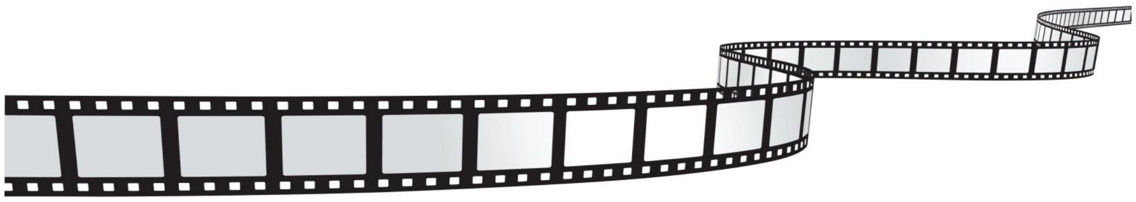
Leon Utility Functions

18.08.10



Contents

Chapter 1

Introduction

This document describes the Leon Utilities provided with Myriad2.

Chapter 2

Deprecated List

Global swcShaveProfStopFieldsGatehering (u32 shaveNumber, performanceCounterDef perf-Defines) __Deprecated__("Use swcShaveProfStopFieldsGathering instead")

This function is deprecated. Use swcShaveProfStopFieldsGathering instead.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Module Documentation

6.1 Debug Tracer

Debug Tracer module API.

Macros

- #define DEBUG_LOG_LEVEL_LOW LOG_LEVEL_INFO
- #define DEBUG_LOG_LEVEL_MEDIUM LOG_LEVEL_WARNING
- #define DEBUG_LOG_LEVEL_HIGH LOG_LEVEL_ERROR

6.1.1 Detailed Description

Debug Tracer module API. Header abstract API for debug trace logging

6.1.2 Macro Definition Documentation

#define DEBUG_LOG_LEVEL_HIGH **LOG_LEVEL_ERROR**

#define DEBUG_LOG_LEVEL_LOW **LOG_LEVEL_INFO**

#define DEBUG_LOG_LEVEL_MEDIUM **LOG_LEVEL_WARNING**

6.2 Memory Transfer

Memory Transfer module API.

Functions

- void swcU32memcpy (u32 ∗dst, u32 ∗src, u32 len)

  Function that copies from source to destination.
- void swcU32memsetU32 (u32 ∗addr, u32 lenB, u32 val)

  Function that sets memory with a givven value.

6.2.1 Detailed Description

Memory Transfer module API. Used for manipulating memory transfers

6.2.2 Function Documentation

void swcU32memcpy (u32 ∗ dst, u32 ∗ src, u32 len)

Function that copies from source to destination.

Parameters

`in`	-	Destination address
`in`	-	Source address
`in`	-	Length to copy

Returns

void

void swcU32memsetU32 (u32 ∗ addr, u32 lenB, u32 val)

Function that sets memory with a givven value.

Parameters

`in`	-	Destination address
`in`	-	Length to copy
`in`	-	Value to set

Returns

void

6.3 Shave Loader

API for the Shave Loader module.

Macros

- #define ADDR_DDRL2(x) (((u32)(x)) & 0xF0FFFFFF)

  use DDR address through L2 cache. Force it's use.
- #define ACCEPT_ALTERNATIVE_SHAVE_START_METHOD FALSE
- #define SHAVE_INTERRUPT_LEVEL 3

Typedefs

- typedef u32 swcShaveUnit_t

Enumerations

- enum context_t { SHVXDATA = 0, SHVZDATA, SHVDLIB }

Functions

- void swcSetAbsoluteDefaultStack (u32 shave_num)

  Set absolute default stack for a specific shave.
- void swcStateConsideredShaveStackSize (u32 shaveNumber, u32 size)

  Allows the user to assert a stack size against which checks may be implemented. This does not represent a guarantee that the system will allocate this stack it only allows users to specify how much space they themselves have considered and made available through other means for the application. Calling this function allows the system to perform checks which would detect if this size was overrun at any stage.
- u32 swcGetShaveStackSize (u32 shaveNumber)

  Reads back the stack size for a specified shave. When calling either swcSetAbsoluteDefaultStack or swcSetWindowedDefaultStack the stack size set to register i20 will be stored and can be read back with the help of this function.
- u32 swcGetUnusedShaveFreeStack (u32 shaveNumber, u32 canaryValue)

  If stack painter was used, this function searches for the size of unused stack given pattern checks N-OTE!: this function does nothing relevant if user did not call swcStateConsideredShaveStackSize and swcStackPainter before running a shave application.
- void swcStackPainter (u32 shaveNumber, u32 canaryValue)

  Paint stack with a specific canary value. NOTE: one must have called the swcStateConsideredShave-StackSize on the shaveNumber used here in advance of calling this function.
- void swcGetShaveWindowRegs (u32 shaveNumber, u32 ∗windows)

  Get Shave window register values.
- void swcSetShaveWindow (u32 shave_num, u32 window_num, u32 window_addr)

  Set a specific window register with a target window base address.
- void swcSetShaveWindows (u32 shaveNumber, u32 windowA, u32 windowB, u32 windowC, u32 windowD)

  Set each window register with the corresponding window base address.
- void swcSetShaveWindowsToDefault (u32 shaveNumber)

Reset windows to default values in case they are rewritten by other shaves param[in] shaveNumber - shave number for which default value will be set.

- u32 swcShaveRunning (u32 svu)

  Check if a specific Shave is running or it is stopped.
- void swcRunShave (u32 shave_nr, u32 entry_point)

  Start shave shave_nr from entry_point.
- void swcStartShave (u32 shave_nr, u32 entry_point)

  Starts non blocking execution of a shave.
- void swcDynStartShave (u32 shave_nr, u32 Context)

  Starts non blocking execution of a shave using dynamic sub module alocator.
- void swcShaveStartAsync (u32 shave_nr, u32 entry_point, irq_handler function)

  Starts non blocking execution of a shave.
- void swcStartShaveAsync (u32 shave_nr, u32 entry_point, irq_handler function) __Deprecated_-_("Please use swcShaveStartAsync instead.")
- void swcDynShaveStartAsync (u32 shave_nr, u32 Context, irq_handler function)

  Starts dynamic non blocking execution of a shave. A master entry point is executed prior to jumping into shave entry point.
- void swcAssignShaveCallback (u32 shave_nr, irq_handler function)

  Assigns a callback to a shave for end of execution. Alternative way to the swcStartShaveAsync way of working.
- void swcSetRegsCC (u32 shave_num, const char ∗fmt, va_list a_list)
- void swcStartShaveCC (u32 shave_num, u32 pc, const char ∗fmt,...)

  Write the value to a IRF/SRF/VRF Registers from a specific Shave.
- void swcDisableShaveCallback (u32 shave_nr)

  Disables the interrupt for shave end. Useful for cases where the shave needs to be run for a few times in Async mode with interrupts but then the same shave needs to stop triggering interrupts.
- void swcStartShaveAsyncCC (u32 shave_num, u32 pc, irq_handler function, const char ∗fmt,...)

  Write the value to a IRF/SRF/VRF Registers from a specific Shave.
- void swcSetupShaveCC (u32 shave_num, const char ∗fmt,...)

  Write the value to a IRF/SRF/VRF Registers from a specific Shave.
- void swcSetRounding (u32 shave_no, u32 roundingBits)

  Function that starts one shave but at the same time also sets rounding bits.
- void swcResetShave (u32 shaveNumber)

  Reset shave.
- void swcResetShaveLite (u32 shaveNumber)

  Reset shave without resetting the fifo.
- int swcWaitShaves (u32 no_of_shaves, swcShaveUnit_t ∗shave_list)

  Function that waits for the shaves used to finish.
- int swcWaitShave (u32 shave_nr)

  Wait for a specific shave to finish execution.
- u32 swcShavesRunning (u32 first, u32 last)

  Check if a list of shaves is running or not.
- u32 swcShavesRunningArr (u32 arr[ ], u32 N)

  Check if a list of shaves stored in an array is running or not.
- u32 swcSolveShaveRelAddr (u32 vAddr, u32 shaveNumber)

  Translate windowed address into real physical address.

- void swcLoadMbin (u8 ∗sAddr, u32 targetS)

  Load a mbin file to a specific target address on shave.
- void swcSetWindowedDefaultStack (u32 shave_num)

  Sets a default value for stack.
- void swcLoadshvdlib (u8 ∗sAddr, u32 targetS)

  Dynamically load shvdlib file - These are elf object files stripped of any symbols.
- void swcLoadDynLibraryCacheAware (u8 ∗sAddr, u32 targetS, context_t contextType, u32 ∗vpProgrammedMemAddress, u32 ∗flushLength)

  Dynamically load library file and return start memory address and length that need to be flushed - These are elf object files stripped of any symbols.
- void swcLoadDynLibrary (u8 ∗sAddr, u32 targetS, context_t contextType)

  Dynamically load library file - These are elf object files stripped of any symbols.
- s32 swcRunShaveAlgo (DynamicContext_t ∗moduleStData, int ∗const shaveNumber)

  Sets up and launches one dynamic application instance. Uses the shaves preliminary assigned by user via function swcSetupDynShaveApps(). Allocates all necessary memory, loads the dynamic library, then starts the shave.
- s32 swcRunShaveAlgoCC (DynamicContext_t ∗moduleStData, int ∗const shaveNumber, const char ∗fmt,...)

  Sets up and launches one dynamic application instance. Uses the shaves preliminary assigned by user via function swcSetupDynShaveApps(). Allocates all necessary memory, loads the dynamic library, then starts the shave.
- s32 swcRunShaveAlgoOnAssignedShave (DynamicContext_t ∗moduleStData, u32 shaveNumber)

  Sets up and launches one dynamic application instance on a specifically requested SHAVE Uses the shaves preliminary assigned by user via function swcSetupDynShaveApps(). Allocates all necessary memory, loads the dynamic library, then starts the shave. Checks if the requested shave has bee configured in advance and if it is not running.
- s32 swcRunShaveAlgoOnAssignedShaveCC (DynamicContext_t ∗moduleStData, u32 shaveNumber, const char ∗fmt,...)

  Sets up and launches one dynamic application instance on a specifically requested SHAVE Uses the shaves preliminary assigned by user via function swcSetupDynShaveApps(). Allocates all necessary memory, loads the dynamic library, then starts the shave. Checks if the requested shave has bee configured in advance and if it is not running.
- s32 swcSetupDynShaveApps (DynamicContext_t ∗moduleStData, const swcShaveUnit_t ∗svuList, const uint32_t instances)

  This function allocates heap and group data memory for all configured instances of one application. It must be called prior to using swcRunShaveAlgo(). Can be used from both Leons. svuList below is not copied internally, instead just the pointer is assigned to an internal structure. Please ensure the svuList memory is alive until the call of swcCleanupDynShaveApps. Note: be careful about stack declared svuList.
- s32 swcCleanupDynShaveApps (DynamicContext_t ∗moduleStData)

  This function frees the heap and group data memory for all configured instances of one application. It can be called after usage of swcRunShaveAlgo(). Can be used from both Leons.
- s32 swcDynShaveAppSetWindows (DynamicContext_t ∗moduleStData, u32 cmxCriticalCodeSize)

  This function allows hinting how much code/data is desired to be allocated TODO: add functionality to precompute these sizes based on .textCrit size.
- u32 swcCheckFreeAndValidShave (DynamicContext_t ∗moduleStData, u32 shaveNumber)

  This function is used to check if the user has called a correct shave. We define "correct" as: configured to be used by the current dyncontext and not currently running.

- s32 swcRequestUnallocatedShaves (swcShaveUnit_t ∗svuList, u32 shavesNumber)

  This functions gives a list of unallocated shaves in the system.
- s32 swcGetUnallocatedShavesNumber (void)

  This function return the number of unallocated shave in the system.
- s32 swcCleanupDynShaveListApps (DynamicContext_t ∗mData, swcShaveUnit_t ∗svuList, uint32_t elementsNumber)

  This function frees the heap and group data memory for the specified instances of one application. Can be used from both Leons.
- void swcSetNewHeapLocation (DynamicContext_t ∗mData, unsigned char ∗newAddress, swcShaveUnit_t shaveNumber)

  This function set a new heap location for a specific shave. Can be used from both Leons.
- void swcSetNewAppDynDataLocation (DynamicContext_t ∗mData, unsigned char ∗newAddress, swcShaveUnit_t shaveNumber)

  This function set a new memory location where to load the application dynamic data. Can be used from both Leons.
- void swcSetGrpDynDataLocation (DynamicContext_t ∗mData, unsigned char ∗newAddress, swcShaveUnit_t shaveNumber)

  This function set a new memory location where to load the grup dynamic data. Can be used from both Leons.
- int swcIsoSetupShaveApplication (DynamicContext_t ∗moduleStData, swcShaveUnit_t ∗svuList, uint32_t shavesNumber, MISA_PARADIGM_TYPE paradigmType)

  This function allocates heap and group data memory for all configured instances of one application and loads the dynamic library. It must be called prior to using swcRunShaveAlgo(). Can be used from both Leons. Please ensure the svuList memory is alive until the call of swcCleanupDynShaveApps. Note: be careful about stack declared svuList.
- int swcStartEntryPointDC (DynamicContext_t ∗moduleStData, uint32_t shaveNumber, const char ∗functionName)

  This function launch a shave application with a specific function as entry point. Can be used from both Leons.
- int swcStartEntryPointDCCC (DynamicContext_t ∗moduleStData, uint32_t shaveNumber, const char ∗functionName, const char ∗fmt,...)

  This function launch a shave application with a specific function as entry point. Can be used from both Leons.
- int swcStartFC (DynamicContext_t ∗moduleStData, uint32_t shaveNumber)

  This function launch a shave application by calling the main function. Can be used from both Leons.
- int swcIsoCleanShaveApplication (DynamicContext_t ∗moduleStData, swcShaveUnit_t ∗svuList, uint32_t shavesNumber, MISA_PARADIGM_TYPE paradigmType)

  This function frees the heap and group data memory for all configured instances of one application. It can be called after usage of swcRunShaveAlgo(). Can be used from both Leons.

Shave dummy wrappers

- #define SVU(x) x
- #define IRF(x) x
- #define SRF(x) x
- #define VRF(x) x

6.3.1 Detailed Description

API for the Shave Loader module. Used for executing different functionalities on SHAVEs

6.3.2 Macro Definition Documentation

#define ACCEPT_ALTERNATIVE_SHAVE_START_METHOD FALSE

#define ADDR_DDRL2(x) (((u32)(x)) & 0xF0FFFFFF)

use DDR address through L2 cache. Force it's use.

#define IRF(x) x

#define SHAVE_INTERRUPT_LEVEL 3

#define SRF(x) x

#define SVU(x) x

#define VRF(x) x

6.3.3 Typedef Documentation

typedef u32 **swcShaveUnit_t**

6.3.4 Enumeration Type Documentation

enum **context_t**

Enumerator

SHVXDATA
SHVZDATA
SHVDLIB

6.3.5 Function Documentation

void swcAssignShaveCallback (u32 shave_nr, irq_handler function)

Assigns a callback to a shave for end of execution. Alternative way to the swcStartShaveAsync way of working.

Parameters

in	*shave_nr*	u32 shave number to start
in	*function*	to call when shave finished execution

Returns

void

u32 swcCheckFreeAndValidShave (**DynamicContext_t** ∗ moduleStData, u32 shaveNumber)

This function is used to check if the user has called a correct shave. We define "correct" as: configured to be used by the current dyncontext and not currently running.

