Nutaq® INNOVATION TODAY FOR TOMORROW®

Record/Playback

Programmer's Reference Guide

February 2016

Revision history

Revision	Date	Comments
0.1	October 2012	First draft.
0.9	November 2012	Ready for revision
1.0	February 2013	Linguistic revision
1.1	March 2013	Added 4GB SODIMM support
1.2	September 2013	Corrected layout
1.3	January 2014	Added timing information Up to date for Software Tools Release 6.6
1.4	July 2015	Added section 6.2: Using and Modifying the Record/Playback Example Applications Up to date for Software Tools Release 7
1.5	October 2015	New glossary
1.6	February 2016	Up to date for PicoSDR and BAS Software Tools 7.0
1.7	May 2016	Removed functional examples section

To ensure the accuracy of the information contained herein, particular attention was given to usage in preparing this document. It corresponds to the product version manufactured prior to the date appearing on the title page. There may be differences between the document and the product, if the product was modified after the production of the document.

Nutaq reserves itself the right to make changes and improvements to the product described in this document at any time and without notice.

Version 1.7

Trademarks

Acrobat, Adobe, and Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. IBM is a registered trademark of International Business Machines Corporation in the United States, other countries, or both. Intel and Pentium are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. Microsoft, MS-DOS, Windows, Windows NT, and the Windows logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. MATLAB, Simulink, and Real-Time Workshop are registered trademarks of The MathWorks, Inc. Xilinx, Spartan, and Virtex are registered trademarks of Xilinx, Inc. Texas Instruments, Code Composer Studio, C62x, C64x, and C67x are trademarks of Texas Instruments Incorporated. All other product names are trademarks or registered trademarks of their respective holders.

The TM and ® marks have been omitted from the text.

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 Nutaq's Perseus AMC.

This document contains all the information necessary to understand and use the record/playback module of 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

Nutaq's Record/Playback FPGA Core allows you to store data to and get data from the Perseus DDR3 memory.

2.1 Outstanding Features

Nutaq's FPGA devices can be equipped with a four-port memory controller core that allows it to sequentially access the system memory. The core makes it possible to store data from the ADC channels; for example, in the memory. Similarly, it makes it possible to read data from the memory, and then sent it to the DAC channels. The core also has two ports that can be connected to the RTDEx core, allowing a host device to read data from or write data to the memory of the device.

Record
The core can be used to record data generated by the user logic to the memory.

Playback
The core can be used to playback data from the memory to the user logic.

Memory to host transfers
The core can be used to retrieve data from the memory through the RTDEx module.

Host to memory transfers
The core can be used to write data to the memory through the RTDEx module.

It is possible to configure the core several different ways at compilation, but not at runtime. Configurable parameters are the number of recording and playback ports, as well as the widths of these ports, and this is achieved through asymmetrical FIFOs.

1 GBytes and 4 GBytes support

The Record/Playback support both the 1GB and the 4GB SODIMM available for the Perseus. It is not necessary to recompile the bitstream to switch between the two memory sizes. The memory size is detected automatically by the Central Command Engine upon the Perseus' boot.

3 FPGA Core Description

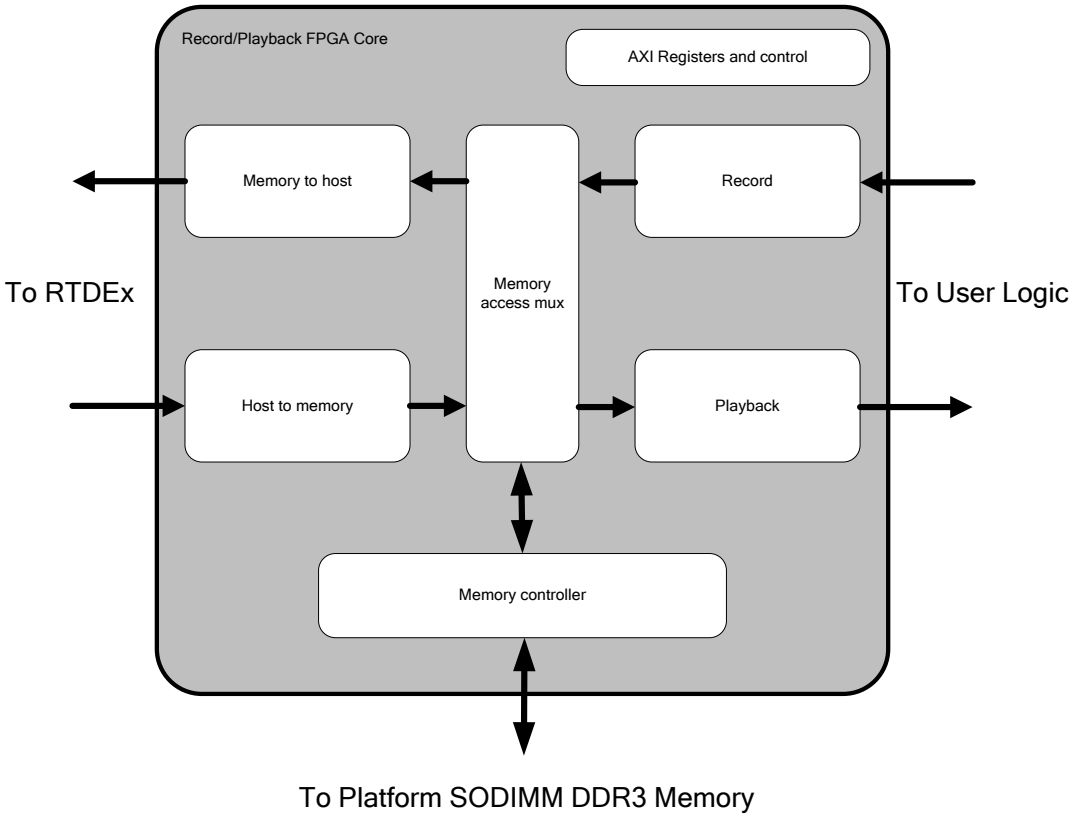

Figure 3-1 Record/Playback core diagram

3.1 Core Level Features

The Record/Playback FPGA core has a total of four data transfer modes to access the system memory. Only one transfer mode can be used at a time. In addition to the transfer modes, the Record/Playback core contains the following features:

- Core Reset:
  - A core reset flushes all FIFOs and resets the core logic and registers to the default values.
- Memory Controller Reset:
  - A memory controller reset resets the memory controller logics and forces an alignment recalibration of the interface.

3.2 Record Features

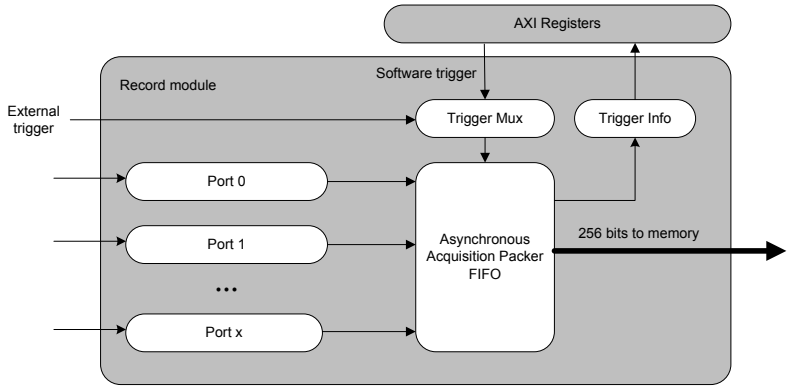

Figure 3-2 Record diagram

Record enable

Once the Record/Playback core mode of operation is set to Record, the module starts sending data received on the Record ports to the memory controller for storage. The memory is used as a circular buffer until a trigger (hardware or software) is applied.

Record ports

The Record module writes to the memory on a width of 256 bits. The number of record ports and their width both can be configured by the user. The following table shows the valid combinations for these two variables.

Note:
All the ports must have the same width because the acquisition packer FIFO packs its input data on 256 bits; the product of the number of ports and of the width of these ports cannot exceed 256.

