond to the pattern

Table 21 MI125CALIBPATTERNERROR register field description

Offset 0x20 — MI125FREQCNTCLK

This register controls the MI125 frequency counters.

31 to 23	22	21 to 16	15 to 0
Reserved	FreqCntRst	FreqCntClkSel	FreqCntClkCnt
R	R/W	R/W	R
0	0	0	-

Table 22 MI125STATUS register

Bit	Description	Configuration
FreqCntClkCnt	Frequency counter indicating the rounded frequency of the selected clock.	Rounded clock frequency in MHz (*Unsigned*)
FreqCntClkSel	Clock selector for the frequency counter	0 = 200-MHz reference clock 2 = ADC data clock
FreqCntRst	Reset of the frequency counter module	1 = Reset frequency counter module 0 = Clear frequency counter module reset

Table 23 MI125STATUS register field description

3.4 FPGA Interfacing

3.4.1 Using the MI125 Core

The MI125 core is an AXI-Lite compatible ip core. That way, the Microblaze has access to its registers for control and status.

An example is provided to show how the MI125 ip core can be used in an EDK environment: “perseus6010_mi125_acquisition”.

3.4.2 Interfacing the MI125 with User Logic

The MI125 user interface provides a 14-bit data bus for each channel, one data valid signal per channel and one common user data clock. It also includes 4 ADC chip enable statuses, one common data format, and an input trigger and an output trigger signals. Refer to the Core Parameters and Ports section for core user ports.

Example: interfacing with the Record/Playback FPGA core

The MI125 core can be connected to the Record/Playback core for data recording on the Perseus SODIMM memory. The host PC can then grab the recorded data and save it in the form of a bin file.

To interface the MI125 core with the Record/Playback core, an interface pcore is provided. As the SODIMM memory throughput cannot handle 32 channels with 14 bits of precision at 125 MS/s the pcore has various operation modes.

The pcore can select channels 1 to 16, channels 17 to 32, or a down-sampled version of channels 1 to 32. The pcore extends the 14-bit input data to 16 bits with the signed extension corresponding to the ADC data format. Also, the MI125-Record interface pcore allows the Record/Playback module to be triggered by the MI125 input trigger or by a Perseus custom register.

An example is provided to show how to connect the MI125 core to the Record/Playback core for data recording on the host PC.

4 Low-Level and Host Programming

This chapter presents the information needed to program the MI125.

4.1 MI125 Driver

Function	Use
MI125_mi125_open	Opens an MI125 module instance.
MI125_mi125_reset	Resets the MI125 module instance to default operating conditions.
MI125_mi125_close	Closes the MI125 module instance.
MI125_mi125_digital_adccalibration	Forces a digital calibration of all ADCs with active channels.
MI125_mi125_set_config	Configures the board with user-specified options.
MI125_mi125_set_clksrc	Configures the clock source that the boards ADC will use for all channels.
MI125_mi125_set_trigout	Configures the trigger out I/O to be disconnected (default) or connected from the FPGA to the outside.
MI125_mi125_set_testmode	Configures the special test mode (mostly use at production time for test purposes).
MI125_mi125_get_temperature	Reads the PCB temperature.
MI125_mi125_get_channelcalibstatus	Reads the ADC channels calibration information.
MI125_mi125_get_versions	Reads the MI125 FPGA core module version and driver version information.
MI125_mi125_checkadc	Checks the ADC communication for diagnostic purposes.
MI125_mi125_checkcore	Attempts to detect the MI125 FPGA core, for diagnostic purposes.
MI125_mi125_get_clkmaster	Detects if this current module instance is a clock master for the FPGA (uses its own MMCM).
MI125_mi125_clockreset	Forces an MCM clock reset.
MI125_mi125_scan_i2c	Scan the i2c bus to detect all module devices (mostly use at production time for test purpose).
MI125_mi125_test_localoscsw	Test the local onboard clock oscillator switch control (mostly use at production time for test purpose).
MI125_mi125_set_datapaths_calibration	Specifies the data paths lanes calibration and calculates the fpga tap correction for each data lane.