Parameters

in	*moduleStData*	- DynamicContext_t pointer to application's module data structure
in	*shaveNumber*	- shave to be verified

Returns

0 if it is not a valid shave, or 1 if valid

s32 swcCleanupDynShaveApps (**DynamicContext_t** ∗ moduleStData)

This function frees the heap and group data memory for all configured instances of one application. It can be called after usage of swcRunShaveAlgo(). Can be used from both Leons.

Parameters

in	*moduleStData*	- DynamicContext_t pointer to application's module data structure

Returns

operation status

s32 swcCleanupDynShaveListApps (**DynamicContext_t** ∗ mData, **swcShaveUnit_t** ∗ svuList, uint32_t elementsNumber)

This function frees the heap and group data memory for the specified instances of one application. Can be used from both Leons.

Parameters

in	*mData*	- DynamicContext_t pointer to application's module data structure
in	*svuList*	- pointer to a shave list which will specify the shaves to be freed from the application
in	*elements-Number*	- number of shaves in the list

Returns

operation status

void swcDisableShaveCallback (u32 shave_nr)

Disables the interrupt for shave end. Useful for cases where the shave needs to be run for a few times in Async mode with interrupts but then the same shave needs to stop triggering interrupts.

Parameters

`in`	*shave_nr*	- u32 shave number to start
`in`	*function*	- to call when shave finished execution

Returns

void

s32 swcDynShaveAppSetWindows (**DynamicContext_t** ∗ moduleStData, u32 cmxCriticalCodeSize)

This function allows hinting how much code/data is desired to be allocated TODO: add functionality to precompute these sizes based on .textCrit size.

Parameters

`in`	*moduleStData*	- DynamicContext_t pointer to application's module data structure
`in`	*cmx*	critical code size - Desired value for the cmx critical code size. If not set, the default will accommodate 32K

Returns

operation status

void swcDynShaveStartAsync (u32 shave_nr, u32 Context, irq_handler function)

Starts dynamic non blocking execution of a shave. A master entry point is executed prior to jumping into shave entry point.

Parameters

`in`	*shave_nr*	u32 shave number to start
`in`	*Context*	u32 memory address of ModuleData structure
`in`	*function*	to call when shave finished execution

Returns

void

void swcDynStartShave (u32 shave_nr, u32 Context)

Starts non blocking execution of a shave using dynamic sub module alocator.

Parameters

`in`	*shave_nr*	u32 shave number to start
`in`	*Context*	u32 memory address of ModuleData structure

Returns

void

u32 swcGetShaveStackSize (u32 shaveNumber)

Reads back the stack size for a specified shave. When calling either swcSetAbsoluteDefaultStack or swcSetWindowedDefaultStack the stack size set to register i20 will be stored and can be read back with the help of this function.

Parameters

in	*shaveNumber*	- shave number whose stack is to be checked

Returns

u32, stackSize - the stored stack size for the specified shave

void swcGetShaveWindowRegs (u32 shaveNumber, u32 $*$ windows)

Get Shave window register values.

Parameters

in	*shaveNumber*	- shave number for which window register values are retrieved
out	*windows*	- pointer to window registers

Returns

void

s32 swcGetUnallocatedShavesNumber (void)

This function return the number of unallocated shave in the system.

Returns

unallocated shaves number

u32 swcGetUnusedShaveFreeStack (u32 shaveNumber, u32 canaryValue)

If stack painter was used, this function searches for the size of unused stack given pattern checks N-OTE!: this function does nothing relevant if user did not call swcStateConsideredShaveStackSize and swcStackPainter before running a shave application.

Parameters

in	*shaveNumber*	- shave number whose stack is to be checked
in	*canaryValue*	- canary value used for stack painting this particular shave

Returns

void

int swcIsoCleanShaveApplication (**DynamicContext_t** ∗ moduleStData, **swcShaveUnit_t** ∗ svuList, uint32_t shavesNumber, **MISA_PARADIGM_TYPE** paradigmType)

This function frees the heap and group data memory for all configured instances of one application. It can be called after usage of swcRunShaveAlgo(). Can be used from both Leons.

Parameters

in	*moduleStData*	- DynamicContext_t pointer to application's module data structure
in	*svuList*	- pointer to a shave list to be used for all application instances
in	*shaveNumber*	- the shave number for which to set the new location
in	*paradigmType*	- the type of paradigm which is used for running the applications

Returns

operation status

int swcIsoSetupShaveApplication (**DynamicContext_t** ∗ moduleStData, **swcShaveUnit_t** ∗ svuList, uint32_t shavesNumber, **MISA_PARADIGM_TYPE** paradigmType)

This function allocates heap and group data memory for all configured instances of one application and loads the dynamic library. It must be called prior to using swcRunShaveAlgo(). Can be used from both Leons. Please ensure the svuList memory is alive until the call of swcCleanupDynShaveApps. Note: be careful about stack declared svuList.

Parameters

in	*moduleStData*	- DynamicContext_t pointer to application's module data structure
in	*svuList*	- pointer to a shave list to be used for all application instances
in	*shaveNumber*	- the shave number for which to set the new location
in	*paradigmType*	- the type of paradigm which is used for running the applications

Returns

operation status

void swcLoadDynLibrary (u8 ∗ sAddr, u32 targetS, **context_t** contextType)

Dynamically load library file - These are elf object files stripped of any symbols.

Parameters

in	*sAddr*	- starting address where to load the library file
in	*targetS*	- the target Shave
in	*contextType*	- type of the loaded library

Returns

void

void swcLoadDynLibraryCacheAware (u8 ∗ sAddr, u32 targetS, **context_t** contextType, u32 ∗ vpProgrammedMemAddress, u32 ∗ flushLength)

Dynamically load library file and return start memory address and length that need to be flushed - These are elf object files stripped of any symbols.

Parameters

in	*sAddr*	- starting address where to load the library file
in	*targetS*	- the target Shave
in	*contextType*	- type of the loaded library
out	*vp-Programmed-MemAddress*	- first memory address written
out	*flushLength*	- the length of the data written

Returns

void

void swcLoadMbin (u8 ∗ sAddr, u32 targetS)

Load a mbin file to a specific target address on shave.

Parameters

in	*sAddr*	- source address
in	*targetS*	- target shave number

Returns

void

void swcLoadshvdlib (u8 ∗ sAddr, u32 targetS)

Dynamically load shvdlib file - These are elf object files stripped of any symbols.

Parameters

in	*sAddr*	- starting address where to load the shvdlib file
in	*targetS*	- the target Shave

Returns

void

s32 swcRequestUnallocatedShaves (**swcShaveUnit_t** ∗ svuList, u32 shavesNumber)

This functions gives a list of unallocated shaves in the system.

Parameters

in	*svulist*	- pointer to a shave list in which will be assigned the unallocated shaves found in the system

in	*shavesNumber*	- number of unallocated shaves to find ins the system

Returns

operation status

void swcResetShave (u32 shaveNumber)

Reset shave.

Parameters

in	*shaveNumber*	- shave number to be reset

Returns

void

void swcResetShaveLite (u32 shaveNumber)

Reset shave without resetting the fifo.

Parameters

in	*shaveNumber*	- shave number to be reset

Returns

void

void swcRunShave (u32 shave_nr, u32 entry_point)

Start shave shave_nr from entry_point.

Parameters

in	*shave_nr*	- shave number to be started
in	*entry_point*	- entry point

Returns

void

s32 swcRunShaveAlgo (**DynamicContext_t** ∗ moduleStData, int ∗const shaveNumber)

Sets up and launches one dynamic application instance. Uses the shaves preliminary assigned by user via function swcSetupDynShaveApps(). Allocates all necessary memory, loads the dynamic library, then starts the shave.

Parameters

in	*moduleStData*	- DynamicContext_t pointer to ModuleData structure
out	∗*shaveNumber*	- assigned shave number if operation is successful

Returns

operation status

s32 swcRunShaveAlgoCC (**DynamicContext_t** ∗ moduleStData, int ∗const shaveNumber, const char ∗ fmt, ...)

Sets up and launches one dynamic application instance. Uses the shaves preliminary assigned by user via function swcSetupDynShaveApps(). Allocates all necessary memory, loads the dynamic library, then starts the shave.

Parameters

in	*moduleStData*	- DynamicContext_t pointer to ModuleData structure
out	∗*shaveNumber*	- assigned shave number if operation is successful
in	∗*fmt*	- string containing i, s, or v according to irf, srf or vrf ex. "iisv"
in	...	- variable number of params according to fmt

Returns

operation status

s32 swcRunShaveAlgoOnAssignedShave (**DynamicContext_t** ∗ moduleStData, u32 shaveNumber)

Sets up and launches one dynamic application instance on a specifically requested SHAVE Uses the shaves preliminary assigned by user via function swcSetupDynShaveApps(). Allocates all necessary memory, loads the dynamic library, then starts the shave. Checks if the requested shave has bee configured in advance and if it is not running.

Parameters

in	*moduleStData*	- DynamicContext_t pointer to ModuleData structure
in	*shaveNumber*	- specific shave requested by the user to run the algorithm on

Returns

operation status

s32 swcRunShaveAlgoOnAssignedShaveCC (**DynamicContext_t** ∗ moduleStData, u32 shaveNumber, const char ∗ fmt, ...)

Sets up and launches one dynamic application instance on a specifically requested SHAVE Uses the shaves preliminary assigned by user via function swcSetupDynShaveApps(). Allocates all necessary memory, loads the dynamic library, then starts the shave. Checks if the requested shave has bee configured in advance and if it is not running.

Parameters

in	*moduleStData*	- DynamicContext_t pointer to ModuleData structure
in	*shaveNumber*	- specific shave requested by the user to run the algorithm on
in	*∗fmt*	- string containing i, s, or v according to irf, srf or vrf ex. "iisv"
in	...	- variable number of params according to fmt

Returns

operation status

void swcSetAbsoluteDefaultStack (u32 shave_num)

Set absolute default stack for a specific shave.

Parameters

in	*shave_num*	- shave number whose stack is to be set

Returns

void

void swcSetGrpDynDataLocation (**DynamicContext_t** ∗ mData, unsigned char ∗ newAddress, **swcShaveUnit_t** shaveNumber)

This function set a new memory location where to load the grup dynamic data. Can be used from both Leons.

Parameters

in	*mData*	- DynamicContext_t pointer to application's module data structure
in	*newAddress*	- pointer to the memory location of the new location
in	*shaveNumber*	- the shave number for which to set the new location

Returns

operation status

void swcSetNewAppDynDataLocation (**DynamicContext_t** ∗ mData, unsigned char ∗ newAddress, **swcShaveUnit_t** shaveNumber)

This function set a new memory location where to load the application dynamic data. Can be used from both Leons.

Parameters

in	*mData*	- DynamicContext_t pointer to application's module data structure
in	*newAddress*	- pointer to the memory location of the new location
in	*shaveNumber*	- the shave number for which to set the new location

Returns

operation status

void swcSetNewHeapLocation (**DynamicContext_t** ∗ mData, unsigned char ∗ newAddress, **swcShaveUnit_t** shaveNumber)

This function set a new heap location for a specific shave. Can be used from both Leons.

Parameters

in	*mData*	- DynamicContext_t pointer to application's module data structure
in	*newAddress*	- pointer to the memory location of the new location
in	*shaveNumber*	- the shave number for which to set the new location

Returns

void

void swcSetRegsCC (u32 shave_num, const char ∗ fmt, va_list a_list)

void swcSetRounding (u32 shave_no, u32 roundingBits)

Function that starts one shave but at the same time also sets rounding bits.

Parameters

in	*shave_no*	- shave number to start
in	*roundingBits*	- rounding bits

Returns

void

void swcSetShaveWindow (u32 shave_num, u32 window_num, u32 window_addr)

Set a specific window register with a target window base address.

Parameters

in	*shave_num*	- shave number for which window register will be set
in	*window_num*	- window number that should be set
in	*window_addr*	- window address to be put in the window register

Returns

void

void swcSetShaveWindows (u32 shaveNumber, u32 windowA, u32 windowB, u32 windowC, u32 windowD)

Set each window register with the corresponding window base address.

Parameters

in	*shaveNumber*	- shave number for which window registers will be set
in	*windowA*	- base address for window A
in	*windowB*	- base address for window B
in	*windowC*	- base address for window C
in	*windowD*	- base address for window D

Returns

void

void swcSetShaveWindowsToDefault (u32 shaveNumber)

Reset windows to default values in case they are rewritten by other shaves param[in] shaveNumber - shave number for which default value will be set.

Returns

void

s32 swcSetupDynShaveApps (**DynamicContext_t** ∗ moduleStData, const **swcShaveUnit_t** ∗ svuList, const uint32_t instances)

This function allocates heap and group data memory for all configured instances of one application. It must be called prior to using swcRunShaveAlgo(). Can be used from both Leons. svuList below is not copied internally, instead just the pointer is assigned to an internal structure. Please ensure the svuList memory is alive until the call of swcCleanupDynShaveApps. Note: be careful about stack declared svuList.

Parameters

in	*moduleStData*	- DynamicContext_t pointer to application's module data structure
in	*svuList*	- pointer to a shave list to be used for all application instances
in	*instances*	- number of application instances; must correspond to number of shaves configured in svuList

Returns

operation status

void swcSetupShaveCC (u32 shave_num, const char ∗ fmt, ...)

Write the value to a IRF/SRF/VRF Registers from a specific Shave.

Parameters

in	*shave_num*	- shave number to read T-Register value from
in	*∗fmt*	- string containing i, s, or v according to irf, srf or vrf ex. "iisv"
in	...	- variable number of params according to fmt

Returns

void

void swcSetWindowedDefaultStack (u32 shave_num)

Sets a default value for stack.

Attention

Only use this if you are using the default ldscript or really know what you're doing!

Parameters

`in`	*shave_num*	- Shave for which to set the default stack value

Returns

void

u32 swcShaveRunning (u32 svu)

Check if a specific Shave is running or it is stopped.

Parameters

`in`	*svu*	- shave number

Returns

- 0 if stopped
- 1 if running

u32 swcShavesRunning (u32 first, u32 last)

Check if a list of shaves is running or not.

Parameters

`in`	*first*	- first shave in the list
`in`	*last*	- last shave in the list

Returns

- 0 if stopped
- 1 if running

u32 swcShavesRunningArr (u32 arr[ ], u32 N)

Check if a list of shaves stored in an array is running or not.

Parameters

in	*arr*	- array in which are stored shave numbers
in	*N*	- number of elements in the array

Returns

- 0 if stopped
- 1 if running

void swcShaveStartAsync (u32 shave_nr, u32 entry_point, irq_handler function)

Starts non blocking execution of a shave.

Parameters

in	*shave_nr*	u32 shave number to start
in	*entry_point*	u32 memory address to load in the shave instruction pointer before starting
in	*function*	to call when shave finished execution

Returns

void

u32 swcSolveShaveRelAddr (u32 vAddr, u32 shaveNumber)

Translate windowed address into real physical address.

Non-windowed address are passed through.

Parameters

in	*vAddr*	- Input virtual(windowed) Address
in	*shaveNumber*	- Shave to which the virtual address relates

Returns

void swcStackPainter (u32 shaveNumber, u32 canaryValue)

Paint stack with a specific canary value. NOTE: one must have called the swcStateConsideredShave-StackSize on the shaveNumber used here in advance of calling this function.

Parameters

in	*shaveNumber*	- Shave number for shave to paint stack
in	*canaryValue*	- canary value for fill of stack area

Returns

void

int swcStartEntryPointDC (**DynamicContext_t** $*$ moduleStData, uint32_t shaveNumber, const char $*$ functionName)

This function launch a shave application with a specific function as entry point. Can be used from both Leons.