Size of Port	Number of Ports	Design Clock Maximum
8 bits	1	Design specific
	2	Design specific
	4	Design specific
	8	Design specific
	16	Design specific
16 bits	1	Design specific
	2	Design specific
	4	Design specific
	8	Design specific
	16	180 MHz
32 bits	1	Design specific
	2	Design specific
	4	Design specific
	8	180 MHz
64 bits	1	Design specific
	2	Design specific
	4	180 MHz

Table 2 Valid combinations for the number of ports and port width parameters

Transfer size

A transfer size in bytes must be specified. This refers to the amount of bytes read on a single playback.

Hardware trigger

The Record module has an external trigger signal. This trigger is rising-edge sensitive. The rising-edge event is only monitored once the recording process is running. If no data is written (i_RecWriteEn_p = '0') during the trigger rising-edge clock cycle, the trigger event will be applied the next time a data write operation is performed.

Software trigger

The user can trigger a record by writing to the RECORD_SOFT_TRIG register.

Transfer completion

A bit in the TRANSFER_OVER register allows the user to know the record completed successfully.

Recorded data address

The user can retrieve the recorded data from the memory using the trigger address, the parity address, and the trigger offset. Data is recorded to the memory in blocks of two 256-bit words and the address is incremented by 8. The address points to 64-bits words since this is the size of the DDR3 bus.

- Once a trigger is received by the Record module, the location in the memory of the current block of 512 bits being written is recorded. This is the "trigger address".
- The "parity address" (either 0 or 1) identifies if the trigger was received during the write of the first (0) or the second (1) 256-bit word of the 512-bit block.
- The "trigger offset" refers to the delay in bytes in the 256-bit bloc identified by the "parity address" at which the trigger was recorded.

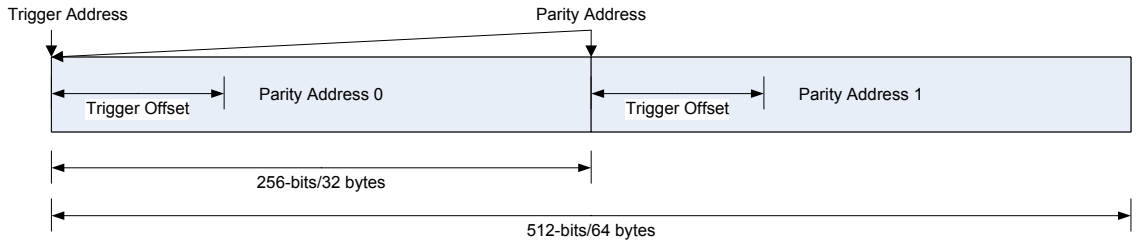

Figure 3-3 Trigger position description

3.3 Playback Features

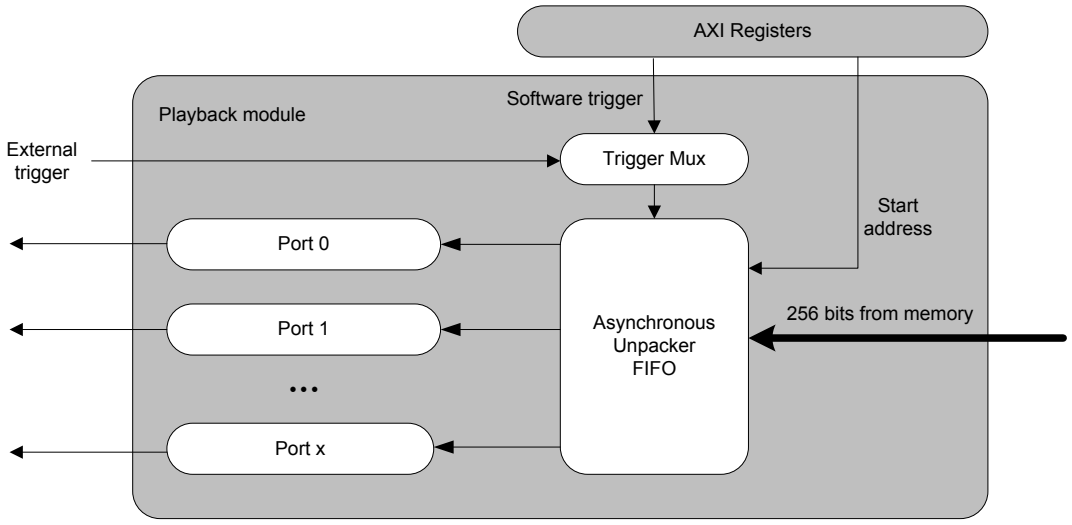

Figure 3-4 Playback diagram

Playback ports

The Playback module reads in the memory on a width of 256 bits. The number of Playback ports and their widths both can be configured by the user. The following table shows the valid combinations for these two variables.

Note:

All the ports must have the same width because the unpacker FIFO initiates reads of 256 bits in the memory to feed each port contiguously, the product of the number of ports and of the width of these ports cannot exceed 256.

Size of Port	Number of Ports	Design Clock Maximum
8 bits	1	Design specific
	2	Design specific
	4	Design specific
	8	Design specific
	16	Design specific
16 bits	1	Design specific
	2	Design specific
	4	Design specific
	8	Design specific
	16	180 MHz
32 bits	1	Design specific
	2	Design specific
	4	Design specific
	8	180 MHz
64 bits	1	Design specific
	2	Design specific
	4	180 MHz

Table 3 Valid combinations for the number of ports and port width parameters

Start address

A start address must be specified for the playback operation. The start address refers to the location in the memory from which the playback will start. The address must be specified on a 64-bit boundary.

Transfer size

A transfer size in bytes must be specified. This refers to the amount of bytes read on a single playback.

Hardware trigger

The Playback module has an external trigger signal which can be used to start a Playback transfer.

Software trigger

The user can trigger a Playback transfer by writing to the PLAYBACK_SOFT_TRIG register.

Playback transfer mode

The user can determine whether to use the Single Playback mode or the Continuous Playback mode.

- Single transfer: The playback stops once the transfer size has been reached.
- Continuous transfer: The playback restarts from the start address once the transfer size has been reached. The transfer is repeated infinitely.

Transfer completion

In the case of the Single Playback mode, a bit in the TRANSFER_OVER register allows the user to know the playback completed successfully.

3.4 Memory-to-Host Features

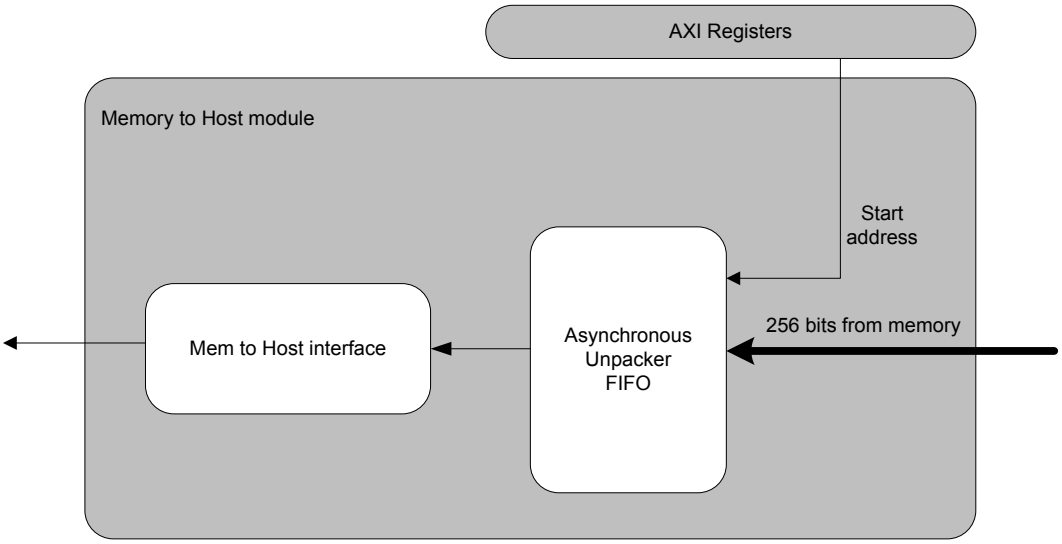

Figure 3-5 Memory-to-host module diagram

Memory-to-host interface

The asynchronous unpacker FIFO triggers reads of 256 bits of data from the memory. The FIFO unpacks the data to 32-bit samples and transfers them to the host to memory interface for RTDEx transmission.