Table 24 MI125 driver functions

4.2 MI125 EAPI Library

Function	Use
MI125_Presence_send	Minimally verifies if an FMC board is detected and an MI125 FPGA core is detected.
MI125_powerup_send	Powers up the MI125 before use.
MI125_mi125_reset_send	Resets the MI125 module instance to default operating conditions.
MI125_mi125_digital_adccalibration_send	Forces a digital calibration of all SDCs with active channels.
MI125_mi125_set_config_send	Configures the board with user-specified options.

Function	Use
MI125_mi125_set_clksrc_send	Configures the clock source that the boards ADC will use for all channels.
MI125_mi125_set_trigout_send	Configures the trigger out I/O to be disconnected (default) or connected from the FPGA to the outside.
MI125_mi125_set_testmode_send	Configures the special test mode (mostly use at production time for test purposes).
MI125_mi125_get_temperature_send	Reads the PCB temperature.
MI125_mi125_get_channelcalibstatus_send	Reads the ADC channels calibration information.
MI125_mi125_get_versions_send	Reads the MI125 FPGA core module version and driver version information.
MI125_mi125_checkadc_send	Checks the ADC communication for diagnostic purposes.
MI125_mi125_checkcore_send	Attempts to detect the MI125 FPGA core, for diagnostic purposes.
MI125_mi125_get_clkmaster_send	Detects if this current module instance is a clock master for the FPGA (uses its own MMCM).
MI125_mi125_clockreset_send	Forces an MCM clock reset.
MI125_mi125_scan_i2c_send	Scan the i2c bus to detect all module devices (mostly use at production time for test purpose).
MI125_mi125_test_localoscsw_send	Test the local onboard clock oscillator switch control (mostly use at production time for test purpose).

Table 25 MI125 EAPI functions

4.3 MI125 Command Line Interface (CLI) Commands

Function	Use
mi125_powerup	Powers up the MI125 before use.
mi125_reset	Resets the MI125 module instance to default operating conditions.
mi125_digital_adccalibration	Forces a digital calibration of all ADCs with active channels.
mi125_set_config	Configures the board with user-specified options.
mi125_set_clksrc	Configures the clock source that the boards ADC will used for all channels.
mi125_checkcore	Attempts to detect the MI125 FPGA core, for diagnostic purposes.
mi125_set_testmode	Configures the special test mode (mostly use at production time for test purposes).
mi125_get_temperature	Reads the PCB temperature.
mi125_get_channelcalibstatus	Reads the ADC channels calibration information.
mi125_get_versions	Reads the MI125 FPGA core module version and driver version information.
mi125_checkadc	Checks the ADC communication for diagnostic purposes.
mi125_set_trigout	Configures the trigger out I/O to be disconnected (default) or connected from the FPGA to the outside.
mi125_get_clkmaster	Detects if this current module instance is a clock master for the FPGA (uses its own MMCM).
mi125_clockreset	Forces an MCM clock reset.
mi125_scan_i2c	Scan the i2c bus to detect all module devices (mostly use at production time for test purpose).
mi125_test_localoscsw	Test the local onboard clock oscillator switch control (mostly use at production time for test purpose).

Table 26 MI125 CLI functions

4.4 Building a Host Application Using the MI125

The following diagram illustrates the typical flow of an MI125 application.