Parameters

in	*moduleStData*	- DynamicContext_t pointer to application's module data structure
in	*functionName*	- pointer to a string containing the name of the entry point to be started on shave side.
in	*shaveNumber*	- the shave number for which to set the new location

Returns

operation status

int swcStartEntryPointDCCC (**DynamicContext_t** ∗ moduleStData, uint32_t shaveNumber, const char ∗ functionName, const char ∗ fmt, ...)

This function launch a shave application with a specific function as entry point. Can be used from both Leons.

Parameters

in	*moduleStData*	- DynamicContext_t pointer to application's module data structure
in	*functionName*	- pointer to a string containing the name of the entry point to be started on shave side.
in	*shaveNumber*	- the shave number for which to set the new location
in	∗*fmt*	- string containing i, s, or v according to irf, srf or vrf ex. "iisv"
in	...	- variable number of params according to fmt

Returns

operation status

int swcStartFC (**DynamicContext_t** ∗ moduleStData, uint32_t shaveNumber)

This function launch a shave application by calling the main function. Can be used from both Leons.

Parameters

in	*moduleStData*	- DynamicContext_t pointer to application's module data structure
in	*shaveNumber*	- the shave number for which to set the new location

Returns

operation status

void swcStartShave (u32 shave_nr, u32 entry_point)

Starts non blocking execution of a shave.

Parameters

in	*shave_nr*	u32 shave number to start
in	*entry_point*	u32 memory address to load in the shave instruction pointer before starting

Returns

void

void swcStartShaveAsync (u32 shave_nr, u32 entry_point, irq_handler function)

void swcStartShaveAsyncCC (u32 shave_num, u32 pc, irq_handler function, const char ∗ fmt, ...)

Write the value to a IRF/SRF/VRF Registers from a specific Shave.

Parameters

in	*shave_num*	- shave number to read T-Register value from
in	*pc*	- function called from the pc
in	*function*	- function to call when shave finished execution
in	*∗fmt*	- string containing i, s, or v according to irf, srf or vrf ex. "iisv"
in	...	- variable number of params according to fmt

Returns

void

void swcStartShaveCC (u32 shave_num, u32 pc, const char ∗ fmt, ...)

Write the value to a IRF/SRF/VRF Registers from a specific Shave.

Parameters

in	*shave_num*	- shave number to read T-Register value from
in	*pc*	- function called from the pc
in	*∗fmt*	- string containing i, s, or v according to irf, srf or vrf ex. "iisv"
in	...	- variable number of params according to fmt

Returns

void

void swcStateConsideredShaveStackSize (u32 shaveNumber, u32 size)

Allows the user to assert a stack size against which checks may be implemented. This does not represent a guarantee that the system will allocate this stack it only allows users to specify how much space they themselves have considered and made available through other means for the application. Calling this function allows the system to perform checks which would detect if this size was overrun at any stage.

Parameters

in	*shaveNumber*	- shave number whose stack is to be set
in	*size*	- Size desired to limit one's application to

Returns

void

int swcWaitShave (u32 shave_nr)

Wait for a specific shave to finish execution.

Parameters

in	*shve_nr*	- shave number we wait for

Returns

void

int swcWaitShaves (u32 no_of_shaves, **swcShaveUnit_t** ∗ shave_list)

Function that waits for the shaves used to finish.

Parameters

in	*no_of_shaves*	- number of shaves that are used
in	∗*shave_list*	- list of shaves used(an array which contains all the shaves used within the application)

Returns

void

6.4 Slice Utils

Slice Utilities API.

Functions

- void swcSliceReleaseMutex (unsigned int mutexNo)

  Function that releases a certain hardware mutex.
- int swcSliceRequestMutex (unsigned int mutexNo, int requestOption)

  Function that requests a certain hardware mutex.
- void swcSetMutexInterrupt (irq_handler mutexHandler, int intMask)

  Function that requests a certain hardware mutex.
- int swcSliceIsMutexFree (unsigned int mutexNo)

  Checks if a mutex is free.

6.4.1 Detailed Description

Slice Utilities API. Used for manipulating slice functionalities

6.4.2 Function Documentation

void swcSetMutexInterrupt (irq_handler mutexHandler, int intMask)

Function that requests a certain hardware mutex.

Parameters

`in`	*mutexHandler*	handler function
`in`	*intMask*	mask to mutex's used

Returns

void

int swcSliceIsMutexFree (unsigned int mutexNo)

Checks if a mutex is free.

Parameters

mutexNo	- mutex number: [0,31]

Returns

1 if the mutex is free and 0 if it is in use

void swcSliceReleaseMutex (unsigned int mutexNo)

Function that releases a certain hardware mutex.

Parameters

in	*mutexNo*	mutex to release

Returns

void

int swcSliceRequestMutex (unsigned int mutexNo, int requestOption)

Function that requests a certain hardware mutex.

Parameters

mutexNo	- mutex number: [0,31]
autoRetry	- If the mutex requested is available, it will be taken , otherwise: • autoRetry=2: :the application will be blocked until the mutex will be taken • autoRetry=1: only the request will be locked , the user may come later to check if the mutex has been taken • autoRetry=0: it will exit the function

Returns

1 if the mutex has been taken, 0 otherwise

6.5 Test Utilities API

Test Utils functions API.

Functions

- tyProcessorType swcGetProcessorType (void)

    This function recognizes the processor type the simulations are running on.
- void swcShaveProfInit (performanceStruct ∗perfStruct)

    Function that initializes the benchmark structure's elements.
- void swcShaveProfStartGathering (u32 shaveNumber, performanceStruct ∗perfStruct)

    Function that starts the counters for structure's members.
- int swcShaveProfGatheringDone (performanceStruct ∗perfStruct)

    Function that verifies if all the structure's parameters are updated with the values from the counters.
- void swcShaveProfStopGathering (u32 shaveNumber, performanceStruct ∗perfStruct)

    Function that reads the values from the counters.
- void swcShaveProfPrint (u32 shaveNumber, performanceStruct ∗perfStruct)

    Function that prints the values that were read from the counters.
- void swcShaveProfStartGatheringFields (u32 shaveNumber, performanceCounterDef perfDefines)

    Function that starts one counter at the time, finding information about one possible field only.
- void swcShaveProfStopFieldsGathering (u32 shaveNumber, performanceCounterDef perfDefines)

    Function that prints and reads values from counters.
- void swcShaveProfStopFieldsGatehering (u32 shaveNumber, performanceCounterDef perfDefines) __Deprecated__("Use swcShaveProfStopFieldsGathering instead")

    Function that prints and reads values from counters.
- void swcShaveProfileCyclesStart (u32 shaveNumber)

    Function that start gathering information about cycles, stalls and instructions.
- void swcShaveProfileCyclesStop (u32 shaveNumber)

    Function that prints and reads values from counters.

6.5.1 Detailed Description

Test Utils functions API. Series of utility functions to facilitate automated test

6.5.2 Function Documentation

tyProcessorType swcGetProcessorType (void)

This function recognizes the processor type the simulations are running on.

Returns

Processor type

int swcShaveProfGatheringDone (**performanceStruct** ∗ perfStruct)

Function that verifies if all the structure's parameters are updated with the values from the counters.

Parameters

in	*perfStruct*	- pointer to the structure that should be updated with the values read from counters

Returns

returns -1 if not all structure‘s filed are updated and 1 if they are

void swcShaveProfileCyclesStart (u32 shaveNumber)

Function that start gathering information about cycles, stalls and instructions.

Parameters

in	*shaveNumber*	- shave number to start

Returns

void

void swcShaveProfileCyclesStop (u32 shaveNumber)

Function that prints and reads values from counters.

Parameters

in	*shaveNumber*	- shave number to start

Returns

void

void swcShaveProfInit (**performanceStruct** * perfStruct)

Function that initializes the benchmark structure‘s elements.

Initializes with either 0, or -1(-1 is used to avoid cases when execution cycles or stalls are 0)

Parameters

in	*perfStruct*	- pointer to the structure that should be initialized

Returns

void

```
swcShaveProfInit(0, &perfStr);

while( swcShaveProfGatheringDone(&perfStr) == -1)
{

    swcShaveProfStartGathering(0, &perfStr);
    swcStartShave(0,(u32)&SVE0_main);
```

```
        swcWaitShave(0);
        swcShaveProfStopGathering(0, &perfStr);

    }
    swcShaveProfPrint(0, &perfStr);
```

void swcShaveProfPrint (u32 shaveNumber, **performanceStruct** ∗ perfStruct)

Function that prints the values that were read from the counters.

Parameters

`in`	*shaveNumber*	- shave number to start
`in`	*perfStruct*	- pointer to the structure whose params are all updated

Returns

void

void swcShaveProfStartGathering (u32 shaveNumber, **performanceStruct** ∗ perfStruct)

Function that starts the counters for structure‘s members.

Parameters

`in`	*shaveNumber*	- shave number to start
`in`	*perfStruct*	- pointer to the structure that should be initialized

Returns

void

void swcShaveProfStartGatheringFields (u32 shaveNumber, **performanceCounterDef** perfDefines)

Function that starts one counter at the time, finding information about one possible field only.

Parameters

`in`	*shaveNumber*	- shave number to start
`in`	*perfDefines*	- one of the fields from the enum perfCounterDef

Returns

void

void swcShaveProfStopFieldsGatehering (u32 shaveNumber, **performanceCounterDef** perfDefines)

Function that prints and reads values from counters.

Parameters

`in`	*shaveNumber*	- shave number to start
`in`	*perfDefines*	- one of the fields from the enum perfCounterDef(for stalls, instructions, branches, timer and clk cycles)

Returns

void

Deprecated This function is deprecated. Use swcShaveProfStopFieldsGathering instead.

void swcShaveProfStopFieldsGathering (u32 shaveNumber, **performanceCounterDef** perfDefines)

Function that prints and reads values from counters.

Parameters

`in`	*shaveNumber*	- shave number to start
`in`	*perfDefines*	- one of the fields from the enum perfCounterDef(for stalls, instructions, branches, timer and clk cycles)

Returns

void

void swcShaveProfStopGathering (u32 shaveNumber, **performanceStruct** ∗ perfStruct)

Function that reads the values from the counters.

Parameters

`in`	*shaveNumber*	- shave number to start
`in`	*perfStruct*	- pointer to the structure that should be updated with the counter values

Returns

void

6.6 Test Utils Defines

Definitions and types needed by software test library.

Data Structures

- struct performanceStruct

Enumerations

- enum tyProcessorType {
  MVI_UNKNOWN, MVI_IC, MVI_VCS, MVI_FSIM,
  MVI_FPGA }
- enum performanceCounterDef {
  PERF_STALL_COUNT, PERF_INSTRUCT_COUNT, PERF_CLK_CYCLE_COUNT, PERF_-BRANCH_COUNT,
  PERF_TIMER_COUNT }

6.6.1 Detailed Description

Definitions and types needed by software test library. This file contains all the definitions of constants, typedefs, structures, enums and exported variables for the Test Utilities

6.6.2 Enumeration Type Documentation

enum **performanceCounterDef**

Enumerator

PERF_STALL_COUNT	counts the stalls
PERF_INSTRUCT_COUNT	counts the instruction cycles
PERF_CLK_CYCLE_COUNT	counts the clock cycles
PERF_BRANCH_COUNT	counts the branches taken
PERF_TIMER_COUNT	counts the total execution of the program

enum **tyProcessorType**

Enumerator

MVI_UNKNOWN	Platform type unknwon.
MVI_IC	ASIC.
MVI_VCS	VCS Simulation.
MVI_FSIM	FagrakSim Simulation.
MVI_FPGA	FPGA Simulation.

6.7 Tracer Log Events

Header for Event ID list.

Enumerations

- enum Event_t {
LOG_EVENT_LOS_RUN = 1, LOG_EVENT_LRT_RUN, LOG_EVENT_WAIT_FOR_LRT, LOG_EVENT_SHAVE_0_RESET = 10,
LOG_EVENT_SHAVE_1_RESET, LOG_EVENT_SHAVE_2_RESET, LOG_EVENT_SHAVE_3_RESET, LOG_EVENT_SHAVE_4_RESET,
LOG_EVENT_SHAVE_5_RESET, LOG_EVENT_SHAVE_6_RESET, LOG_EVENT_SHAVE_7_RESET, LOG_EVENT_SHAVE_8_RESET,
LOG_EVENT_SHAVE_9_RESET, LOG_EVENT_SHAVE_10_RESET, LOG_EVENT_SHAVE_11_RESET, LOG_EVENT_SHAVE_0_RUN,
LOG_EVENT_SHAVE_1_RUN, LOG_EVENT_SHAVE_2_RUN, LOG_EVENT_SHAVE_3_RUN, LOG_EVENT_SHAVE_4_RUN,
LOG_EVENT_SHAVE_5_RUN, LOG_EVENT_SHAVE_6_RUN, LOG_EVENT_SHAVE_7_RUN, LOG_EVENT_SHAVE_8_RUN,
LOG_EVENT_SHAVE_9_RUN, LOG_EVENT_SHAVE_10_RUN, LOG_EVENT_SHAVE_11_RUN, LOG_EVENT_WAIT_FOR_SHAVE_0,
LOG_EVENT_WAIT_FOR_SHAVE_1, LOG_EVENT_WAIT_FOR_SHAVE_2, LOG_EVENT_WAIT_FOR_SHAVE_3, LOG_EVENT_WAIT_FOR_SHAVE_4,
LOG_EVENT_WAIT_FOR_SHAVE_5, LOG_EVENT_WAIT_FOR_SHAVE_6, LOG_EVENT_WAIT_FOR_SHAVE_7, LOG_EVENT_WAIT_FOR_SHAVE_8,
LOG_EVENT_WAIT_FOR_SHAVE_9, LOG_EVENT_WAIT_FOR_SHAVE_10, LOG_EVENT_WAIT_FOR_SHAVE_11, LOG_EVENT_CSS_DIGITAL_POWER,
LOG_EVENT_CSS_ANALOG_POWER, LOG_EVENT_RETENTION, LOG_EVENT_SHAVE_0_POWER, LOG_EVENT_SHAVE_1_POWER,
LOG_EVENT_SHAVE_2_POWER, LOG_EVENT_SHAVE_3_POWER, LOG_EVENT_SHAVE_4_POWER, LOG_EVENT_SHAVE_5_POWER,
LOG_EVENT_SHAVE_6_POWER, LOG_EVENT_SHAVE_7_POWER, LOG_EVENT_SHAVE_8_POWER, LOG_EVENT_SHAVE_9_POWER,
LOG_EVENT_SHAVE_10_POWER, LOG_EVENT_SHAVE_11_POWER, LOG_EVENT_PMB_POWER, LOG_EVENT_MSS_DIGITAL_POWER,
LOG_EVENT_MSS_ANALOG_POWER, LOG_EVENT_DSS_DIGITAL_POWER, LOG_EVENT_DSS_ANALOG_POWER, LOG_EVENT_POWER_M2x5x_BASE = 70,
LOG_EVENT_MSS_CPU_POWER = 86, LOG_EVENT_MSS_AMC_POWER, LOG_EVENT_MSS_SIPP_POWER, LOG_EVENT_DSS_POWER,
LOG_EVENT_USB_POWER, LOG_EVENT_198_RAIL_BASE = 100, LOG_EVENT_198_RAIL_VDDCV_I_MA = LOG_EVENT_198_RAIL_BASE, LOG_EVENT_198_RAIL_VDDCR_I_MA,
LOG_EVENT_198_RAIL_VDDIO_I_MA, LOG_EVENT_198_RAIL_MIPI_VDD_I_MA, LOG_EVENT_198_RAIL_PLL_AVDD_I_MA, LOG_EVENT_198_RAIL_DRAM_MVDDQ_I_MA,
LOG_EVENT_198_RAIL_DRAM_MVDDA_I_MA, LOG_EVENT_198_RAIL_DRAM_VDD1_I_MA, LOG_EVENT_198_RAIL_DRAM_VDD2_I_MA, LOG_EVENT_198_RAIL_DRAM_VDDQ_I_MA,
LOG_EVENT_198_RAIL_USB_VDD330_I_MA, LOG_EVENT_198_RAIL_USB_VP_VDD_I_MA, LOG_EVENT_198_RAIL_VDDCV_V_MV, LOG_EVENT_198_RAIL_MIPI_VDD_-