Start address

A start address must be specified for the memory-to-host operation. The start address refers to the location in the memory from which the memory-to-host transfer will start. The address must be specified on a 64-bit boundary.

Transfer size

A transfer size in bytes must be specified. This refers to the amount of bytes to read on a single memory-to-host transfer. The transfer size must be a multiple of 64 bytes.

3.5 Host-to-Memory Features

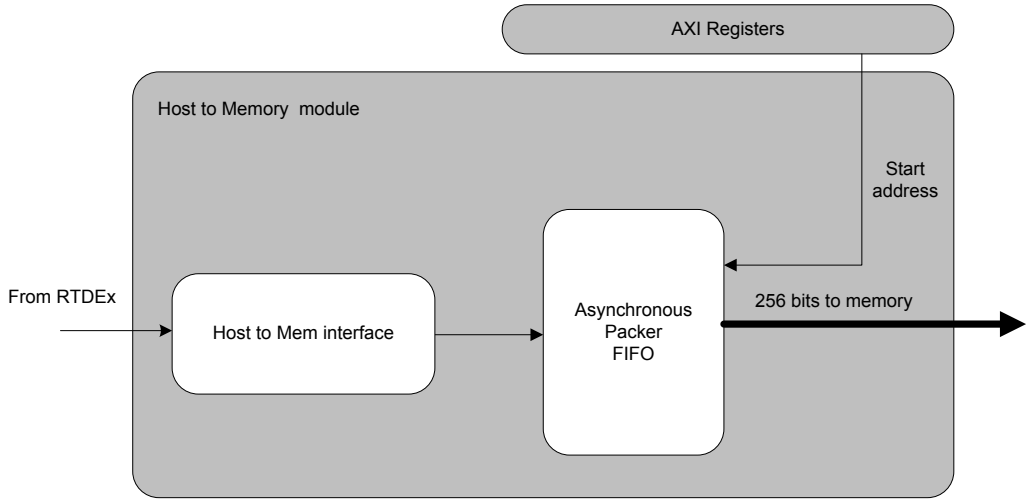

Figure 3-6 Host-to-memory module diagram

Host-to-memory interface

The interface receives 32-bit samples from the RTDEx core. The samples are then sent to the asynchronous packer FIFO which packs them into chunks of data 256 bits wide and triggers a memory write.

Start address

A start address must be specified for the host-to-memory operation. The start address refers to the location in the memory to which the first received byte will be written. The address must be specified on a 64-bit boundary.

Transfer size

A transfer size in bytes must be specified. This refers to the amount of bytes read on a single memory-to-host transfer. The transfer size must be a multiple of 64 bytes.

3.6 Core Parameters and Ports

The table below describes the generic parameters of the core.

Parameter	Range	Default	Description
RecordPortWidth_g	Integer	8	Specifies the width of each recording port. See the recording and playback configurations table above for the valid parameter values.
NumberOfRecordPorts_g	Integer	1	Specifies the number of recording ports. See the recording and playback configurations table above for the valid parameter values.
PlayBackPortWidth_g	Integer	8	Specifies the width of each playback port. See the recording and playback configurations table above for the valid parameter values.
NumberOfPlayBackPorts_g	Integer	1	Specifies the number of playback ports. See the recording and playback configurations table above for the valid parameter values.

Table 4 Core parameter description

The core ports interface is divided into three sections: user interface, RTDEx interface, and memory controller interface. These ports interfaces are shown in tables below.

Port	Range	Direction	Description
i_RecDataClk_p	–	In	User clock for the recording interface.
i_RecTrigger_p	–	In	Triggers the acquisition and tags the position of the data currently being sampled. Also indicates to the recording interface to start counting the number of bytes written in order to stop the acquisition after the amount specified by the user. The trigger is rising-edge sensitive. If no data is written (i_RecWriteEn_p = '0') during the trigger rising-edge clock cycle, the trigger event will be applied the next time a data write operation is performed.
iv_RecDataPort#_p	[RecordPortWidth_g-1:0]	In	Holds data from the recording interface to be written to the memory.
i_RecWriteEn_p	–	In	Indicates that data on ports *iv_RecDataPort#_p* is valid and can be written to the memory.
o_RecFifoFull_p	–	Out	Indicates whether the recording interface FIFO is full.
o_RecFifoEmpty_p	–	Out	Indicates whether the recording interface FIFO is empty.
i_PlayDataClk_p	–	In	User clock for the playback interface.
i_PlayTriggerIn_p	–	In	When set to *1*, indicates to the playback port that data can be read from the memory and be output by the ov_PlayDataPort#_p ports.
ov_PlayDataPort#_p	[PlayBackPortWidth_g-1:0]	Out	Holds data read from the memory and sent to the playback interface.
o_PlayValid_p	–	Out	Indicates that the data on the ov_PlayDataPort#_p ports is valid.
i_PlayReadEn_p	–	In	Allows sending read commands to the playback interface FIFO to output data on the ov_PlayDataPort#_p ports.
o_PlayEmpty_p	–	Out	Indicates whether the playback interface FIFO is empty.

Table 5 User interface ports description

Port	Range	Direction	Description
i_RTDExRxClock_p	–	In	User clock for the RTDEx RX interface.
o_RxRe_p	–	Out	Used to send read signals to the RTDEx RX FIFO to intercept data from the host device.
i_RxReady_p	–	In	Indicates whether the RTDEx RX FIFO is ready.
iv32_RxDataCh0_p	[31:0]	In	RX FIFO data ports from RTDEx. Must be connected to RTDEx channels.
i_RTDExTxClock_p	–	In	User clock for the RTDEx TX interface.
o_TxWe_p	–	Out	Used to send write enable signals to the RTDEx TX FIFO, allowing data to be transferred to the host device.
i_TxReady_p	–	In	Indicates whether the RTDEx TX FIFO is ready to write.
ov32_TxDataCh0_p	[31:0]	Out	TX FIFO data ports from RTDEx. Must be connected to RTDEx channels.

Table 6 RTDEx interface ports description

Port	Range	Direction	Description
sys_clk	–	In	Memory system clock. Must be connected to the Perseus differential 400 MHz clock pair (o_pDdr3MpmcClk_p and o_nDdr3MpmcClk_p) through an IBUFDS buffer.
clk_ref	–	In	200-MHz reference clock normally used in calibrating the IDELAYCTRLs. Not used at present because IDELAYCTRLs are instantiated in the MPMC.
ddr3_dq	[63:0]	Inout	Must be connected to the DDR3 memory data bus.
ddr3_addr	[15:0]	Out	Must be connected to the DDR3 memory address bus. When a 1 GB DDR3 memory is used, only 14 bits are used. When the DDR3 controller try to write at a higher memory space, the address wraps and the data is written in the 1 GB memory.
ddr3_ba	[2:0]	Out	Must be connected to the DDR3 memory bank bus.
ddr3_ras_n	–	Out	Must be connected to the DDR3 memory row address strobe port.
ddr3_cas_n	–	Out	Must be connected to the DDR3 memory column address strobe port.
ddr3_we_n	–	Out	Must be connected to the DDR3 memory write enable port.
ddr3_reset_n	–	Out	Must be connected to the DDR3 memory reset port.
ddr3_cs_n	[0:0]	Out	Must be connected to the DDR3 memory chip selection port.
ddr3_odt	[0:0]	Out	Must be connected to the DDR3 memory on-die termination port.
ddr3_cke	[0:0]	Out	Must be connected to the DDR3 memory clock enable port.
ddr3_dm	[7:0]	Out	Must be connected to the DDR3 memory data mask port.
ddr3_dqs_p	[7:0]	Inout	Must be connected to the DDR3 memory differential data strobe ports.
ddr3_dqs_n	[7:0]	Inout	
ddr3_ck_p	[1:0]	Out	Must be connected to the DDR3 memory differential clock ports.
ddr3_ck_n	[1:0]	Out	
iodelay_ctr_lrdy	–	In	Indicates whether IDELAYCTRLs are ready. Must be connected to the Perseus MPMC MPMC_Idelayctrl_Rdy_O port. If the MPMC is not instantiated in the design, the user must manually instantiate the IDELAYCTRLs.
o_MemClk_p	–	Out	200-MHz memory controller clock.