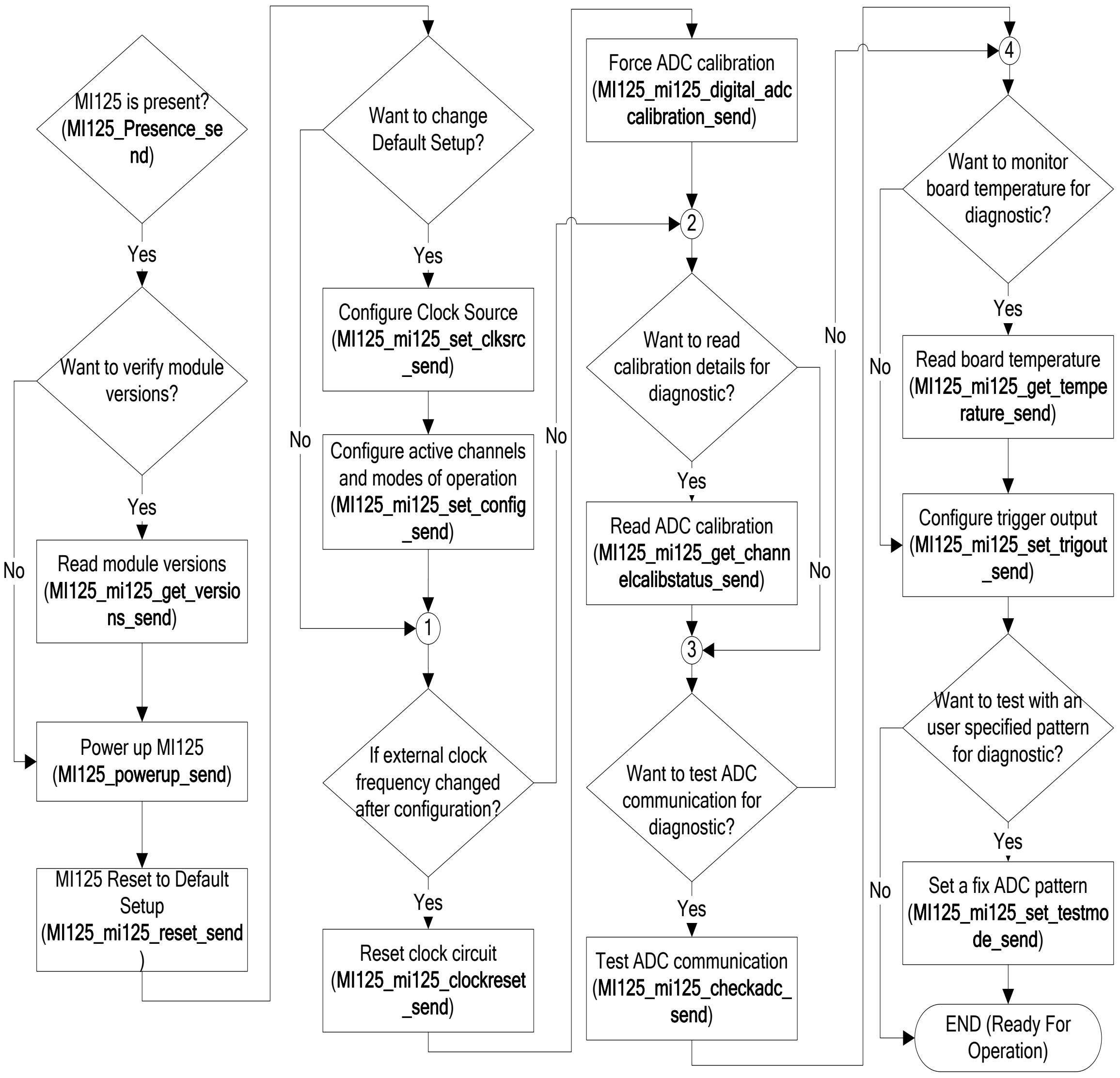
MI125 is present? (MI125_Presence_send)
Yes
Want to verify module versions?
Yes
No
Read module versions (MI125_mi125_get_versions_send)
Power up MI125 (MI125_powerup_send)
MI125 Reset to Default Setup (MI125_mi125_reset_send)
Want to change Default Setup?
Yes
No
Configure Clock Source (MI125_mi125_set_clksrc_send)
Configure active channels and modes of operation (MI125_mi125_set_config_send)
1
If external clock frequency changed after configuration?
Yes
No
Reset clock circuit (MI125_mi125_clockreset_send)
Force ADC calibration (MI125_mi125_digital_adccalibration_send)
2
Want to read calibration details for diagnostic?
Yes
No
Read ADC calibration (MI125_mi125_get_channelcalibstatus_send)
3
Want to test ADC communication for diagnostic?
Yes
No
Test ADC communication (MI125_mi125_checkadc_send)
4
Want to monitor board temperature for diagnostic?
Yes
No
Read board temperature (MI125_mi125_get_temperature_send)
Configure trigger output (MI125_mi125_set_trigout_send)
Want to test with an user specified pattern for diagnostic?
Yes
No
Set a fix ADC pattern (MI125_mi125_set_testmode_send)
END (Ready For Operation)