V_MV,
LOG_EVENT_198_RAIL_VDDIO_B_I_MUL_I_MA_MA2150, LOG_EVENT_198_TOTAL-_CURRENT, LOG_EVENT_198_TOTAL_POWER, LOG_EVENT_198_DDR_CURRENT, LOG_EVENT_198_DDR_POWER, LOG_EVENT_SYS_CLK_CHANGE = 200, LOG_EVEN-T_LAST_EVENT = 9999 }

6.7.1 Detailed Description

Header for Event ID list. This file contains a list of event IDs for the Tracer

6.7.2 Enumeration Type Documentation

enum **Event_t**

Enumerator

LOG_EVENT_LOS_RUN

LOG_EVENT_LRT_RUN

LOG_EVENT_WAIT_FOR_LRT

LOG_EVENT_SHAVE_0_RESET

LOG_EVENT_SHAVE_1_RESET

LOG_EVENT_SHAVE_2_RESET

LOG_EVENT_SHAVE_3_RESET

LOG_EVENT_SHAVE_4_RESET

LOG_EVENT_SHAVE_5_RESET

LOG_EVENT_SHAVE_6_RESET

LOG_EVENT_SHAVE_7_RESET

LOG_EVENT_SHAVE_8_RESET

LOG_EVENT_SHAVE_9_RESET

LOG_EVENT_SHAVE_10_RESET

LOG_EVENT_SHAVE_11_RESET

LOG_EVENT_SHAVE_0_RUN

LOG_EVENT_SHAVE_1_RUN

LOG_EVENT_SHAVE_2_RUN

LOG_EVENT_SHAVE_3_RUN

LOG_EVENT_SHAVE_4_RUN

LOG_EVENT_SHAVE_5_RUN

LOG_EVENT_SHAVE_6_RUN

LOG_EVENT_SHAVE_7_RUN

LOG_EVENT_SHAVE_8_RUN

LOG_EVENT_SHAVE_9_RUN

LOG_EVENT_SHAVE_10_RUN

LOG_EVENT_SHAVE_11_RUN

LOG_EVENT_WAIT_FOR_SHAVE_0

LOG_EVENT_WAIT_FOR_SHAVE_1
LOG_EVENT_WAIT_FOR_SHAVE_2
LOG_EVENT_WAIT_FOR_SHAVE_3
LOG_EVENT_WAIT_FOR_SHAVE_4
LOG_EVENT_WAIT_FOR_SHAVE_5
LOG_EVENT_WAIT_FOR_SHAVE_6
LOG_EVENT_WAIT_FOR_SHAVE_7
LOG_EVENT_WAIT_FOR_SHAVE_8
LOG_EVENT_WAIT_FOR_SHAVE_9
LOG_EVENT_WAIT_FOR_SHAVE_10
LOG_EVENT_WAIT_FOR_SHAVE_11
LOG_EVENT_CSS_DIGITAL_POWER
LOG_EVENT_CSS_ANALOG_POWER
LOG_EVENT_RETENTION
LOG_EVENT_SHAVE_0_POWER
LOG_EVENT_SHAVE_1_POWER
LOG_EVENT_SHAVE_2_POWER
LOG_EVENT_SHAVE_3_POWER
LOG_EVENT_SHAVE_4_POWER
LOG_EVENT_SHAVE_5_POWER
LOG_EVENT_SHAVE_6_POWER
LOG_EVENT_SHAVE_7_POWER
LOG_EVENT_SHAVE_8_POWER
LOG_EVENT_SHAVE_9_POWER
LOG_EVENT_SHAVE_10_POWER
LOG_EVENT_SHAVE_11_POWER
LOG_EVENT_PMB_POWER
LOG_EVENT_MSS_DIGITAL_POWER
LOG_EVENT_MSS_ANALOG_POWER
LOG_EVENT_DSS_DIGITAL_POWER
LOG_EVENT_DSS_ANALOG_POWER
LOG_EVENT_POWER_M2x5x_BASE
LOG_EVENT_MSS_CPU_POWER
LOG_EVENT_MSS_AMC_POWER
LOG_EVENT_MSS_SIPP_POWER
LOG_EVENT_DSS_POWER
LOG_EVENT_USB_POWER
LOG_EVENT_198_RAIL_BASE
LOG_EVENT_198_RAIL_VDDCV_I_MA
LOG_EVENT_198_RAIL_VDDCR_I_MA
LOG_EVENT_198_RAIL_VDDIO_I_MA

LOG_EVENT_198_RAIL_MIPI_VDD_I_MA
LOG_EVENT_198_RAIL_PLL_AVDD_I_MA
LOG_EVENT_198_RAIL_DRAM_MVDDQ_I_MA
LOG_EVENT_198_RAIL_DRAM_MVDDA_I_MA
LOG_EVENT_198_RAIL_DRAM_VDD1_I_MA
LOG_EVENT_198_RAIL_DRAM_VDD2_I_MA
LOG_EVENT_198_RAIL_DRAM_VDDQ_I_MA
LOG_EVENT_198_RAIL_USB_VDD330_I_MA
LOG_EVENT_198_RAIL_USB_VP_VDD_I_MA
LOG_EVENT_198_RAIL_VDDCV_V_MV
LOG_EVENT_198_RAIL_MIPI_VDD_V_MV
LOG_EVENT_198_RAIL_VDDIO_B_I_MUL_I_MA_MA2150
LOG_EVENT_198_TOTAL_CURRENT
LOG_EVENT_198_TOTAL_POWER
LOG_EVENT_198_DDR_CURRENT
LOG_EVENT_198_DDR_POWER
LOG_EVENT_SYS_CLK_CHANGE
LOG_EVENT_LAST_EVENT

6.8 Fp16 Convert

Fp16 manipulation and conversion utility minimal set of fp16 conversions functions for sharing data between Leon and SHAVES or other HW blocks which expect fp16 data.

Macros

- #define MOVIDIUS_FP32
- #define F32_NAN_DEFAULT 0xFFC00000
- #define EXTRACT_F16_SIGN(x) ((x >> 15) & 0x1)
- #define EXTRACT_F16_EXP(x) ((x >> 10) & 0x1F)
- #define EXTRACT_F16_FRAC(x) (x & 0x000003FF)
- #define EXTRACT_F32_SIGN(x) ((x >> 31) & 0x1)
- #define EXTRACT_F32_EXP(x) ((x >> 23) & 0xFF)
- #define EXTRACT_F32_FRAC(x) (x & 0x007FFFFF)
- #define RESET_SNAN_BIT(x) x = x | 0x00400000
- #define PACK_F32(x, y, z) ((x << 31) + (y << 23) + z)
- #define PACK_F16(x, y, z) ((x << 15) + (y << 10) + z)
- #define F16_IS_NAN(x) ((x & 0x7FFF)> 0x7C00)
- #define F16_IS_SNAN(x) (((x & 0x7E00) == 0x7C00)&&((x & 0x1FF)> 0))
- #define F32_IS_NAN(x) ((x & 0x7FFFFFFF)> 0x7F800000)
- #define F32_IS_SNAN(x) (((x & 0x7FC00000) == 0x7F800000)&&((x & 0x3FFFFF)> 0))

Functions

- unsigned int f32Tof16 (float x)

  Convert fp32 to fp16 param[in] x - float(fp32) input to be converted.
- float f16Tof32 (unsigned int x)

  Convert fp16 to fp32 param[in] x - fp16 input to be converted.

Rounding modes

- #define F32_RND_NEAREST_EVEN 0
- #define F32_RND_MINUS_INF 1
- #define F32_RND_PLUS_INF 2
- #define F32_RND_TO_ZERO 3

Detect tinyness mode

- #define F32_DETECT_TINY_AFTER_RND 0
- #define F32_DETECT_TINY_BEFORE_RND 1

Exceptions

- #define F32_EX_INEXACT 0x00000001
- #define F32_EX_DIV_BY_ZERO 0x00000002
- #define F32_EX_INVALID 0x00000004
- #define F32_EX_UNDERFLOW 0x00000008
- #define F32_EX_OVERFLOW 0x00000010

6.8.1 Detailed Description

Fp16 manipulation and conversion utility minimal set of fp16 conversions functions for sharing data between Leon and SHAVES or other HW blocks which expect fp16 data.

6.8.2 Macro Definition Documentation

#define EXTRACT_F16_EXP(x) ((x >> 10) & 0x1F)

#define EXTRACT_F16_FRAC(x) (x & 0x000003FF)

#define EXTRACT_F16_SIGN(x) ((x >> 15) & 0x1)

#define EXTRACT_F32_EXP(x) ((x >> 23) & 0xFF)

#define EXTRACT_F32_FRAC(x) (x & 0x007FFFFF)

#define EXTRACT_F32_SIGN(x) ((x >> 31) & 0x1)

#define F16_IS_NAN(x) ((x & 0x7FFF)> 0x7C00)

#define F16_IS_SNAN(x) (((x & 0x7E00) == 0x7C00)&&((x & 0x1FF)> 0))

#define F32_DETECT_TINY_AFTER_RND 0

#define F32_DETECT_TINY_BEFORE_RND 1

#define F32_EX_DIV_BY_ZERO 0x00000002

#define F32_EX_INEXACT 0x00000001

#define F32_EX_INVALID 0x00000004

#define F32_EX_OVERFLOW 0x00000010

#define F32_EX_UNDERFLOW 0x00000008

#define F32_IS_NAN(x) ((x & 0x7FFFFFFF)> 0x7F800000)

#define F32_IS_SNAN(x) (((x & 0x7FC00000) == 0x7F800000)&&((x & 0x3FFFFF)> 0))

#define F32_NAN_DEFAULT 0xFFC00000

#define F32_RND_MINUS_INF 1

#define F32_RND_NEAREST_EVEN 0

#define F32_RND_PLUS_INF 2

#define F32_RND_TO_ZERO 3

#define MOVIDIUS_FP32

#define PACK_F16(x, y, z) ((x << 15) + (y << 10) + z)

#define PACK_F32(x, y, z) ((x << 31) + (y << 23) + z)

#define RESET_SNAN_BIT(x) x = x | 0x00400000

6.8.3 Function Documentation

float f16Tof32 (unsigned int x)

Convert fp16 to fp32 param[in] x - fp16 input to be converted.

Returns

float(fp32) value

unsigned int f32Tof16 (float x)

Convert fp32 to fp16 param[in] x - float(fp32) input to be converted.

Returns

fp16 value

6.9 CMXDMA API

CMXDMA driver common API.

Functions

- dmaRequesterId dmaInitRequester (int priority)

  Initialize a requester ID which will be used to properly initialize and distinguish single tasks or groups of tasks.
- dmaTransactionList ∗ dmaCreateTransactionFullOptions (dmaRequesterId ReqId, dmaTransactionList ∗NewTransaction, u8 ∗Src, u8 ∗Dst, u32 ByteLength, u32 SrcLineWidth, u32 DstLineWidth, s32 SrcStride, s32 DstStride)

  Initialize a new CMXDMA task structure which can be used to realize a DMA data transfer using source and destination strides.
- dmaTransactionList ∗ dmaCreateTransaction (dmaRequesterId ReqId, dmaTransactionList ∗NewTransaction, u8 ∗Src, u8 ∗Dst, u32 ByteLength)

  Initialize a new CMXDMA task structure which can be used to realize a simple DMA data transfer.
- dmaTransactionList ∗ dmaCreateTransactionSrcStride (dmaRequesterId ReqId, dmaTransactionList ∗NewTransaction, u8 ∗Src, u8 ∗Dst, u32 ByteLength, u32 LineWidth, s32 SrcStride)

  Initialize a new CMXDMA task structure which can be used to realize a DMA data transfer using source stride only.
- dmaTransactionList ∗ dmaCreateTransactionDstStride (dmaRequesterId ReqId, dmaTransactionList ∗NewTransaction, u8 ∗Src, u8 ∗Dst, u32 ByteLength, u32 LineWidth, s32 DstStride)

  Initialize a new CMXDMA task structure which can be used to realize a DMA data transfer using destination stride only.
- dmaTransactionList ∗ dmaCreate3DTransaction (dmaRequesterId ReqId, dmaTransactionList ∗NewTransaction, u8 ∗Src, u8 ∗Dst, u32 ByteLength, u32 SrcLineWidth, u32 DstLineWidth, s32 SrcStride, s32 DstStride, u32 NumPlanes, s32 SrcPlaneStride, s32 DstPlaneStride)

  Creates a new 3D transaction.
- void dmaLinkTasks (dmaTransactionList ∗listHead, u32 nmbTasks,...)

  Link multiple tasks in a single linked list. Please note that this function allows linking just for single tasks.
- int dmaStartListTask (dmaTransactionList ∗ListPtr)

  Set-up CMXDMA to execute the given list of tasks.
- void dmaWaitTask (dmaTransactionList ∗ListPtr)

  Wait in a blocking way for a given task to finish.
- int dmaIsTaskFinished (dmaTransactionList ∗ListPtr)

  Check whether a task finished it's execution or is still running/pending.

6.9.1 Detailed Description

CMXDMA driver common API. This driver lets you perform fast data transfers using CMXDMA hardware

6.9.2 Function Documentation

dmaTransactionList∗ dmaCreate3DTransaction (dmaRequesterId ReqId, **dmaTransactionList** ∗ NewTransaction, u8 ∗ Src, u8 ∗ Dst, u32 ByteLength, u32 SrcLineWidth, u32 DstLineWidth, s32 SrcStride, s32 DstStride, u32 NumPlanes, s32 SrcPlaneStride, s32 DstPlaneStride)

Creates a new 3D transaction.

The function returns a handle to the new transaction.

Parameters

in	*ReqId*	- A requester ID returned by function dmaInitRequester used to set the task priority and the task ID
in	*New-Transaction*	- Pointer to user-allocated space for a new task structure
in	*Src*	- source address for the transaction.
in	*Dst*	- destination address for the transaction.
in	*ByteLength*	- Size(in bytes) of the transfer
in	*SrcLineWidth*	- line width for source in bytes.
in	*DstLineWidth*	- line width for destination in bytes.
in	*SrcStride*	- stride size for source, defined as the size in bytes from the start of a line to the start of the following line.
in	*DstStride*	- stride size for destination, defined as the size in bytes from the start of a line to the start of the following line.
in	*NumPlanes*	- number of planes of data to be transfered. The value needs to be greater than zero for 3D transaction.
in	*SrcPlaneStride*	- plane stride size for source, defined as the size in bytes from the start of a plane to the start of the following plane.
in	*DstPlaneStride*	- plane stride size for destination, defined as the size in bytes from the start of a plane to the start of the following plane.

Returns

Pointer to initialized CMXDMA structure

dmaTransactionList∗ dmaCreateTransaction (dmaRequesterId ReqId, **dmaTransactionList** ∗ NewTransaction, u8 ∗ Src, u8 ∗ Dst, u32 ByteLength)

Initialize a new CMXDMA task structure which can be used to realize a simple DMA data transfer.