Table 7 Memory controller interface ports description

3.7 Core Registers

3.7.1 Register Map

Register name	Address from offset	Direction	Description
INFO	0×00	R	Core ID and version information
CONTROL	0x04	R/W	Resets the Record/Playback core and specifies memory initialization state.
SET_MODE	0x08	R/W	Latches the mode value.
MODE_VALUE	0×0C	R/W	Specifies the active mode.
SET_START_ADDRESS	0×10	R/W	Latches the start address value.
START_ADDRESS	0×14	R/W	Specifies the start address in the memory.

Register name	Address from offset	Direction	Description
DELAY_VAL	0×18	R/W	Holds the delay value.
SET_TRANSFER_SIZE	0x1C	R/W	Latches the transfer size value.
TRANSFER_SIZE	0x20	R/W	Specifies the transfer size.
TRIG_ADDRESS	0x24	R	Specifies the trigger address.
TRIG_ADDRESS_INDEX	0x28	R	Specifies the offset from the trigger address.
TRANSFER_OVER	0x2C	R	Specifies whether the transfer is over.
MODE_RECORD	0x30	R	Holds the numerical value of the Record mode.
MODE_PLAYBACKSINGLE	0x34	R	Holds the numerical value of the Playback mode.
MODE_PLAYBACKCONTINUOUS	0x38	R	Holds the numerical value of the Continuous Playback mode.
MODE_RTDEXMEM2HOST	0x3C	R	Holds the numerical value of the memory-to-Host mode
MODE_RTDEXHOST2MEM	0x40	R	Holds the numerical value of the host-to-memory mode.
Reserved	0x44	-	Reserved
Reserved	0x48	-	Reserved
PROG_FULL_THRES_ASSERT	0x4C	R/W	Hold the values of the programmable full threshold of the playback FIFO.
PROG_FULL_THRES_NEGATE	0x50	R/W	Hold the values of the programmable full threshold of the playback FIFO.
PARITY_ADDR_REG	0×54	R	Holds the value of the trigger parity indicator for record address decoding.
NB_RECORD_PORTS	0x58	R	Number of instantiated record ports
NB_PLAYBACK_PORTS	0x5C	R	Number of instantiated playback ports
RECORD_PORTS_WIDTH	0x60	R	Width of the instantiated record ports
PLAYBACK_PORTS_WIDTH	0x64	R	Width of the instantiated playback ports
RECORD_TRIG_MUX	0x68	R/W	Specifies the Record trigger multiplexer position.
RECORD_SOFT_TRIG	0x6C	R/W	Allows the user to trigger a Record.
MODE_VALIDATION	0x70	R	Reserved for internal debugging
PLAYBACK_TRIG_MUX	0x74	R/W	Specifies the Playback trigger multiplexer position.
PLAYBACK_SOFT_TRIG	0x78	R/W	Allows the user to trigger a Playback.
DDR3_SIZE	0x7C	R/W	Register to store the DDR3 size in GB
RECORD_STORAGE_CNT	0x80	R	Counter register of the remaining 32-Byte words of the

Register name	Address from offset	Direction	Description
			Record transaction minus 1 word.
RTDEX_STORAGE_CNT	0x84	R	Counter register of the remaining 32-Byte words of the Host2Mem transaction minus 1 word.
PLAYBACK_READ_CNT	0x88	R	Counter register of the remaining 32-Byte words of the Playback transaction.
RTDEX_READ_CNT	0x8C	R	Counter register of the remaining 32-Byte words of the Mem2Host transaction.

Table 8 Core register map

3.7.2 Register Description

Offset 0x00 — CORE_RST

This register allows the user to apply a core level reset and a memory reset.

31 to 16	15 to 0
Core ID	**Core Version**
R	R
0xEA00	0x0201

Table 9 CORE_RST register

Bit	Description	Configuration
Core Version	Record Playback core version.	0x0201
Core ID	Record Playback core identification number.	0xEA00

Table 10 CORE_RST register fields description

Offset 0x04 — PHY_INIT_DONE

This register specifies the memory initialization state.

31 to 1	2	1	0
Reserved	**PhyInitDone**	**CoreReset**	**MemReset**
R	R	R/W	R/W
0	0	0	0

Table 11 PHY_INIT_DONE register

Bit	Description	Configuration
MemReset	Reset the DDR3 memory controler	1 = Reset 0 = Clear Reset
CoreReset	Reset the Record Playback core register and state machines	1 = Pulse Reset
PhyInitDone	Shows the init done bit status.	1 = PHY init done 0 = PHY init not done yet

Table 12 PHY_INIT_DONE register field description

Offset 0x08 — SET_MODE

This register latches the Record/Playback mode.

31 to 1	0
Reserved	**SetMode**
R	R/W
0	0

Table 13 SET_MODE register

Bit	Description	Configuration
SetMode	Latches the mode from the MODE_VALUE register.	Writes 1 to the latch mode. This bit is self cleared.

Table 14 SET_MODE register field description

Offset 0x0C — MODE_VALUE

This register allows the user to set the Record/Playback mode.

31 to 3	2 to 0
Reserved	**ModeValue**
R	R/W
0	0

Table 15 MODE_VALUE register

Bit	Description	Configuration
ModeValue	Specifies the current Record/Playback mode.	0 = No mode selected 1 = Record mode 2 = Single Playback mode 3 = Continuous Playback mode 4 = RTDEx MEM_TO_HOST mode 5 = RTDEx HOST_TO_MEM mode

Table 16 MODE_VALUE register field description

Offset 0x10 — SET_START_ADDRESS

This register latches the start address.

31 to 1	0
Reserved	**SetStartAddress**
R	R/W
0	0

Table 17 SET_START_ADDRESS register

Bit	Description	Configuration
SetStartAddress	Latches the start address from the START_ADDRESS register.	Writes 1 to the latch address. This bit is self cleared.

Table 18 SET_START register field description

Offset 0x14 — START_ADDRESS

This register allows the user to set the start address. The address is a multiple of 64 bits and must be aligned to point to a quad octaword (512 bits). This means that the address value is always a multiple of 8 (512/64). For example, the start address can be used to control where the recording will start in the memory. It is usually set to 0 to start at the beginning of the memory. If an address higher than the memory size is specified, its value will wrap and the StartAddress will point to an address inside the memory space.

31 to 30	29 to 0
Reserved	**StartAddress**
R	R/W
0	0

Table 19 START_ADDRESS register

Bits	Description	Configuration
StartAddress	Specifies the start address.	Writes the start address.

Table 20 START_ADDRESS register field description

Offset 0x18 — DELAY_VAL

This register allows the user to set the delay value. The delay value allows controlling the recorded zone according to the trigger event position. When this value is 0, the recording stops after the specified transfer size is reached from the trigger event. When this value is > 0, the recording will continue after the trigger event until the amount of data captured reaches the value of the specified transfer size plus the specified trigger delay. The specified value must be in quad octaword (512 bits).

To calculate the number of sample clock periods represented by a given delay, use this formula:

Delay in sample clock periods from trigger event = valueof(DelayVal) * 512 / (valueof(NB_RECORD_PORTS) * valueof(RECORD_PORTS_WIDTH))

31	30 to 0
Reserved	**DelayVal**
R	R/W
0	0

Table 21 DELAY_VAL register

Bits	Description	Configuration
DelayVal	Specifies the delay value.	Writes the delay value.

Table 22 DELAY_VAL register field description

Offset 0x1C — SET_TRANSFER_SIZE

This register latches the transfer size.

31 to 1	0
Reserved	**SetTransferSize**
R	R/W
0	0

Table 23 SET_TRANSFER_SIZE register

Bit	Description	Configuration
SetTransferSize	Latches the transfer size from the TRANSFER_SIZE register.	Writes 1 to the transfer size. This bit is self cleared.

Table 24 SET_TRANSFER_SIZE register field description

Offset 0x20 — TRANSFER_SIZE

This register allows the user to set the transfer size in bytes. The transfer size is specified in terms of a multiple of 64 bits and the desired size must be a quad octaword (512 bits = 64 bytes) multiple. This means that the transfer size value is always a multiple of 8 (512/64).