Figure 4-1 MI125 typical host application flow chart

The first step a programmer should take when building an application for an MI125 is to verify the presence of the FPGA core within the bitstream programmed in the FPGA and the presence of the MI125 card on the FMC site. Once the verification has been made, the FMC site can be powered up by the carrier. The previous flow chart shows a typical application for a single MI125 (MI125-16-S) module. If a double-stack configuration (MI125-32) is used the function "mi125_get_clkmaster" can be used to know which module is the FPGA clock master. If one MI125 module is not a clock master, this means that it is configured to be clocked from the other module, that is, any changes to the clock made to the master module will also apply to the slave module which will require a clock reset and calibration as well. The "END" of the flow above (Ready For Operation) means that the board is initialized correctly. To trigger a capture, an external trigger condition (outside MI125 core) must be generated. It should be noted that only active configured channel will generate valid data, while the others are in power down.

Warning:
There is a single module external “ET” pin (trigger) shared between the core trigger in/out signals: the function “MI125_mi125_set_trigout_send” must be used to disable the trigger out signal if the external “ET” pin is used as a trigger in function (default).

5 MBDK Programming and Interfacing

The description of the MI125 FPGA MBDK block is presented in the *%BASROOT%\doc\html\mbdk_fpga_mi125.htm* documentation.

The description of the MI125 Simulink host MBDK block is presented in the *%BASROOT%\doc\html\mbdk_simulink_mi125.htm* documentation.

6 Host Applications Using the Record/Playback EAPI Library

The BAS software suite provides an example utility application that the user can modify. This utility application completely configures the MI125, making it ready for data processing in your FPGA design or data recording. It uses a configuration file which describes the parameters to apply. This is explained in section 6.1.

6.1 Using and Modifying the MI125 Initialization Example Application

An example utility application, called *MI125_Init*, is provided with your BAS software suite, with a pre-compiled executable made available in your %BASROOT%\tools\bin directory.

An applicative example showcases how this application is used and is available in your %BASROOT%\examples\mi125\host\scripts directory. It demonstrates how to

- configure the MI125 FMC and
- perform a data record using the Record/Playback core.

6.1.1 MI125_Init description

Usage: *MI125_Init <ip address> <configuration file name>*

- Parameter 1 must be the carrier IP address, ex 192.168.0.101
- Parameter 2 must be the Configuration file name, ex MI125_Init.ini

This application configures and enables MI125 FMCs according to parameters stored in the configuration file. The structure and content of this configuration file is described in section 6.1.2.

1. It parses MI125 parameters from the configuration file whose name is passed as command line argument.
2. It validates whether parameters for at least one MI125 were found in the configuration file.
3. the application connects to the carrier CCE using `connect_cce()`, which gets the application a `connection_state` object. This object will be used for all other EAPI function calls.
4. It initializes/configures MI125 FMCs for which parameters were found in the configuration file using function `MI125_Init()`.
    - For a maximum of four MI125 boards (preprocessor definition `MAX_MI125`), `MI125_Init()` loops and:
        - Validates if a MI125 is to be configured on the present index by verifying whether the handle points to a valid configuration parameters structure.
        - Calls `MI125_Presence_send()` to verify if a MI125 is indeed present in the system.
        - Calls `MI125_mi125_get_clkmaster_send()` to inform the user whether the present board to be configured is a clock master. *MI125_Init* is designed so that this function call only provides information to the user using the console. User can then use this

information to validate whether the configuration file used by *MI125_Init* is properly written. See documentation for `MI125_mi125_get_clkmaster_send()` for details.