The transaction type is a 2D transaction Please make sure the Src and Dst parameters are received with the proper restrictions if your application has particular ones.

Parameters

in	*ReqId*	- A requester ID returned by function dmaInitRequester used to set the task priority and the task ID

in	*New-Transaction*	- Pointer to user-allocated space for a new task structure
in	*Src*	- Source address of data transfer
in	*Dst*	- Destination address of data transfer
in	*ByteLength*	- Size(in bytes) of the transfer

Returns

Pointer to initialized CMXDMA structure

dmaTransactionList∗ dmaCreateTransactionDstStride (dmaRequesterId ReqId, **dmaTransactionList** ∗ NewTransaction, u8 ∗ Src, u8 ∗ Dst, u32 ByteLength, u32 LineWidth, s32 DstStride)

Initialize a new CMXDMA task structure which can be used to realize a DMA data transfer using destination stride only.

Please make sure the Src and Dst parameters are received with the proper restrictions if your application has particular ones.

Parameters

in	*ReqId*	- A requester ID returned by function dmaInitRequester used to set the task priority and the task ID
in	*New-Transaction*	- Pointer to user-allocated space for a new task structure
in	*Src*	- Source address of data transfer
in	*Dst*	- Destination address of data transfer
in	*ByteLength*	- Size(in bytes) of the transfer
in	*LineWidth*	- Destination line width
in	*DstStride*	- Destination stride

Returns

Pointer to initialized CMXDMA structure

dmaTransactionList∗ dmaCreateTransactionFullOptions (dmaRequesterId ReqId, **dmaTransactionList** ∗ NewTransaction, u8 ∗ Src, u8 ∗ Dst, u32 ByteLength, u32 SrcLineWidth, u32 DstLineWidth, s32 SrcStride, s32 DstStride)

Initialize a new CMXDMA task structure which can be used to realize a DMA data transfer using source and destination strides.

Please make sure the Src and Dst parameters are received with the proper restrictions if your application has particular ones.

Parameters

in	*ReqId*	- A requester ID returned by function dmaInitRequester used to set the task priority and the task ID
in	*New-Transaction*	- Pointer to user-allocated space for a new task structure
in	*Src*	- Source address of data transfer
in	*Dst*	- Destination address of data transfer
in	*ByteLength*	- Size(in bytes) of the transfer
in	*SrcLineWidth*	- Source line width
in	*DstLineWidth*	- Destination line width
in	*SrcStride*	- Source stride
in	*DstStride*	- Destination stride

Returns

Pointer to initialized CMXDMA structure

dmaTransactionList∗ dmaCreateTransactionSrcStride (dmaRequesterId ReqId, **dmaTransactionList** ∗ NewTransaction, u8 ∗ Src, u8 ∗ Dst, u32 ByteLength, u32 LineWidth, s32 SrcStride)

Initialize a new CMXDMA task structure which can be used to realize a DMA data transfer using source stride only.

Please make sure the Src and Dst parameters are received with the proper restrictions if your application has particular ones.

Parameters

in	*ReqId*	- A requester ID returned by function dmaInitRequester used to set the task priority and the task ID
in	*New-Transaction*	- Pointer to user-allocated space for a new task structure
in	*Src*	- Source address of data transfer
in	*Dst*	- Destination address of data transfer
in	*ByteLength*	- Size(in bytes) of the transfer
in	*LineWidth*	- Source line width
in	*SrcStride*	- Source stride

Returns

Pointer to initialized CMXDMA structure

dmaRequesterId dmaInitRequester (int priority)

Initialize a requester ID which will be used to properly initialize and distinguish single tasks or groups of tasks.

Parameters

in	*priority*	- The priority that will be assigned to all the tasks created using the returned ID

Returns

a new requester ID

int dmaIsTaskFinished (**dmaTransactionList** ∗ ListPtr)

Check whether a task finished it's execution or is still running/pending.

Parameters

in	*ListPtr*	- Pointer to the task to be checked

Returns

- 0 - Task is still executed/pending
- 1 - Task finished it's execution

void dmaLinkTasks (**dmaTransactionList** ∗ listHead, u32 nmbTasks, ...)

Link multiple tasks in a single linked list. Please note that this function allows linking just for single tasks.

Note

One can not link together in this way two or more linked lists of tasks in order to form a single list.

Parameters

in	*listHead*	- Pointer to the structure which will represent the start of linked task list.
in	*nmbTasks*	- Number of tasks to be linked to list head
in	...	- Pointers to the tasks to be linked. The structures passed here will be linked to listHead from left to right, in order of their placement on function call.

Returns

void

int dmaStartListTask (**dmaTransactionList** ∗ ListPtr)

Set-up CMXDMA to execute the given list of tasks.

Note

Please note if there is heavy use of CMXDMA, the task list won't start immediately, it will be put in a waiting queue until CMXDMA will become available to execute the current task.

Parameters

`in`	*ListPtr*	- Pointer to the task or list of tasks to be executed

Returns

- 0 - CMXDMA waiting queue is full, no new tasks can be added now
- 1 - Tasks have been submitted directly to CMXDMA and are executing now
- 2 - Tasks have been added to a waiting queue and are pending execution

void dmaWaitTask (**dmaTransactionList** ∗ ListPtr)

Wait in a blocking way for a given task to finish.

Parameters

`in`	*ListPtr*	- Pointer to the task to be waited

Returns

void

6.10 CMXDMA Defines

Common definitions and types needed by CMXDMA driver.

Data Structures

- struct configBits

  Bit field for fine-grained configuration of CMXDMA transaction.
- struct dmaTransactionList_t

  2D transaction type

Macros

- #define ALIGNED8 __attribute__ ((aligned (8)))
- #define SVU_SLICE_OFFSET 0x10000
- #define SWC_CMX_DMA_DEFAULT_NUM_PLANE (0)
- #define SWC_CMX_DMA_DEFAULT_PLANE_STRIDE (0)
- #define MIN_NUM_PLANES (1)
- #define MAX_NUM_PLANES (256)

Typedefs

- typedef dmaTransactionList_t dmaTransactionList
- typedef void(∗ dmaIrqHandler)(dmaTransactionList ∗ListPtr, void ∗userContext)

6.10.1 Detailed Description

Common definitions and types needed by CMXDMA driver. This file contains all the definitions of constants, typedefs, structures, enums and exported variables for CMXDMA driver for Shave and PC

6.10.2 Macro Definition Documentation

#define ALIGNED8 **__attribute__** ((aligned (8)))

#define MAX_NUM_PLANES (256)

#define MIN_NUM_PLANES (1)

#define SVU_SLICE_OFFSET 0x10000

#define SWC_CMX_DMA_DEFAULT_NUM_PLANE (0)

#define SWC_CMX_DMA_DEFAULT_PLANE_STRIDE (0)

6.10.3 Typedef Documentation

typedef void(∗ dmaIrqHandler)(**dmaTransactionList** ∗ListPtr, void ∗userContext) typedef **dmaTransactionList_t dmaTransactionList**

6.11 CRC Utility

Simple Table based CRC Calculation library.

Functions

- u32 swcCalcCrc32 (u8 ∗pBuffer, u32 byteLength, pointer_type pt)

  Calculate simple CRC32 over a byte buffer of byteLength.

6.11.1 Detailed Description

Simple Table based CRC Calculation library. Offers cyclic redundancy check functionality in order to perform data correctness checkup

6.11.2 Function Documentation

u32 swcCalcCrc32 (u8 ∗ pBuffer, u32 byteLength, **pointer_type** pt)

Calculate simple CRC32 over a byte buffer of byteLength.

Parameters

in	*pBuffer*	- byte pointer to buffer
in	*byteLength*	- length of buffer in bytes
in	*pt*	- initial endianness of the buffer

Returns

32 bit crc of the buffer

6.12 Leon Math Utilities

API for some required Leon Math functions.

Functions

- float swcMathSinf (float angle)

  Utility trigonometric function to calculate the sine of an angle.
- float swcMathCosf (float angle)
- u32 swcIPow (u32 base, u32 exp)

  Utility Integer function to raise base^exp.
- double swcLongLongToDouble (unsigned long long longVal)

  Utility function to cast a 64 bit int to a double.

6.12.1 Detailed Description

API for some required Leon Math functions. Used to implement math functions

6.12.2 Function Documentation

u32 swcIPow (u32 base, u32 exp)

Utility Integer function to raise base^exp.

Parameters

`in`	*base*	- 32 bit value on which to operate
`in`	*exp*	- 32 bit exponent

Returns

base^exp

double swcLongLongToDouble (unsigned long long longVal)

Utility function to cast a 64 bit int to a double.

This function is used to avoid the need to link in glibc for memory frugality reasons

Parameters

`in`	*longVal*	- 64 bit integer to be cast

Returns

floating point double equivalent of longVal

float swcMathCosf (float angle)

trigonometric function to calculate the cosine of an angle

Parameters

in	*angle*	- angle on which to calculate cosine

Returns

cosin(angle)

float swcMathSinf (float angle)

Utility trigonometric function to calculate the sine of an angle.

Parameters

in	*angle*	- angle on which to calculate sine

Returns

sin(angle)

6.13 Leon Utilities API

API manipulating Leon functionalities.

Macros

- #define NATIVE_POINTER_TYPE le_pointer
- #define swcLeonSwapU32(value)

  Swaps endianness of a 32-bit integer (usefull when sharing data between Leon and Shave)
- #define swcLeonSwapU16(value)

  Swaps endianness of a 16-bit integer (usefull when sharing data between Leon and Shave)
- #define swcLeonReadNoCacheU8(addr)

  Reads data bypassing leon LRAM cache.
- #define swcLeonReadNoCacheI8(addr)

  Reads data bypassing leon LRAM cache.
- #define swcLeonReadNoCacheU16(addr)

  Reads data bypassing leon LRAM cache.
- #define swcLeonReadNoCacheI16(addr)

  Reads data bypassing leon LRAM cache.
- #define swcLeonReadNoCacheU32(addr)

  Reads data bypassing leon LRAM cache.
- #define swcLeonReadNoCacheI32(addr) ((int)swcLeonReadNoCacheU32(addr))

  Reads data bypassing leon LRAM cache.
- #define swcLeonReadNoCacheU64(addr)

  Reads data bypassing leon L1 cache.
- #define swcLeonReadNoCacheI64(addr) ((s64)swcLeonReadNoCacheU64(addr))

  Reads data bypassing leon L1 cache.
- #define swcLeonWriteNoCache8(addr, data)

  Writes data bypassing leon LRAM cache.
- #define swcLeonWriteNoCache16(addr, data)

  Writes data bypassing leon LRAM cache.
- #define swcLeonWriteNoCache32(addr, data)

  Writes data bypassing leon LRAM cache.
- #define swcLeonWriteNoCache64(addr, data)

  Writes data bypassing leon L1 cache.
- #define swcLeonFlushCaches() asm volatile("flush" ::: "memory")

  Flush Leon Instruction and Data Caches.
- #define swcLeonDataCacheFlush()

  Flush Leon Data Cache.
- #define swcLeonFlushDcache() swcLeonDataCacheFlush()
- #define swcLeonDataCacheFlushNoWait() swcLeonDataCacheFlush()
- #define swcLeonInstructionCacheFlush()

  Flush Leon Instruction Cache.
- #define swcLeonFlushIcache() swcLeonInstructionCacheFlush()
- #define swcLeonIsCacheFlushPending()

Check if Leon cache flush is pending.

- #define swcLeonEnableCaches(flush)

  Enable Leon Instruction and Data Caches.
- #define swcLeonEnableIcache(flush)

  Enable Leon Instruction Cache.
- #define swcLeonEnableDcache(flush)

  Enable Leon Data Cache.
- #define swcLeonDisableCaches() asm volatile("sta %%g0, [%%g0] 2" ::: "memory")

  Disable Leon Instruction and Data Caches.
- #define swcLeonDisableDcache()

  Disable Leon Data Cache.
- #define swcLeonDisableIcache()

  Disable Leon Instruction Cache.
- #define swcLeonDisableTraps()

  Disable traps.
- #define swcLeonEnableTraps()

  Enable traps.
- #define swcLeonL1DForceCacheLineMiss(addr) swcRead32Asi01(addr)

  Force a Leon L1 data cache miss.

Enumerations

- enum pointer_type { be_pointer, le_pointer }

  Pointer type.

Functions

- void swcLeonDataCacheFlushBlockWhilePending (void)

  Flushes Leon data cache, and wait while the flush is pending. (DO NOT USE)
- void swcLeonHalt (void)

  Stops Leon.
- int swcLeonSetPIL (u32 pil)

  Sets the Processor Interrupt Level atomically.
- void swcLeonFlushWindows (void)

  Flushes all the interrupt windows before the caller's to the stack.
- void swcLeonMemCpy (void ∗dst, pointer_type dst_pt, const void ∗src, pointer_type src_pt, u32 count)

  Generic memory copying function to copy le/be buffers to le/be buffers.
- void swcLeonMemMove (void ∗dst, pointer_type dst_pt, const void ∗src, pointer_type src_pt, u32 count)

  Same as swcLeonMemCpy, except buffers may overlap.
- void swcLeonSwapBuffer (void ∗buf, pointer_type pt, u32 count)

  Swap the endianness of a buffer in place.

6.13.1 Detailed Description

API manipulating Leon functionalities. Allows manipulating leon caches and other features

6.13.2 Macro Definition Documentation

#define NATIVE_POINTER_TYPE **le_pointer**

#define swcLeonDataCacheFlush( )

Value:

```
asm volatile(                                       \
        "sta %%g0, [%%g0] %[dcache_flush_asi]"          \
        :                                               \
        : [dcache_flush_asi] "I" (__DCACHE_FLUSH_ASI)   \
        : "memory"                                      \
    )
```

Flush Leon Data Cache.

#define swcLeonDataCacheFlushNoWait( ) **swcLeonDataCacheFlush**()

#define swcLeonDisableCaches( ) asm volatile("sta %%g0, [%%g0] 2" ::: "memory")

Disable Leon Instruction and Data Caches.

#define swcLeonDisableDcache( )

Value:

```
({ \
        unsigned local_var_tmp; \
        asm volatile( \
            "lda [%%g0] %[__CCR_ASI_], %[tmp]"        "\n\t" \
            "bclr %[ccr_clearthese], %[tmp]"          "\n\t" \
            "sta %[tmp], [%%g0] %[__CCR_ASI_]"              \
            : [tmp] "=&r" (local_var_tmp)                 \
            : [ccr_clearthese] "r" (CCR_DCS_ENABLED),       \
              [__CCR_ASI_] "I" (__CCR_ASI)                  \
            : "memory" \
        ); \
    })
```

Disable Leon Data Cache.

#define swcLeonDisableIcache( )

Value:

```
({ \
        unsigned local_var_tmp; \
        asm volatile( \
            "lda [%%g0] %[__CCR_ASI_], %[tmp]"        "\n\t" \
            "bclr %[ccr_cleathese], %[tmp]"           "\n\t" \
            "sta %[tmp], [%%g0] %[__CCR_ASI_]"                    \
            : [tmp] "=&r" (local_var_tmp)                       \
            : [ccr_clearthese] "r" (CCR_IB | CCR_ICS_ENABLED), \
              [__CCR_ASI_] "I" (__CCR_ASI)                      \
            : "memory"                                          \
        ); \
    })
```

Disable Leon Instruction Cache.