31 to 30	29 to 0
Reserved	**TransferSize**
R	R/W
0	0

Table 25 TRANSFER_SIZE register

Bits	Description	Configuration
TransferSize	Specifies the transfer size.	Writes the transfer size.

Table 26 TRANSFER_SIZE register fields description

Offset 0x24 — TRIG_ADDRESS

This register contains the memory address block where there was a rising trigger (when valueof(DELAY_VAL) = 0). The address is a multiple of 64 bits and is aligned to point to a quad octaword (512 bits). This means that the address value is always a multiple of 8 (512/64). When valueof(DELAY_VAL) is not 0, this contains the memory address block from the trigger event after the delay has elapsed (see the description of the DELAY_VAL register for details). For example, this register can be used to transfer data to the host according to the trigger event and the specified delay.

Because the trigger address is a 30-bit counter, it is possible than the value exceed the memory size. When apply a bit mask or a modulo operation, the real address can be retrieve:

Example for 1 GB memory: `TrigAddr & 0x0FFFFFFF` or `TrigAddr % (1<<28)`

31 to 30	29 to 0
Reserved	**TrigAddr**
R	R
0	0

Table 27 TRIG_ADDRESS register

Offset 0x28 — TRIG_ADDRESS_INDEX

This register contains the byte index in the form of a double octaword (256 bits) of the memory block pointed by valueof(TRIG_ADDRESS) where there was a rising trigger (when valueof(DELAY_VAL) = 0)). For example, when the transfer is made with the host starting from the valueof(TRIG_ADDRESS) position, this register allows the calculation of the effective byte index from the host buffer memory where the trigger event happened. To calculate this effective index in bytes, use this formula:

Effective bytes index = ((valueof(PARITY_ADDR_REG) × 32) + 32 - Log2(valueof(TRIG_ADDRESS_INDEX)) – valueof(NB_RECORD_PORTS) * valueof(RECORD_PORTS_WIDTH) / 8)

31 to 29	28 to 0
Reserved	**TrigAddrIndex**
R	R
0	0

Table 28 TRIG_ADDRESS register fields description

Offset 0x2C — TRANSFER_OVER

This register specifies whether the current transfer is over or not.

31 to 0	0
Reserved	**TransferOver**
R	R
0	0

Table 29 TRANSFER_OVER register

Bit	Description	Configuration
TransferOver	Specifies whether the current transfer is over or not.	1 if the transfer is over.

Table 30 TRANSFER_SIZE register field description

Offset 0x30 — MODE_RECORD

This register contains the constant numerical value of the Record mode. This value can be specified in the MODE_VALUE register to start writing in the memory from the recording ports.

31 to 3	2 to 0
Reserved	ModeRecordValue
R	R
0	1

Table 31 MODE_RECORD register

Offset 0x34 — MODE_PLAYBACKSINGLE

This register contains the constant numerical value of the Single Playback mode. This value can be specified in the MODE_VALUE register to start a single read from the memory to the playback ports.

31 to 3	2 to 0
Reserved	ModePlaybackSingle
R	R
0	2

Table 32 MODE_PLAYBACKSINGLE register

Offset 0x38 — MODE_PLAYBACKCONTINUOUS

This register contains the constant numerical value of the Continuous Playback mode. This value can be specified in the MODE_VALUE register to start a continuous read from the memory to the playback ports.

31 to 3	2 to 0
Reserved	ModePlaybackContinuous
R	R
0	3

Table 33 MODE_PLAYBACKCONTINUOUS register

Offset 0x3C — MODE_RTDEXMEM2HOST

This register contains the constant numerical value of the memory-to-RTDEx mode. This value can be specified in the MODE_VALUE register to start reading from the memory to the RTDEx.

31 to 3	2 to 0
Reserved	ModeRTDEx2HostValue
R	R
0	4

Table 34 MODE_RTDEXMEM2HOST register

Offset 0x40 — MODE_RTDEXHOST2MEM

This register contains the constant numerical value of the RTDEx-to-memory mode. This value can be specified in the MODE_VALUE register to start writing from the RTDEx to the memory.

31 to 3	2 to 0
Reserved	ModeHost2RTDExValue
R	R
0	5

Table 35 MODE_RTDEXHOST2MEM register

Offset 0x4C — PROG_FULL_THRES_ASSERT

This register specifies the full assert threshold value of the playback FIFO.

31 to 8	7 to 0
Reserved	ProgFullThresAssert
R	R/W
0	0

Table 36 PROG_FULL_THRES_ASSERT register

Bit	Description	Configuration
ProgFullThresAssert	Specifies the full assert threshold value of the playback FIFO.	Usually set to 32.

Table 37 PROG_FULL_THRES_ASSERT register field description

Offset 0x50 PROG_FULL_THRES_NEGATE

This register specifies the full Negate threshold value of the playback FIFO.

31 to 8	7 to 0
Reserved	ProgFullThresNegate
R	R/W
0	0

Table 38 PROG_FULL_THRES_NEGATE register

Bit	Description	Configuration
ProgFullThresNegate	Specifies the full negate threshold value of the playback FIFO.	Usually set to 32.

Table 39 PROG_FULL_THRES_NEGATE register field description

Offset 0x58 NB_RECORD_PORTS

This register specifies the number of instantiated record ports.

31 to 0
NbRecordPorts
R
0

Table 40 NB_RECORD_PORTS register

Bit	Description	Configuration
NbRecordPorts	Number of instantiated record ports	1 to 8 ports

Table 41 NB_RECORD_PORTS register fields description

Offset 0x5C NB_PLAYBACK_PORTS

This register specifies the number of instantiated playback ports.

31 to 0
NbPlaybackPorts
R
0

Table 42 NB_PLAYBACK_PORTS register

Bit	Description	Configuration
NbPlaybackPorts	Number of instantiated playback ports	1 to 8 ports

Table 43 NB_PLAYBACK_PORTS register field description

Offset 0x60 RECORD_PORTS_WIDTH

This register specifies the width of the instantiated record ports.

31 to 0
RecordPortsWidth
R
0

Table 44 RECORD_PORTS_WIDTH register

Bit	Description	Configuration
RecordPortsWidth	Width of the record ports	8, 16, 32, 64 bits

Table 45 RECORD_PORTS_WIDTH register field description

Offset 0x64 PLAYBACK_PORTS_WIDTH

This register specifies the width of the instantiated playback ports.

31 to 0
PlaybackPortsWidth
R/W
0

Table 46 PLAYBACK_PORTS_WIDTH register

Bit	Description	Configuration
PlaybackPortsWidth	Width of the playback ports	8, 16, 32, 64 bits

Table 47 PLAYBACK_PORTS_WIDTH register field description

Offset 0x68 — RECORD_TRIG_MUX

This register specifies whether the record module uses a hardware or software trigger.

31 to 1	0
Reserved	**RecTrigMux**
R	R/W
0	0

Table 48 RECORD_TRIG_MUX register

Bit	Description	Configuration
RecTrigMux	Specifies the position of the record trigger multiplexer.	0 for hardware trigger 1 for software trigger

Table 49 RECORD_TRIG_MUX register field description

Offset 0x6C — RECORD_SOFT_TRIGGER

This register allows the user to trigger a record transfer.

31 to 1	0
Reserved	**RecSoftTrig**
R	R/W
0	0

Table 50 RECORD_SOFT_TRIGGER register

Bit	Description	Configuration
RecSoftTrig	Triggers a record transfer.	1 to trigger

Table 51 RECORD_SOFT_TRIGGER register field description

Offset 0x70 — MODE_VALIDATION

Reserved for debug use.

31 to 1	0
Reserved	**ModeValidationValue**
R	R
0	0

Table 52 MODE_VALIDATION register

Bit	Description	Configuration
ModeValidationValue	Reserved for debug use.	Reserved for debug use.

Table 53 MODE_VALIDATION register field description

Offset 0x74 — PLAYBACK_TRIG_MUX

This register specifies whether the playback module uses a hardware or software trigger.