- Calls `MI125_mi125_get_versions_send()` to inform the user on core and driver versions.
- Calls `MI125_powerup_send` and `MI125_mi125_reset_send()` to power up and reset the MI125 board.
- Uses function `MI125_mi125_set_clksrc_send()` to configure the clock source for the present MI125 board. *MI125_Init* passes the parameter specified in the configuration file to `MI125_mi125_set_clksrc_send()`. Care must be taken when writing the configuration file. Referring to the call to `MI125_mi125_get_clkmaster_send()` and its documentation, a MI125 board that is not clock master may only use the bottom FMC as its clock source. That is, `MI125_CLKSRCBOTTOMFMC` must be passed when calling `MI125_mi125_set_clksrc_send()` on such board.
- Calls `MI125_mi125_set_trigout_send()` to set the trigger out function.
- Calls `MI125_mi125_set_config_send()` to configure options for the present MI125 board. Four options are configured when this function is called. The provided implementation of *MI125_Init* does not permit these options to be specified in the configuration file.
- Calls `MI125_mi125_get_temperature_send()` to display an estimate of the MI125 PCB temperature.

5. It disconnects from the CCE using function `disconnect_cce()`.

6.1.2 MI125_Init configuration file

Instead of getting its parameters through command line arguments, application *MI125_Init* uses a separate configuration file to gather them. The path to this file is passed as an argument to the application. Details on how Nutaq's configuration files are built are available in the application note "*App Note - Parsing and Creating Configuration Files*" residing in the %BASROOT%\doc\app_notes repository.

To get a feel of what the *MI125_Init* configuration file looks like, the reader is encouraged to open file *MI125_Init.ini* in %BASROOT%\examples\mi125\host\scripts with a text editor. It is provided with the MI125 applicative example. The file is similar to this example:

```
#This is the FMC type in this slot position
[FMC1,FMC2]
type=mi125

#This is the flag to indicate if the FMC will be initialized or not
#Bypass Initialization = 0
#Initialize FMC = 1
[FMC1]
active_flag=1
[FMC2]
active_flag=1

#This is the sampling clock source
```

```
#MI125_CLKSRC125MHZ                    = 0,        ///< Internal 125 MHz (default)
#MI125_CLKSRCEXT                       = 1,        ///< External clock
#MI125_CLKSRCBOTTOMFMC                 = 2,        ///< This setting must not be used for bottom FMC
(main) card, clock top board from bottom board
#MI125_CLKSRCFMCCARRIER                = 4,        ///< FMC carrier clock (future)
[FMC1]
sampling_clock_source=0
[FMC2]
sampling_clock_source=2

#This is the trigger output IO to be disconnected (default) or connected from FPGA to outside.
[FMC1,FMC2]
trigger_out=OFF
```

Configuration files for a MI125 must contain all MI125 parameters.

- `type` (an instance of that parameter must be equal to `mi125`)
- `active_flag`. Tells *MI125_Init* to enable (1), or not (0), the MI125.
- `sampling_clock_source`. Tells *MI125_Init* the source of the sampling clock. Possible values:
    - `0:` The MI125 uses the internal 125 Mhz oscillator
    - `1:` The MI125 uses an external clock
    - `2:` The MI125 uses the bottom FMC as clock source. Setting only applicable to top MI125s
    - `4:` The MI125 uses the clock coming from the carrier (reserved, not yet available)
- `trigger_out`: Tells *MI125_Init* to route “trigger out” signal from the FPGA to the MI125 front panel (on), or not (`off`).

7 Functional Examples

Functional examples for the MI125 have been implemented for Nutaq's Perseus AMC carrier. This chapter presents the examples and offers links to the example documentation.

7.1 BSDK Example

The MI125 BSDK Perseus examples showcase the MI125 design with the Record/Playback module on the Perseus using the Xilinx's Platform Studio, Microsoft's Visual Studio, and Nutaq's Command Line Interface.

The procedures for the examples are presented in the MI125 Perseus example document:

- *%BASROOT%\doc\fmc\MI125\MI125 Examples Guide.pdf*

7.2 MBDK Example

The MI125 MBDK Perseus examples showcase the MI125 design with the Record/Playback module on the Perseus using Matlab Simulink, Xilinx's System Generator, Microsoft's Visual Studio, and Nutaq's Command Line Interface.

The procedures for the examples are presented in the MI125 Perseus example document:

- *%BASROOT%\doc\fmc\MI125\MI125 Examples Guide.pdf*