#define swcLeonDisableTraps( )

Value:

```
({ \
    int temp; \
    int old_et; \
    asm volatile ( \
        "rd      %%psr, %[ret]"            "\n\t" \
        "andn    %[ret], 0x00000020, %[temp]"  "\n\t" \
        "wr      %[temp], %%psr"           "\n\t" \
        " and    %[ret], 0x00000020, %[ret]"   "\n\t" \
        " srl    %[ret], 5, %[ret]"        "\n\t" \
        " nop" \
        : [ret] "=r" (old_et), \
          [temp] "=r" (temp) \
        :: "memory", "cc"); \
    old_et; \
})
```

Disable traps.

Attention

Enter/leave a critical section in a clean way - do NOT call any function between these!

Returns

- 1 if traps were enabled
- 0 if traps were not enabled

#define swcLeonEnableCaches(flush)

Value:

```
asm volatile(                                                     \
        "sta %[ccr_value], [%%g0] %[__CCR_ASI_]"                      \
        :                                                             \
        : [ccr_value] "r" (CCR_IB | CCR_ICS_ENABLED |
      CCR_DCS_ENABLED |    \
                  ((flush) ? (CCR_FI | CCR_FD) : 0)),                 \
          [__CCR_ASI_] "I" (__CCR_ASI)                                \
        : "memory"                                                    \
    )
```

Enable Leon Instruction and Data Caches.

Parameters

in	*flush*	flag: 0 = don't flush cache ; 1 = flush cache

#define swcLeonEnableDcache(flush)

Value:

```
({ \
        unsigned local_var_tmp; \
        asm volatile(                                               \
            "lda [%%g0] %[__CCR_ASI_], %[tmp]"        "\n\t" \
            "bset %[ccr_flags], %[tmp]"               "\n\t" \
            "sta %[tmp], [%%g0] %[__CCR_ASI_]"              \
            : [tmp] "=&r" (local_var_tmp)                  \
            : [ccr_flags] "r" ((CCR_DCS_ENABLED) | ((flush) ?
      CCR_FD : 0)), \
              [__CCR_ASI_] "I" (__CCR_ASI)                  \
            : "memory" \
        ); \
    })
```

Enable Leon Data Cache.

Parameters

in	*flush*	flag: 0 = don't flush cache ; 1 = flush cache

#define swcLeonEnableIcache(flush)

Value:

```
({ \
        unsigned local_var_tmp; \
        asm volatile(                                             \
            "lda [%%g0] %[__CCR_ASI_], %[tmp]"      "\n\t" \
            "bset %[ccr_flags], %[tmp]"             "\n\t" \
            "sta %[tmp], [%%g0] %[__CCR_ASI_]"            \
            : [tmp] "=&r" (local_var_tmp)                         \
            : [ccr_flags] "r" (CCR_IB | CCR_ICS_ENABLED | ((flush)?
      CCR_FI:0)), \
              [__CCR_ASI_] "I" (__CCR_ASI)                \
            : "memory"                                    \
        ); \
    })
```

Enable Leon Instruction Cache.

Parameters

in	*flush*	flag: 0 = don't flush cache ; 1 = flush cache

#define swcLeonEnableTraps( )

Value:

```
({ \
    int temp; \
    int old_et; \
    asm volatile ( \
        "rd      %%psr, %[ret]"             "\n\t" \
        "or      %[ret], 0x00000020, %[temp]"   "\n\t" \
        "wr      %[temp], %%psr"            "\n\t" \
        " and    %[ret], 0x00000020, %[ret]"    "\n\t" \
        " srl    %[ret], 5, %[ret]"         "\n\t" \
        " nop" \
        : [ret] "=r" (old_et), \
          [temp] "=r" (temp) \
        :: "memory", "cc"); \
    old_et; \
})
```

Enable traps.

Attention

Enter/leave a critical section in a clean way - do NOT call any function between these!!!

Returns

- 1 if traps were enabled
- 0 if traps were not enabled

#define swcLeonFlushCaches( ) asm volatile("flush" ::: "memory")

Flush Leon Instruction and Data Caches.

#define swcLeonFlushDcache( ) **swcLeonDataCacheFlush**()

#define swcLeonFlushIcache( ) **swcLeonInstructionCacheFlush**()

#define swcLeonInstructionCacheFlush( )

Value:

```
({                                                    \
        unsigned local_var_tmp;                                         \
        asm volatile(                                         \
            "lda [%%g0] %[__CCR_ASI_], %[tmp]"   "\n\t" \
            "bset %[ccr_flags], %[tmp]"          "\n\t" \
            "sta %[tmp], [%%g0] %[__CCR_ASI_]"          \
            : [tmp] "=&r" (local_var_tmp)                        \
            : [ccr_flags] "r" (CCR_FI),                 \
              [__CCR_ASI_] "I" (__CCR_ASI)              \
            : "memory"                                  \
        ); \
    })
```

Flush Leon Instruction Cache.

#define swcLeonIsCacheFlushPending( )

Value:

```
({                                          \
        unsigned local_var_tmp, local_var_result;              \
        asm (                                   \
            "lda [%%g0] %[__CCR_ASI_], %[tmp]"  \
            : [tmp] "=r" (local_var_tmp)                     \
            : [__CCR_ASI_] "I" (__CCR_ASI)        \
            : "memory"                          \
        );                                      \
        local_var_result = (local_var_tmp >> 14) & 3;          \
        local_var_result;                                 \
    })
```

Check if Leon cache flush is pending.

#define swcLeonL1DForceCacheLineMiss(addr) swcRead32Asi01(addr)

Force a Leon L1 data cache miss.

Reads the value pointed by addr directly from memory. Fills/Updates the whole L1C line

Parameters

in	*addr*	u32 address to read

Note

swcLeonReadNoCache∗ will become deprecated

#define swcLeonReadNoCacheI16(addr)

Value:

```
({ \
        signed short local_var_result; \
        asm volatile( \
            "ldsha [%[addr_]] 1, %[result]" \
            : [result] "=r"(local_var_result) \
            : [addr_] "r" (addr) \
            : "memory" ); \
        local_var_result; \
    })
```

Reads data bypassing leon LRAM cache.

Parameters

in	*addr*	u32 address to read

Returns

- i16 variable value read bypassing cache

#define swcLeonReadNoCacheI32(addr) ((int)**swcLeonReadNoCacheU32**(addr))

Reads data bypassing leon LRAM cache.

Parameters

in	*addr*	u32 address to read

Returns

- s32 variable value read bypassing cache

#define swcLeonReadNoCacheI64(addr) ((s64)**swcLeonReadNoCacheU64**(addr))

Reads data bypassing leon L1 cache.

Parameters

in	*addr*	s64 address to read

Returns

- s64 variable value read bypassing cache

#define swcLeonReadNoCacheI8(addr)

Value:

```
({ \
        signed char local_var_result; \
        asm volatile( \
            "ldsba [%[addr_]] 1, %[result]" \
            : [result] "=r" (local_var_result) \
            : [addr_] "r" (addr) \
            : "memory" ); \
        local_var_result; \
    })
```

Reads data bypassing leon LRAM cache.

Parameters

in	*addr*	u32 address to read

Returns

- i8 variable value read bypassing cache

#define swcLeonReadNoCacheU16(addr)

Value:

```
({ \
        unsigned short local_var_result; \
        asm volatile( \
            "lduha [%[addr_]] 1, %[result]" \
            : [result] "=r" (local_var_result) \
            : [addr_] "r" (addr) \
            : "memory" ); \
        local_var_result; \
    })
```

Reads data bypassing leon LRAM cache.

Parameters

in	*addr*	u32 address to read

Returns

- u16 variable value read bypassing cache

#define swcLeonReadNoCacheU32(addr)

Value:

```
({ \
        unsigned int local_var_result; \
        asm volatile( \
            "lda [%[addr_]] 1, %[result]" \
            : [result] "=r" (local_var_result) \
            : [addr_] "r" (addr) \
            : "memory" ); \
        local_var_result; \
    })
```

Reads data bypassing leon LRAM cache.

Parameters

in	*addr*	u32 address to read

Returns

- u32 variable value read bypassing cache

#define swcLeonReadNoCacheU64(addr)

Value:

```
({ \
        u64 local_var_result; \
        asm volatile( \
            "ldda  [%[addr_]] 1, %[result]" \
            : [result] "=r" (local_var_result) \
            : [addr_] "r" ((addr)) \
            : "memory" ); \
        local_var_result; \
    })
```

Reads data bypassing leon L1 cache.

Parameters

in	*addr*	address of u64 to read

Returns

- u64 variable value read bypassing cache

#define swcLeonReadNoCacheU8(addr)

Value:

```
({ \
        unsigned char local_var_result; \
        asm volatile( \
            "lduba [%[addr_]] 1, %[result]" \
            : [result] "=r" (local_var_result) \
            : [addr_] "r" (addr) \
            : "memory" \
        ); \
        local_var_result; \
    })
```

Reads data bypassing leon LRAM cache.

Parameters

in	*addr*	u32 address to read

Returns

- u8 variable value read bypassing cache

#define swcLeonSwapU16(value)

Value:

```
((((u16)((value) & 0x00FF)) << 8) | \
                (((u16)((value) & 0xFF00)) >> 8))
```

Swaps endianness of a 16-bit integer (usefull when sharing data between Leon and Shave)

Parameters

in	*value*	u16 integer to be swapped

Returns

swapped integer

#define swcLeonSwapU32(value)

Value:

```
((((u32)((value) & 0x000000FF)) << 24) | \
                ( ((u32)((value) & 0x0000FF00)) <<  8) | \
                ( ((u32)((value) & 0x00FF0000)) >>  8) | \
                ( ((u32)((value) & 0xFF000000)) >> 24))
```

Swaps endianness of a 32-bit integer (usefull when sharing data between Leon and Shave)

Parameters

in	*value*	u32 integer to be swapped

Returns

swapped integer

#define swcLeonWriteNoCache16(addr, **data**)

Value:

```
asm volatile( \
        "stha %[data_], [%[addr_]] 1" \
        : \
        : [addr_] "r" (addr), \
          [data_] "r" (data) \
        : "memory" \
    )
```

Writes data bypassing leon LRAM cache.

Parameters

in	*addr*	- u32 address to write
in	*data*	- i16/u16 variable to write

#define swcLeonWriteNoCache32(addr, **data**)

Value:

```
asm volatile( \
        "sta %[data_], [%[addr_]] 1" \
        : \
        : [addr_] "r" (addr), \
          [data_] "r" (data) \
        : "memory" \
    )
```

Writes data bypassing leon LRAM cache.

Parameters

in	*addr*	- u32 address to write
in	*data*	- s32/u32 variable to write

#define swcLeonWriteNoCache64(addr, **data**)

Value:

```
asm volatile( \
        "stda  %[data_], [%[addr_]] 1" \
        : \
        : [data_] "r"((u64)(data)), \
          [addr_] "r" ((addr)) \
        : "memory" \
    )
```

Writes data bypassing leon L1 cache.

Parameters

in	*addr*	- u64 address to write
in	*data*	- s64/u64 variable to write

#define swcLeonWriteNoCache8(addr, **data**)

Value:

```
asm volatile( \
        "stba %[data_], [%[addr_]] 1" \
        : \
        : [addr_] "r"(addr), \
          [data_] "r"(data) \
        : "memory" \
    )
```

Writes data bypassing leon LRAM cache.

Parameters

in	*addr*	- u32 address to write
in	*data*	- i8/u8 variable to write

6.13.3 Enumeration Type Documentation

enum **pointer_type**

Pointer type.

Note

the pointer type is only relevant if it is addressing $<$ 32bit values

Enumerator

be_pointer normal leon pointer

le_pointer little-endian/shave pointer

6.13.4 Function Documentation

void swcLeonDataCacheFlushBlockWhilePending (void)

Flushes Leon data cache, and wait while the flush is pending. (DO NOT USE)

Note

It is not recommended to use this function Leon DCache flush takes 128 cycles as it processes each line of the cache at 1 cycle per line There is no advantage to wait until the flush is not pending anymore. use the swcLeonDataCacheFlush() macro instead.

void swcLeonFlushWindows (void)

Flushes all the interrupt windows before the caller's to the stack.

Note

> You'd ideally call this before your main app loop, if any - allows you to avoid window_overflow's for the next 6-deep calls

void swcLeonHalt (void)

Stops Leon.

void swcLeonMemCpy (void $*$ dst, **pointer_type** dst_pt, const void $*$ src, **pointer_type** src_pt, u32 count)

Generic memory copying function to copy le/be buffers to le/be buffers.

- The buffers may be unaligned, and they may have an unaligned size.
- The buffers may be anywhere in memory, data is accessed using word-access only
- The buffers may not overlap! If you need overlapping buffers, then see swcLeonMemMove(). Exceptions to the no-overlap rule:
  1. Same endianness buffers may overlap if you know for sure that the destination will always be before the source (meaning (u32)src >= (u32)dst), assert(src >= dst);
  2. Different endianness buffers may overlap if (u32)src >= (u32)dst + 3 if (src_pt != dst_pt) assert(src >= dst + 3);

  Parameters

out	*dst*	The destination buffer.
in	*dst_pt*	The endianness of the destination buffer
in	*src*	The source buffer
in	*src_pt*	The endianness of the source buffer
in	*count*	Number of bytes to copy. It is not required for this to be divisible by 4.

void swcLeonMemMove (void $*$ dst, **pointer_type** dst_pt, const void $*$ src, **pointer_type** src_pt, u32 count)

Same as swcLeonMemCpy, except buffers may overlap.

The distance between overlapping buffer pointers of opposite endianness must be >= 3

if (src_pt != dst_pt) assert(abs(src - dst) >=3);

Parameters

out	*dst*	The destination buffer.
in	*dst_pt*	The endianness of the destination buffer
in	*src*	The source buffer
in	*src_pt*	The endianness of the source buffer
in	*count*	Number of bytes to copy. It is not required for this to be divisible by 4.

int swcLeonSetPIL (u32 pil)

Sets the Processor Interrupt Level atomically.

Parameters

in	*pil*	- processor interrupt level

Returns

- previous processor interrupt level

void swcLeonSwapBuffer (void ∗ buf, **pointer_type** pt, u32 count)

Swap the endianness of a buffer in place.

The buffer pointer and count may be unaligned, but you have to make sure that sufficient bytes are aligned before and after the buffer, to fit the flipped buffer.

Parameters

in,out	*buf*	- Buffer to work on
in	*pt*	- initial endianness of the buffer
in	*count*	- number of bytes

6.14 Leon Utilities Defines

API manipulating Leon functionalities.