31 to 1	0
Reserved	**PlayTrigMux**
R	R/W
0	0

Table 54 PLAYBACK_TRIG_MUX register

Bit	Description	Configuration
PlayTrigMux	Specifies the position of the playback trigger multiplexer.	0 for hardware trigger 1 for software trigger

Table 55 PLAYBACK_TRIG_MUX register field description

Offset 0x78 — PLAYBACK_SOFT_TRIGGER

This register allows the user to trigger a playback transfer.

31 to 1	0
Reserved	**PlaySoftTrig**
R	R/W
0	0

Table 56 PLAYBACK_SOFT_TRIGGER register

Bit	Description	Configuration
PlaySoftTrig	Triggers a playback transfer.	1 to trigger

Table 57 PLAYBACK_SOFT_TRIGGER register field description

Offset 0x7C — DDR3_SIZE

This register allows the user to write and read the Register to store the DDR3 size in GB. Once the DDR3 size has been identified, it is written in this register for easier and quicker access to the DDR3 size.

31 to 0
Ddr3Size
R/W
0

Table 58 DDR3_SIZE register

Bit	Description	Configuration
Ddr3Size	Register to store the DDR3 size in GB	Size in GB

Table 59 DDR3_SIZE register field description

Offset 0x80 — RECORD_STORAGE_CNT

This register allows the user to read the counter register of the remaining 32-Byte words of the Record transaction minus 1 word.

31 to 0
RecordStorageCnt
R
0

Table 60 RECORD_STORAGE_CNT register

Bit	Description	Configuration
RecordStorageCnt	Counter register of the remaining 32-Byte words of the Record transaction minus 1 word	Number of 32-byte word remaining - 1

Table 61 RECORD_STORAGE_CNT register field description

Offset 0x84 — RTDEX_STORAGE_CNT

This register allows the user to read the counter register of the remaining 32-Byte words of the Host2Mem transaction minus 1 word.

31 to 0
RtdexStorageCnt
R
0

Table 62 RTDEX_STORAGE_CNT register

Bit	Description	Configuration
RtdexStorageCnt	Counter register of the remaining 32-Byte words of the Host2Mem transaction minus 1 word	Number of 32-byte word remaining - 1

Table 63 RTDEX_STORAGE_CNT register field description

Offset 0x88 — PLAYBACK_READ_CNT

This register allows the user to read the counter register of the remaining 32-Byte words of the Playback transaction.

31 to 0
PlaybackReadCnt
R
0

Table 64 PLAYBACK_READ_CNTregister

Bit	Description	Configuration
PlaybackReadCnt	Counter register of the remaining 32-Byte words of the Playback transaction.	Number of 32-byte word remaining

Table 65 PLAYBACK_READ_CNT register field description

Offset 0x8C — RTDEX_READ_CNT

This register allows the user to read the counter register of the remaining 32-Byte words of the Mem2Host transaction.

31 to 0
RtdexReadCnt
R
0

Table 66 RTDEX_READ_CNT register

Bit	Description	Configuration
RtdexReadCnt	Counter register of the remaining 32-Byte words of the Mem2Host transaction.	Number of 32-byte word remaining

Table 67 RTDEX_READ_CNT register field description

4 FPGA Core Interfacing

4.1 Using the Record/Playback Core

The user interface is a FIFO-based interface. The user provides the core with a write/read clock, data, and control signals.

An example is provided to show how a user can interface to this core. The link to this example is located in the "Record/Playback Example" section of this document.

The Record/Playback core needs to be connected to the RTDEx core for data transfer with the host. A dedicated interface is used for that purpose.

The following figures describe examples timing diagram of the user ports of the Record/Playback core.

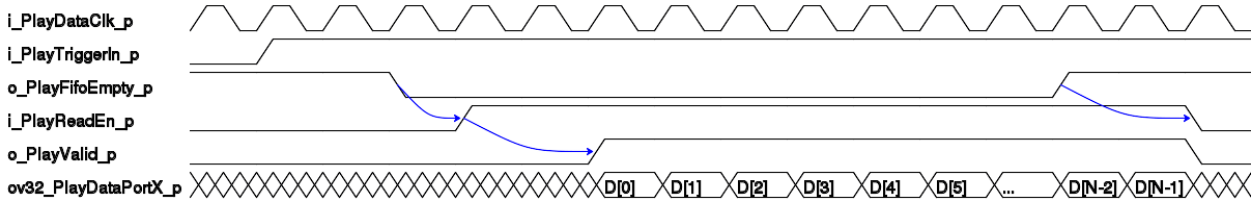

Figure 4-1 Playback user port timing

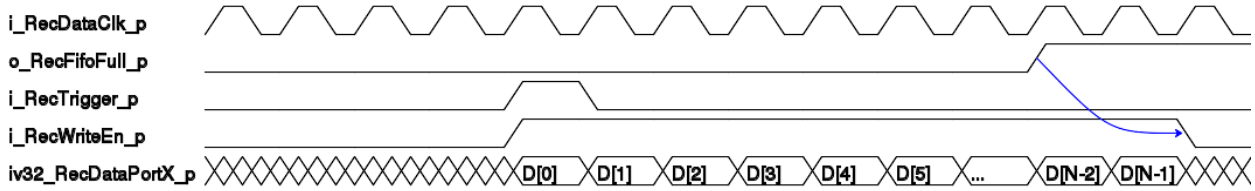

Figure 4-2 Record user port timing

5 Host Programming

5.1 Record/Playback Library

The Record/Playback library is built to allow full control over the memory transfers and to interface with the RTDEx library for memory-to-host and host-to-memory transfers. Each function uses a handle created by the open function. The table below presents a list of the main functions of the Record/Playback library.

For a complete documentation, please refer to the Nutaq's API document.

Function	Description
RecPlay_Open	Creates a RecPlay object and returns a handle to this object.
RecPlayResetCore	Resets the Record/Playback FPGA core.
RecPlayResetMemory	Resets the FPGA onboard memory.
RecPlayGetMemoryInitDoneStatus	Returns the memory initialization status.
RecPlaySetModeRecord	Sets the core mode to Record.
RecPlaySetModePlaybackSingle	Sets the core mode to Single Playback.
RecPlaySetModePlaybackContinuous	Sets the core mode to Continuous Playback.
RecPlaySetModeRtdexHost2Mem	Sets the core mode to Host2Mem.
RecPlaySetModeRtdexMem2Host	Sets the core mode to Mem2Host.
RecPlaySetTriggerSource	Sets the Record trigger source.
RecPlaySoftTrig	Executes a software trigger for the Record.
PlaybackSetTriggerSource	Sets the Playback trigger source.
PlaybackSoftTrig	Executes a software trigger for the Playback.
RecPlayGetTransferStatus	Verifies if transfer is over.
RecPlayGetTriggerPos	Verifies the Record trigger address.
RecPlay_Close	Destroys a RecPlay object.

Table 68 Record/Playback library functions

5.2 Command Line Interface (CLI) Commands

The Record/Playback CLI commands use glue logic implemented on top of the Record/Playback library to offer an easy approach for executing memory transfers.

The following is a list of the Record/Playback CLI commands. For complete documentation refer to the Nutaq's *Board & Systems Software Tools -Command Line Interface API* guide.

Command	Description
ram_put	Sends data from a host device to the Perseus through RTDEx and writes it to the DDR3 SDRAM. Frame size is in bytes. The transfer size will be the same as the file size.
ram_get	Reads the data from the Perseus DDR3 SDRAM and sends it to a host device through RTDEx. Transfer size and frame size are in bytes.
recplay_record	Records data in the Perseus DDR3 SDRAM. The size is specified in bytes and must be a multiple of 8.
recplay_playback_single	Plays back data from the Perseus DDR3 SDRAM. The size is specified in bytes and must be a multiple of 8. The playback stops once the size has been reached.
recplay_playback_continuous	Plays back data from the Perseus DDR3 SDRAM. The size is specified in bytes and must be a multiple of 8. The playback restarts infinitely once the size has been reached.

Table 69 Record/Playback CLI commands

5.3 MBDK Programming and Interfacing

5.3.1 FPGA MBDK

You will find the documentation on the Record/Playback FPGA MBDK block in the following document:

%BASROOT%\doc\html\mbdk_fpga_sdram_record_and_playback.htm

5.3.2 HOST MBDK

You will find the documentation on the Record/Playback Host MBDK block in the following document:

%BASROOT%\doc\html\mbdk_simulink_sdram_record_and_playback.htm

6 Host Applications Using the Record/Playback EAPI Library

6.1 Building Your Own Host Application Using Record/Playback

The first step in building a Record/Playback application is creating a RecPlay object using the open function. All the other functions use the handle returned of that function. The programmer must then reset the memory and the FPGA core. The following diagram illustrates the flow of operation of a Record/Playback application.

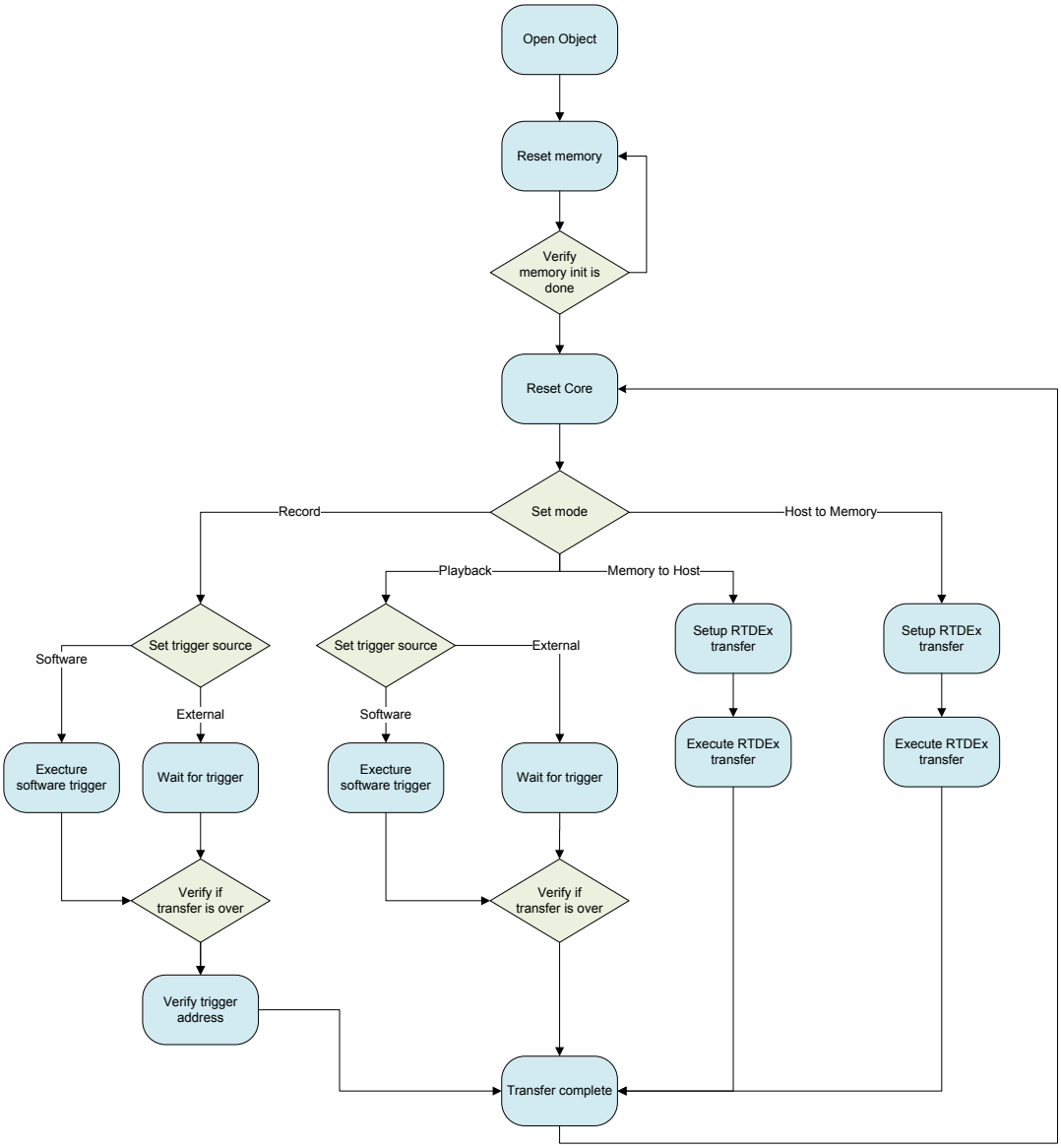

Figure 6-1 Record/Playback application diagram

6.1.1 Typical Application: Recording and Retrieving Data to and from the Host PC

A typical application of the Record/Playback consists of recording data to the memory, and then using the memory–to-host transfer to retrieve that data and send it to the host. The trigger address read at the end of the Record transfer should be used as the start address in the setup of the memory–to-host transfer. Once the data is transferred to the host, the byte index in the host buffer can be calculated where the trigger (+delay) event happened.

6.1.2 Typical Application: Sending Data to the Device and Using the Playback to Feed it to the User Logic

A typical application of the Record/Playback is sending data to the FPGA, and then using Playback to feed it to the user logic (refer to Figure 6-1). The start address used in this case can be 0 for both the host-to-memory and the Playback transfers. The host-to-memory will store the received data beginning at the specified start address and the Playback will be able to retrieve this data from the same memory address.

The BAS software suite provides five example utility applications that the user can modify. Those are explained in section 6.2.

6.2 Using and Modifying the Record/Playback Example Applications

Five example utility applications are provided with your BAS software suite, with pre-compiled executables made available in your *%BASROOT%**tools**bin* directory, and project folders available in your *%BASROOT%**tools**apps**core* directory. All five applications use the Record/Playback core instantiated in your FPGA design and the onboard DDR3 memory.

- *RecordData* is used to record data from the user logic and transfer it to the Perseus memory.
- *RetrieveRecordedData* is used to transfer data from the Perseus memory to the host, through Gigabit Ethernet or PCI-Express.
- *LoadDataToPlayback* is used to transfer data from the host to the Perseus memory, through Gigabit Ethernet or PCI-Express. Data will then be available for playback.
- *PlaybackData* is used to configure the Record/Playback core to transfer data from the Perseus memory to the user logic.
- *PlaybackStop* is used to stop an ongoing data transfer from the Perseus memory to the user logic.

Various examples showcase how these five applications are used and are available in your *%BASROOT%**examples* directory.

- Of those, two applicative examples directly demonstrate the Record/Playback core and their utility functions. They reside in *%BASROOT%**examples**rtdex_recplay**host**scripts*.
    1. *RecordTest* configures the Record/Playback Test core to generate data. It is then recorded by the Record/Playback core and verified on the host PC.
    2. *PlaybackTest* transfers data from the host to the carrier memory. This data is then playbacked by the Record/Playback core and verified by the Record/Playback Test core.

  Please refer to the Record Playback Examples Guide to run those applicative examples.
- Other examples demonstrating the use of Nutaq's FMCs also use the Record/Playback FPGA core and the Record/Playback Example Applications. Please refer to the example guide inherent to a given FMC to run a demo using the Record/Playback Example Applications with this FMC.

6.2.1 RecordData description

Usage: *RecordData <ip address> (0 | 1 | 2) <record size> <start address> <trigger delay>*

- Parameter 1 is the Perseus IP address, ex 192.168.0.101
- Parameter 2 is the Trigger source; External = 0, Software = 1, No Trigger = 2
- Parameter 3 is the Record size in bytes
- Parameter 4 is the Start address in bytes
- Parameter 5 is the Trigger delay in bytes

This application configures the Record/Playback core to record data from the user logic and store it in the Perseus memory.