Macros

- #define MASK_PSR_impl 0xf0000000
- #define POS_PSR_impl 28
- #define MASK_PSR_ver 0x0f000000
- #define POS_PSR_ver 24
- #define MASK_PSR_icc 0x00f00000
- #define POS_PSR_icc 20
- #define PSR_N 0x00800000
- #define PSR_Z 0x00400000
- #define PSR_V 0x00200000
- #define PSR_C 0x00100000
- #define PSR_EC 0x00002000
- #define PSR_EF 0x00001000
- #define MASK_PSR_PIL 0x00000f00
- #define POS_PSR_PIL 8
- #define PSR_PIL0 0x00000000
- #define PSR_PIL1 0x00000100
- #define PSR_PIL2 0x00000200
- #define PSR_PIL3 0x00000300
- #define PSR_PIL4 0x00000400
- #define PSR_PIL5 0x00000500
- #define PSR_PIL6 0x00000600
- #define PSR_PIL7 0x00000700
- #define PSR_PIL8 0x00000800
- #define PSR_PIL9 0x00000900
- #define PSR_PIL10 0x00000a00
- #define PSR_PIL11 0x00000b00
- #define PSR_PIL12 0x00000c00
- #define PSR_PIL13 0x00000d00
- #define PSR_PIL14 0x00000e00
- #define PSR_PIL15 0x00000f00
- #define PSR_S 0x00000080
- #define PSR_PS 0x00000040
- #define PSR_ET 0x00000020
- #define MASK_PSR_CWP 0x0000001f
- #define POS_PSR_CWP 0
- #define PSR_CWP0 0x00000000
- #define PSR_CWP1 0x00000001
- #define PSR_CWP2 0x00000002
- #define PSR_CWP3 0x00000003
- #define PSR_CWP4 0x00000004
- #define PSR_CWP5 0x00000005

- #define PSR_CWP6 0x00000006
- #define PSR_CWP7 0x00000007
- #define MASK_WIM_BITS 0x000000ff
- #define WIM_INVD0 0x00000001
- #define WIM_INVD1 0x00000002
- #define WIM_INVD2 0x00000004
- #define WIM_INVD3 0x00000008
- #define WIM_INVD4 0x00000010
- #define WIM_INVD5 0x00000020
- #define WIM_INVD6 0x00000040
- #define WIM_INVD7 0x00000080
- #define MASK_TBR_tba 0xfffff000
- #define POS_TBR_tba 12
- #define MASK_TBR_tt 0x00000ff0
- #define POS_TBR_tt 4
- #define TBR_tt_reset 0x000
- #define TBR_tt_instr_access_exception 0x010
- #define TBR_tt_illegal_instr 0x020
- #define TBR_tt_privileged_instr 0x030
- #define TBR_tt_fp_disabled 0x040
- #define TBR_tt_window_overflow 0x050
- #define TBR_tt_window_underflow 0x060
- #define TBR_tt_mem_address_not_aligned 0x070
- #define TBR_tt_fp_exception 0x080
- #define TBR_tt_data_access_exception 0x090
- #define TBR_tt_tag_overflow 0x0A0
- #define TBR_tt_watchpoint 0x0B0
- #define TBR_tt_IRQ1 0x110
- #define TBR_tt_IRQ2 0x120
- #define TBR_tt_IRQ3 0x130
- #define TBR_tt_IRQ4 0x140
- #define TBR_tt_IRQ5 0x150
- #define TBR_tt_IRQ6 0x160
- #define TBR_tt_IRQ7 0x170
- #define TBR_tt_IRQ8 0x180
- #define TBR_tt_IRQ9 0x190
- #define TBR_tt_IRQ10 0x1A0
- #define TBR_tt_IRQ11 0x1B0
- #define TBR_tt_IRQ12 0x1C0
- #define TBR_tt_IRQ13 0x1D0
- #define TBR_tt_IRQ14 0x1E0
- #define TBR_tt_IRQ15 0x1F0
- #define TBR_tt_r_register_access_error 0x200
- #define TBR_tt_instr_access_error 0x210
- #define TBR_tt_cp_disabled 0x240
- #define TBR_tt_unimplemented_FLUSH 0x250
- #define TBR_tt_cp_exception 0x280

- #define TBR_tt_data_access_error 0x290
- #define TBR_tt_division_by_0 0x2A0
- #define TBR_tt_data_store_error 0x2B0
- #define TBR_tt_data_access_MMU_miss 0x2C0
- #define TBR_tt_instr_access_MMU_miss 0x3C0
- #define TBR_tt_user_trap_0 0x800
- #define TBR_tt_user_trap_127 0xFF0
- #define MASK_FSR_RD 0xC0000000
- #define POS_FSR_RD 30
- #define FSR_RD_NEAREST 0x00000000
- #define FSR_RD_ZERO 0x40000000
- #define FSR_RD_INF 0x80000000
- #define FSR_RD_NINF 0xC0000000
- #define MASK_FSR_TEM 0x0f800000
- #define POS_FSR_TEM 25
- #define FSR_NVM 0x08000000
- #define FSR_OFM 0x04000000
- #define FSR_UFM 0x02000000
- #define FSR_DZM 0x01000000
- #define FSR_NXM 0x00800000
- #define FSR_NS 0x00400000
- #define MASK_FSR_ver 0x000E0000
- #define POS_FSR_ver 17
- #define MASK_FSR_tt 0x0001C000
- #define POS_FSR_rrm 14
- #define FSR_tt_NONE 0x00000000
- #define FSR_tt_IEEE 0x00004000
- #define FSR_tt_UNF 0x00008000
- #define FSR_tt_SEQUENCE 0x00010000
- #define FSR_QNE 0x00002000
- #define MASK_FSR_fcc 0x00000C00
- #define POS_FSR_fcc 10
- #define FSR_EQ 0x00000000
- #define FSR_LT 0x00000400
- #define FSR_GT 0x00000800
- #define FSR_UNORDERED 0x00000C00
- #define MASK_FSR_AEXC 0x000003E0
- #define POS_FSR_AEXC 5
- #define FSR_NVA 0x00000200
- #define FSR_OFA 0x00000100
- #define FSR_UFA 0x00000080
- #define FSR_DFA 0x00000040
- #define FSR_NXA 0x00000020
- #define MASK_FSR_CEXC 0x0000001F
- #define POS_FSR_CEXC 0
- #define FSR_NVC 0x00000010
- #define FSR_OFC 0x00000008

- #define FSR_UFC 0x00000004
- #define FSR_DFC 0x00000002
- #define FSR_NXC 0x00000001
- #define MASK_HBRK_ADDR 0xC0000000
- #define LEON_PROCESSOR_INDEX_MASK (1 << 28)
- #define ASR17_DWT (0x00004000)
- #define ASR17_SVT (0x00002000)
- #define __CCR_ASI 0x02
- #define __CCR_OFS 0x00000000
- #define CACHE_CONTROL_REG_OFS (0x00000000)
- #define ICACHE_CONFIG_REG_OFS (0x00000008)
- #define DCACHE_CONFIG_REG_OFS (0x0000000C)
- #define CCR_FI (1<<21)
- #define CCR_FD (1<<22)
- #define POS_CCR_IP 15
- #define CCR_IP (1<<POS_CCR_IP)
- #define POS_CCR_DP 14
- #define CCR_DP (1<<POS_CCR_DP)
- #define CCR_DS (1<<23)
- #define CCR_DF (1<<5)
- #define CCR_IF (1<<4)
- #define MASK_CCR_DCS (3<<2)
- #define CCR_DCS_ENABLED (3<<2)
- #define CCR_DCS_FROZEN (1<<2)
- #define CCR_DCS_DISABLED (0<<2)
- #define MASK_CCR_ICS (3<<0)
- #define CCR_ICS_ENABLED (3<<0)
- #define CCR_ICS_FROZEN (1<<0)
- #define CCR_ICS_DISABLED (0<<0)
- #define CCR_IB (1<<16)
- #define __NOCACHE_ASI 0x01
- #define __ICACHE_TAGS_ASI 0x0C
- #define __ICACHE_DATA_ASI 0x0D
- #define __DCACHE_TAGS_ASI 0x0E
- #define __DCACHE_DATA_ASI 0x0F
- #define __ICACHE_FLUSH_ASI_DO_NOT_USE 0x10
- #define __DCACHE_FLUSH_ASI 0x11
- #define _ASM __asm__ __volatile__
- #define NOP _ASM("nop;":::"memory")

6.14.1 Detailed Description

API manipulating Leon functionalities. Register defines for swcLeonUtils

6.14.2 Macro Definition Documentation

#define __CCR_ASI 0x02

#define __CCR_OFS 0x00000000

#define __DCACHE_DATA_ASI 0x0F

#define __DCACHE_FLUSH_ASI 0x11

#define __DCACHE_TAGS_ASI 0x0E

#define __ICACHE_DATA_ASI 0x0D

#define __ICACHE_FLUSH_ASI_DO_NOT_USE 0x10

#define __ICACHE_TAGS_ASI 0x0C

#define __NOCACHE_ASI 0x01

#define _ASM __asm__ __volatile__

#define ASR17_DWT (0x00004000)

#define ASR17_SVT (0x00002000)

#define CACHE_CONTROL_REG_OFS (0x00000000)

#define CCR_DCS_DISABLED (0<<2)

#define CCR_DCS_ENABLED (3<<2)

#define CCR_DCS_FROZEN (1<<2)

#define CCR_DF (1<<5)

#define CCR_DP (1<<**POS_CCR_DP**)

#define CCR_DS (1<<23)

#define CCR_FD (1<<22)

#define CCR_FI (1<<21)

#define CCR_IB (1<<16)

#define CCR_ICS_DISABLED (0<<0)

#define CCR_ICS_ENABLED (3<<0)

#define CCR_ICS_FROZEN (1<<0)

```
#define CCR_IF (1<<4)

#define CCR_IP (1<<POS_CCR_IP)

#define DCACHE_CONFIG_REG_OFS (0x0000000C)

#define FSR_DFA 0x00000040

#define FSR_DFC 0x00000002

#define FSR_DZM 0x01000000

#define FSR_EQ 0x00000000

#define FSR_GT 0x00000800

#define FSR_LT 0x00000400

#define FSR_NS 0x00400000

#define FSR_NVA 0x00000200

#define FSR_NVC 0x00000010

#define FSR_NVM 0x08000000

#define FSR_NXA 0x00000020

#define FSR_NXC 0x00000001

#define FSR_NXM 0x00800000

#define FSR_OFA 0x00000100

#define FSR_OFC 0x00000008

#define FSR_OFM 0x04000000

#define FSR_QNE 0x00002000

#define FSR_RD_INF 0x80000000

#define FSR_RD_NEAREST 0x00000000

#define FSR_RD_NINF 0xC0000000

#define FSR_RD_ZERO 0x40000000

#define FSR_tt_IEEE 0x00004000

#define FSR_tt_NONE 0x00000000
```

```
#define FSR_tt_SEQUENCE 0x00010000

#define FSR_tt_UNF 0x00008000

#define FSR_UFA 0x00000080

#define FSR_UFC 0x00000004

#define FSR_UFM 0x02000000

#define FSR_UNORDERED 0x00000C00

#define ICACHE_CONFIG_REG_OFS (0x00000008)

#define LEON_PROCESSOR_INDEX_MASK ( 1 << 28 )

#define MASK_CCR_DCS (3<<2)

#define MASK_CCR_ICS (3<<0)

#define MASK_FSR_AEXC 0x000003E0

#define MASK_FSR_CEXC 0x0000001F

#define MASK_FSR_fcc 0x00000C00

#define MASK_FSR_RD 0xC0000000

#define MASK_FSR_TEM 0x0f800000

#define MASK_FSR_tt 0x0001C000

#define MASK_FSR_ver 0x000E0000

#define MASK_HBRK_ADDR 0xC0000000

#define MASK_PSR_CWP 0x0000001f

#define MASK_PSR_icc 0x00f00000

#define MASK_PSR_impl 0xf0000000

#define MASK_PSR_PIL 0x00000f00

#define MASK_PSR_ver 0x0f000000

#define MASK_TBR_tba 0xfffff000

#define MASK_TBR_tt 0x00000ff0

#define MASK_WIM_BITS 0x000000ff
```

```
#define NOP _ASM("nop;":::"memory")

#define POS_CCR_DP 14

#define POS_CCR_IP 15

#define POS_FSR_AEXC 5

#define POS_FSR_CEXC 0

#define POS_FSR_fcc 10

#define POS_FSR_RD 30

#define POS_FSR_rrm 14

#define POS_FSR_TEM 25

#define POS_FSR_ver 17

#define POS_PSR_CWP 0

#define POS_PSR_icc 20

#define POS_PSR_impl 28

#define POS_PSR_PIL 8

#define POS_PSR_ver 24

#define POS_TBR_tba 12

#define POS_TBR_tt 4

#define PSR_C 0x00100000

#define PSR_CWP0 0x00000000

#define PSR_CWP1 0x00000001

#define PSR_CWP2 0x00000002

#define PSR_CWP3 0x00000003

#define PSR_CWP4 0x00000004

#define PSR_CWP5 0x00000005

#define PSR_CWP6 0x00000006

#define PSR_CWP7 0x00000007
```

```
#define PSR_EC 0x00002000

#define PSR_EF 0x00001000

#define PSR_ET 0x00000020

#define PSR_N 0x00800000

#define PSR_PIL0 0x00000000

#define PSR_PIL1 0x00000100

#define PSR_PIL10 0x00000a00

#define PSR_PIL11 0x00000b00

#define PSR_PIL12 0x00000c00

#define PSR_PIL13 0x00000d00

#define PSR_PIL14 0x00000e00

#define PSR_PIL15 0x00000f00

#define PSR_PIL2 0x00000200

#define PSR_PIL3 0x00000300

#define PSR_PIL4 0x00000400

#define PSR_PIL5 0x00000500

#define PSR_PIL6 0x00000600

#define PSR_PIL7 0x00000700

#define PSR_PIL8 0x00000800

#define PSR_PIL9 0x00000900

#define PSR_PS 0x00000040

#define PSR_S 0x00000080

#define PSR_V 0x00200000

#define PSR_Z 0x00400000

#define TBR_tt_cp_disabled 0x240

#define TBR_tt_cp_exception 0x280
```

```
#define TBR_tt_data_access_error 0x290

#define TBR_tt_data_access_exception 0x090

#define TBR_tt_data_access_MMU_miss 0x2C0

#define TBR_tt_data_store_error 0x2B0

#define TBR_tt_division_by_0 0x2A0

#define TBR_tt_fp_disabled 0x040

#define TBR_tt_fp_exception 0x080

#define TBR_tt_illegal_instr 0x020

#define TBR_tt_instr_access_error 0x210

#define TBR_tt_instr_access_exception 0x010

#define TBR_tt_instr_access_MMU_miss 0x3C0

#define TBR_tt_IRQ1 0x110

#define TBR_tt_IRQ10 0x1A0

#define TBR_tt_IRQ11 0x1B0

#define TBR_tt_IRQ12 0x1C0

#define TBR_tt_IRQ13 0x1D0

#define TBR_tt_IRQ14 0x1E0

#define TBR_tt_IRQ15 0x1F0

#define TBR_tt_IRQ2 0x120

#define TBR_tt_IRQ3 0x130

#define TBR_tt_IRQ4 0x140

#define TBR_tt_IRQ5 0x150

#define TBR_tt_IRQ6 0x160

#define TBR_tt_IRQ7 0x170

#define TBR_tt_IRQ8 0x180

#define TBR_tt_IRQ9 0x190
```

#define TBR_tt_mem_address_not_aligned 0x070

#define TBR_tt_privileged_instr 0x030

#define TBR_tt_r_register_access_error 0x200

#define TBR_tt_reset 0x000

#define TBR_tt_tag_overflow 0x0A0

#define TBR_tt_unimplemented_FLUSH 0x250

#define TBR_tt_user_trap_0 0x800

#define TBR_tt_user_trap_127 0xFF0

#define TBR_tt_watchpoint 0x0B0

#define TBR_tt_window_overflow 0x050

#define TBR_tt_window_underflow 0x060

#define WIM_INVD0 0x00000001

#define WIM_INVD1 0x00000002

#define WIM_INVD2 0x00000004

#define WIM_INVD3 0x00000008

#define WIM_INVD4 0x00000010

#define WIM_INVD5 0x00000020

#define WIM_INVD6 0x00000040

#define WIM_INVD7 0x00000080

6.15 Random Number Generator

Modules

- Random API

  API for Simple Pseudo Random Number Generator Library.
- Random API Defines

  Definitions and types needed by swcRandom.