1. It opens a record/playback object on the host.
2. It initializes the memory of the Perseus using `RecPlayGetMemoryInitDoneStatus()`.
3. The Record/Playback core is then reset using `RecPlayResetCore()`. This is needed to empty the core's FIFOs and to set the core to its initial state.
4. The trigger source is then selected using `RecordSetTriggerSource()`. Three possibilities exist.
    a. `eRecPlayTrigExternal`: The record may be triggered via a signal external to the Record/Playback core.
    b. `eRecPlayTrigSoft`: It may be triggered via a signal internal to the core. In this case, the `RecordSoftTrig()` function will be called to trig the record
    c. `eRecPlayTrigNone`: The Record/Playback uses no trigger signal and starts recording as soon as valid data is transmitted to it.
5. The record mode of the Record/Playback is then selected using `RecPlaySetModeRecord()`. This function is non-blocking and configures the Record/Playback core to start recording either when
    a. the trigger event occurs if the trigger source is external or software
    b. or as soon as valid data is transmitted to it if the "no trigger mode" (`eRecPlayTrigNone`) is selected.

    depending on the trigger source selected when calling `RecordSetTriggerSource()`. The Record/Playback core will stop recording when `u32RecordSize` bytes have been transmitted to the Perseus memory.
6. A software trigger is then sent using `RecordSoftTrig()` if the selected trigger source is software.

NOTE:

This application returns as soon as the Record/Playback core is configured to record data as per the chosen settings. It does not wait until the Record/Playback core has done recording the requested number of bytes to exit. The `RecPlayWaitTransferDone()` function (not used in this application) has to be called before attempting to retrieve the recorded data from the carrier memory.

6.2.2 RetrieveRecordedData description

Usage: *RetrieveRecordedData <ip address> (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7) <frame size> <record size> <start address> <record timeout> <file name>*

- Parameter 1 is the Perseus IP address, ex 192.168.0.101
- Parameter 2 is the RTDEx channel number. This refers to the RTDEx channel used in the FPGA design to transmit data between the host and the Record/Playback core.
- Parameter 3 is the Frame size in bytes. This is the frame size used by the RTDEx channel to transmit data between the host and the Record/Playback core.
- Parameter 4 is the size, in bytes, of the data to retrieve from memory
- Parameter 5 is the Start address in bytes, or the word "trigger"
- Parameter 6 is the Record timeout in milliseconds
- Parameter 7 is the File name, ex sinewave.bin

This application configures the Record/Playback core and uses RTDEx to retrieve data from the carrier and transfer it to the host.

To do so, the application either directly calls `RetrieveRecordedDataToFile()` or function `RetrieveRecordedDataFromTriggerToFile()`, if argument *<start address>* is 'trigger'. These functions are both helper functions, and they both ultimately lead to a call to `RetrieveRecordedData()`, which performs the actual transfer from the carrier memory to the host.

Transfers between the carrier memory and the Record/Playback core may only be done in complete chunks of 512 bits (64 bytes) from the memory, and transfers from the Record/Playback core to the host using RTDEx use fixed frame sizes (Ethernet or PCI-express frames). So, `RetrieveRecordedData()` calculates real transfer parameters based on the arguments passed to it and taking into account these technical requirements.

Transfer parameters include

- the number of the total number of bytes that will be transferred from the memory (`uRTDExTranferSize`)
- `uRecStartAddress` and `uRecStopAddress`, which are the real start and stop addresses the application will use to get data from memory.

Using function `RecPlaySetModeRtdexMem2Host()`, it then configures the Record/Playback core to send data to RTDEx. Then, it uses RTDEx to transfer the required 512-bit chunks from the Record/Playback core to the host PC.

Copy to the data buffer on the PC is then done according to the actual user parameters passed to the function, regardless of non-pertinent data transferred because of the 512-bit chunks requirements. To do this, the application uses parameters previously calculated such as

- `uOffsetStartRec`, the number of bytes between `uRecStartAddress` (the actual start address of the first chunk of data transferred from memory to host) and `u32StartAddress` (the start address passed to the application by the user).
- `uOffsetStopRec`, the number of bytes between `uRecStopAddress` (the actual stop address of the last chunk of data transferred from memory to host) and `u32StopAddress` (the stop address passed to the application by the user).

The application then finally copies the data buffer into the file specified by *<file name>*.

6.2.3 LoadDataToPlayback

Usage: *LoadDataToPlayback <ip address> (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7) <frame size> <start address> <file name>*

- Parameter 1 is the Perseus IP address, ex 192.168.0.101
- Parameter 2 is the RTDEx channel number. This refers to the RTDEx channel used in the FPGA design to transmit data between the host and the Record/Playback core.
- Parameter 3 is the Frame size in bytes. This is the frame size used by the RTDEx channel to transmit data between the host and the Record/Playback core.
- Parameter 4 is the Start Address in bytes
- Parameter 5 is the File name, ex sinewave.bin

This application sets a data transfer from a file on the host PC to the carrier memory, using RTDEx and the Record/Playback core. Data is placed beginning at address specified by *<start address>*.

The principal function of this application is `LoadDataToPlayback()` and is the one actually performing the data transfer.

1. Using function `RecPlaySetModeRtdexHost2Mem()`, the Record/Playback core is configured to receive data from RTDEx.
2. Data is transferred directly with one call to `RTDExSend()`.
3. The application waits until the transfer is done before exiting, using function `RecPlayWaitTransferDone()`.

6.2.4 PlaybackData

Usage: *PlaybackData <ip address> (0 | 1 | 2) (0 | 1) <playback size> <start address>*

- Parameter 1 is the Perseus IP address, ex 192.168.0.101
- Parameter 2 is the Trigger source; External = 0, Software = 1, No Trigger = 2
- Parameter 3 is the Playback mode; Single = 0, Continuous = 1
- Parameter 4 is the Playback size in bytes
- Parameter 5 is the Start address in bytes

This application configures the Record/Playback core to read data from the carrier memory, beginning at address specified by *<start address>* and transfer it to the user logic in the FPGA, in a playback fashion.

1. After creating a Record/Playback object on the host using function `RecPlay_Open()`, the application uses function `RecPlayWaitTransferDone()` to make sure any previous transfer to or from the carrier memory is completed.
2. The trigger source is then selected using `PlaybackSetTriggerSource()`. Three possibilities exist.
    a. `eRecPlayTrigExternal`: The playback may be triggered via a signal external to the Record/Playback core.
    b. `eRecPlayTrigSoft`: It may be triggered via a signal internal to the core. In this case, the `RecordSoftTrig()` function will be called to trig the playback
    c. `eRecPlayTrigNone`: The Record/Playback uses no trigger signal and starts playbacking as soon as the `PlayReadEn` port of the core is set high.
3. The playback mode is selected according to the parameter passed to the function.
    a. 0: Single playback. The Record/Playback core playbacks u32PlaybackSize bytes from carrier memory and then stops
    b. 1: Continuous playback. The Record/Playback core continuously playbacks u32PlaybackSize bytes from carrier memory, in a looping fashion.

4. A software trigger is then sent using `PlaybackSoftTrig()` if the selected trigger source is software.

6.2.5 PlaybackStop

Usage: *PlaybackStop <ip address>*

- Parameter 1 is the Perseus IP address.

This application calls `RecPlayResetCore()` to reset the core and stop an ongoing transfer from memory to user logic.

6.2.6 Exploring the Applications Source Code and Modifying/Rebuilding it

Project folders, allowing the user to modify the source code of the applications are also available. They reside in *%BASROOT%\tools\apps\core*.

If rebuilding is needed, follow the steps described in the section appropriate to the operating system you are using, where <*application to rebuild*> is either

- *RecordData,*
- *RetrieveRecordedData,*
- *LoadDataToPlayback,*
- *PlaybackData* or
- *PlaybackStop*

6.2.6.1 Windows

A folder named *prj_win* which contains the Visual Studio 2012 solution associated with the application is present in the folders listed above.

1. Start Microsoft Visual Studio.
2. On the **File** menu, point to **Open** and click **Project/Solution**.
3. Browse to *the %BASROOT%\tools\apps\core\<application to rebuild>* folder and select the <*application to rebuild*>*.sln* solution
4. Select the build configuration Release x64.
5. On the **Build** menu, click **Build Solution**.

6.2.6.2 Linux

A folder named *prj_linux* which contains a shell script and a makefile associated with the application is present in the folders listed above. Source files are available in the *inc* and *src* folders

1. Open a terminal and use the *cd* command to browse to the *%BASROOT%\tools\apps\core\<application to rebuild* > repository in the installation.
2. To build the application, run the following command in the Linux terminal.

   sudo ./build_demo.sh