6.15.1 Detailed Description

6.16 Random API

API for Simple Pseudo Random Number Generator Library.

Functions

- void swcRandInit (u64 initValue)

  Reset the base seed of the PRNG.
- u64 swcRandGetRandValue (void)

  Get next 64 bit random value in sequence defined by the global seed which was set using swcRandInit().
- u64 swcRandGetRandValue_r (u64 ∗seed)

  Get next 64 bit random value in sequence defined by seed.
- int swcRandBufferOp (tyRandOperation operation, void ∗targetAddress, u32 len, u64 seed)

6.16.1 Detailed Description

API for Simple Pseudo Random Number Generator Library. Allows for painting memory with a known pseudo random pattern Or verifying memory against the same known pattern

Note

> This is NOT a cryptographically secure PRNG generator. This is a Linear Congruential Generator (LCG). See: `http://en.wikipedia.org/wiki/Linear_congruential_generator`. The magic values used here are from Donald Knuth's MMIX LCG.

6.16.2 Function Documentation

int swcRandBufferOp (**tyRandOperation** operation, void ∗ targetAddress, u32 len, u64 seed)

Paint or verify a buffer with pseudo random pattern

Function which either paints a buffer with a pseudo random pattern, or verifies that buffer against an expected pseudo random pattern. The Seed for the random pattern is passed as a parameter

Parameters

`in`	*operation*	- (RAND_WRITE, RAND_VERIFY, or 32 bit equivalents)
`in`	*targetAddress*	- buffer to be painted of verified (word or byte depending on operation)
`in`	*len*	- length of buffer in bytes
`in`	*seed*	- Seed to be applied to the rand operation

Returns

> 0 on success, non-zero otherwise

u64 swcRandGetRandValue (void)

Get next 64 bit random value in sequence defined by the global seed which was set using swcRandInit().

Note

This is equivalent to rand() from standard C.

Returns

64 bit pseudo random value between [0, RAND_MAX]

u64 swcRandGetRandValue_r (u64 * seed)

Get next 64 bit random value in sequence defined by seed.

The result of this function does not depend on the global seed that was set by swcRandInit. If it is called many times with the parameter pointing to the same value, then the result will be the same.

Note

This is equivalent to rand_r() from standard C.

Parameters

`in,out`	*seed*	- pointer to the 64 bit seed value, which will be updated.

Returns

64 bit pseudo random value between [0, RAND_MAX]

void swcRandInit (u64 initValue)

Reset the base seed of the PRNG.

Note

This is equivalent to srand() from standard C.

Parameters

`in`	*initValue*	- 64 bit initial value

6.17 Random API Defines

Definitions and types needed by swcRandom.

Macros

- #define RAND_MAX ((u64)(-1))

Enumerations

- enum tyRandOperation { RAND_WRITE, RAND_VERIFY, RAND_WRITE_32, RAND_VERIFY_32 }

6.17.1 Detailed Description

Definitions and types needed by swcRandom. This file contains all the definitions of constants, typedefs, structures, enums and exported variables for the Simple Pseudo Random Number Generator Library

6.17.2 Macro Definition Documentation

#define RAND_MAX ((u64)(-1))

6.17.3 Enumeration Type Documentation

enum **tyRandOperation**

Enumerator

RAND_WRITE

RAND_VERIFY

RAND_WRITE_32

RAND_VERIFY_32

Chapter 7

Data Structure Documentation

7.1 configBits Struct Reference

Bit field for fine-grained configuration of CMXDMA transaction.

`#include <swcCdmaCommonDefines.h>`

Data Fields

- u32 type: 2

  Transaction type(1D/2D)
- u32 priority: 2

  Transaction priority(0 - 3)
- u32 brstLength: 4

  Burst length.
- u32 id: 4

  Transaction ID.
- u32 interruptTrigger: 4

  ID of interrupt to be generated when the task is executed.
- u32 reserved1: 4

  Reserved.
- u32 disableInt: 1

  Disable interrupts.
- u32 reserved2: 6

  Reserved.
- u32 skipNr: 5

  Skip descriptor.

7.1.1 Detailed Description

Bit field for fine-grained configuration of CMXDMA transaction.

7.1.2 Field Documentation

u32 configBits::brstLength

Burst length.

u32 configBits::disableInt

Disable interrupts.

u32 configBits::id

Transaction ID.

u32 configBits::interruptTrigger

ID of interrupt to be generated when the task is executed.

u32 configBits::priority

Transaction priority(0 - 3)

u32 configBits::reserved1

Reserved.

u32 configBits::reserved2

Reserved.

u32 configBits::skipNr

Skip descriptor.

u32 configBits::type

Transaction type(1D/2D)

The documentation for this struct was generated from the following file:

- swcCdmaCommonDefines.h

7.2 dmaTransactionList_t Struct Reference

2D transaction type

```
#include <swcCdmaCommonDefines.h>
```

Data Fields

- void ∗ linkAddress

  pointer to the next element in linked list
- union {
  configBits cfgBits
  u32 fullCfgRegister
  } cfgLink
- void ∗ src

  Pointer to the source of the data transfer.
- void ∗ dst

  Pointer to the destination.
- u32 length

  Transaction length.
- u32 no_planes

  Number of planes.
- u32 src_width

  Bytes of data required from one line of source.
- u32 src_stride

  Length in bytes from start of one line of data, to start of next line of data.
- u32 dst_width

  Bytes of data required from one line of destination.
- u32 dst_stride

  Length in bytes from start of one line of data, to start of next line of data.
- u32 src_plane_stride

  Source plane stride.
- u32 dst_plane_stride

  Destination plane stride.
- u32 agentOff
- u32 userData0

7.2.1 Detailed Description

2D transaction type

7.2.2 Field Documentation

u32 dmaTransactionList_t::agentOff

configBits dmaTransactionList_t::cfgBits

union { ... } dmaTransactionList_t::cfgLink

void∗ dmaTransactionList_t::dst

Pointer to the destination.

u32 dmaTransactionList_t::dst_plane_stride

Destination plane stride.

u32 dmaTransactionList_t::dst_stride

Length in bytes from start of one line of data, to start of next line of data.

u32 dmaTransactionList_t::dst_width

Bytes of data required from one line of destination.

u32 dmaTransactionList_t::fullCfgRegister

u32 dmaTransactionList_t::length

Transaction length.

void∗ dmaTransactionList_t::linkAddress

pointer to the next element in linked list

u32 dmaTransactionList_t::no_planes

Number of planes.

void∗ dmaTransactionList_t::src

Pointer to the source of the data transfer.

u32 dmaTransactionList_t::src_plane_stride

Source plane stride.

u32 dmaTransactionList_t::src_stride

Length in bytes from start of one line of data, to start of next line of data.

u32 dmaTransactionList_t::src_width

Bytes of data required from one line of source.

u32 dmaTransactionList_t::userData0

The documentation for this struct was generated from the following file:

- swcCdmaCommonDefines.h

7.3 DynamicContext_elm Struct Reference

```
#include <theDynContext.h>
```

Data Fields

- _ExecutionContext_t ∗ crtContextInfo
- _TorFn_t ∗ ctors_start
- _TorFn_t ∗ ctors_end
- _TorFn_t ∗ dtors_start
- _TorFn_t ∗ dtors_end
- uint32_t heap_size
- uint32_t stack_size
- unsigned char ∗ entryPoint
- DynamicContextInstancesPtr instancesData
- ParadigmSpecificEntry pse [TOTAL_NUM_SHAVES]
- uint32_t groupEntryPoint
- uint64_t ∗ appdynbssdatastart
- uint64_t ∗ appdynbssdataend
- unsigned char ∗ appdyndata
- unsigned int appdyndatasize
- void ∗ appdyndataAllocAddr [TOTAL_NUM_SHAVES]
- uint64_t ∗ groupappdynbssdatastart
- uint64_t ∗ groupappdynbssdataend
- unsigned char ∗ groupappdyndata
- unsigned int groupappdyndatasize
- DYNCONTEXT_HEAP_ACTION_TYPE initHeap
- DYNCONTEXT_APP_REENTRANT_TYPE reentrant
- unsigned int cmxCriticalCodeSize
- void ∗ cmxCriticalCodeAllocAddr [TOTAL_NUM_SHAVES]
- void ∗ iat
- void ∗ iatnames
- void ∗ iat_group
- void ∗ iatnames_group

7.3.1 Field Documentation

uint64_t∗ DynamicContext_elm::appdynbssdataend

uint64_t∗ DynamicContext_elm::appdynbssdatastart

unsigned char∗ DynamicContext_elm::appdyndata

void∗ DynamicContext_elm::appdyndataAllocAddr[TOTAL_NUM_SHAVES]

unsigned int DynamicContext_elm::appdyndatasize void∗ DynamicContext_elm::cmxCriticalCodeAllocAddr[TOTAL_NUM_SHAVES]

unsigned int DynamicContext_elm::cmxCriticalCodeSize

_ExecutionContext_t∗ DynamicContext_elm::crtContextInfo

_TorFn_t∗ DynamicContext_elm::ctors_end

_TorFn_t∗ DynamicContext_elm::ctors_start

_TorFn_t∗ DynamicContext_elm::dtors_end

_TorFn_t∗ DynamicContext_elm::dtors_start

unsigned char∗ DynamicContext_elm::entryPoint

uint64_t∗ DynamicContext_elm::groupappdynbssdataend

uint64_t∗ DynamicContext_elm::groupappdynbssdatastart

unsigned char∗ DynamicContext_elm::groupappdyndata

unsigned int DynamicContext_elm::groupappdyndatasize

uint32_t DynamicContext_elm::groupEntryPoint

uint32_t DynamicContext_elm::heap_size

void∗ DynamicContext_elm::iat

void∗ DynamicContext_elm::iat_group

void∗ DynamicContext_elm::iatnames

void∗ DynamicContext_elm::iatnames_group

DYNCONTEXT_HEAP_ACTION_TYPE DynamicContext_elm::initHeap

DynamicContextInstancesPtr DynamicContext_elm::instancesData

ParadigmSpecificEntry DynamicContext_elm::pse[TOTAL_NUM_SHAVES]

DYNCONTEXT_APP_REENTRANT_TYPE DynamicContext_elm::reentrant

uint32_t DynamicContext_elm::stack_size

The documentation for this struct was generated from the following file:

- theDynContext.h

7.4 DynamicContextGlobal_elm Struct Reference

```
#include <theDynContext.h>
```

Data Fields

- unsigned int DynamicContextAppsNumber
- DynamicContextInfo_t * DynamicContextGlobalArray

7.4.1 Field Documentation

unsigned int DynamicContextGlobal_elm::DynamicContextAppsNumber

DynamicContextInfo_t* DynamicContextGlobal_elm::DynamicContextGlobalArray

The documentation for this struct was generated from the following file:

- theDynContext.h

7.5 DynamicContextInfo_elm Struct Reference

```
#include <theDynContext.h>
```

Data Fields

- DynamicContext_t * module
- char * ContextName

7.5.1 Field Documentation

char* DynamicContextInfo_elm::ContextName

DynamicContext_t* DynamicContextInfo_elm::module

The documentation for this struct was generated from the following file:

- theDynContext.h

7.6 DynamicContextInstances_elm Struct Reference

```
#include <theDynContext.h>
```

Data Fields

- unsigned char * GrpDataPools [TOTAL_NUM_SHAVES]
- unsigned char * GrpDataPoolsStart [TOTAL_NUM_SHAVES]

- unsigned char ∗ HeapPools [TOTAL_NUM_SHAVES]
- unsigned char ∗ HeapPoolsStart [TOTAL_NUM_SHAVES]
- uint32_t appInstances
- swcShaveUnit_t shaveList [TOTAL_NUM_SHAVES]

7.6.1 Field Documentation

uint32_t DynamicContextInstances_elm::appInstances

unsigned char∗ DynamicContextInstances_elm::GrpDataPools[TOTAL_NUM_SHAVES]

unsigned char∗ DynamicContextInstances_elm::GrpDataPoolsStart[TOTAL_NUM_SHAVES]

unsigned char∗ DynamicContextInstances_elm::HeapPools[TOTAL_NUM_SHAVES]

unsigned char∗ DynamicContextInstances_elm::HeapPoolsStart[TOTAL_NUM_SHAVES]

swcShaveUnit_t DynamicContextInstances_elm::shaveList[TOTAL_NUM_SHAVES]

The documentation for this struct was generated from the following file:

- theDynContext.h

7.7 performanceStruct Struct Reference

```
#include <swcTestUtilsDefines.h>
```

Data Fields

- u32 perfCounterStall

  counts the stalls
- u32 perfCounterExec

  counts the execution cycles
- u32 perfCounterClkCycles

  counts the clock cycles
- u32 perfCounterBranch

  counts the branches taken
- unsigned long long perfCounterTimer
- u32 countShCodeRun

  counts how many times the shave code was executed
- u32 stallsTypes

  enables specific stalls from a given list to be counted
- tyTimeStamp executionTimer

7.7.1 Field Documentation

u32 performanceStruct::countShCodeRun

counts how many times the shave code was executed

tyTimeStamp performanceStruct::executionTimer

assignes its value to perfCounterTimer in order to display total execution (in cycles, [us] and [ms])

u32 performanceStruct::perfCounterBranch

counts the branches taken

u32 performanceStruct::perfCounterClkCycles

counts the clock cycles

u32 performanceStruct::perfCounterExec

counts the execution cycles

u32 performanceStruct::perfCounterStall

counts the stalls

unsigned long long performanceStruct::perfCounterTimer

counts the total execution of the program

u32 performanceStruct::stallsTypes

enables specific stalls from a given list to be counted

The documentation for this struct was generated from the following file:

- swcTestUtilsDefines.h

7.8 swcFifo_t Struct Reference

```
#include <swcFifo.h>
```

Data Fields

- uint8_t ∗ memory
- int32_t size

- int32_t unreadSize
- int32_t writeIndex
- int32_t activeWriteSize
- int32_t readIndex
- int32_t activeReadSize

7.8.1 Field Documentation

int32_t swcFifo_t::activeReadSize

int32_t swcFifo_t::activeWriteSize

uint8_t* swcFifo_t::memory

int32_t swcFifo_t::readIndex

int32_t swcFifo_t::size

int32_t swcFifo_t::unreadSize

int32_t swcFifo_t::writeIndex

The documentation for this struct was generated from the following file:

- swcFifo.h

Chapter 8

File Documentation

8.1 dbgLogEvents.h File Reference

Enumerations

- enum Event_t {
LOG_EVENT_LOS_RUN = 1, LOG_EVENT_LRT_RUN, LOG_EVENT_WAIT_FOR_LRT,
LOG_EVENT_SHAVE_0_RESET = 10,
LOG_EVENT_SHAVE_1_RESET, LOG_EVENT_SHAVE_2_RESET, LOG_EVENT_SHAV-E_3_RESET, LOG_EVENT_SHAVE_4_RESET,
LOG_EVENT_SHAVE_5_RESET, LOG_EVENT_SHAVE_6_RESET, LOG_EVENT_SHAV-E_7_RESET, LOG_EVENT_SHAVE_8_RESET,
LOG_EVENT_SHAVE_9_RESET, LOG_EVENT_SHAVE_10_RESET, LOG_EVENT_SHA-VE_11_RESET, LOG_EVENT_SHAVE_0_RUN,
LOG_EVENT_SHAVE_1_RUN, LOG_EVENT_SHAVE_2_RUN, LOG_EVENT_SHAVE_3_-RUN, LOG_EVENT_SHAVE_4_RUN,
LOG_EVENT_SHAVE_5_RUN, LOG_EVENT_SHAVE_6_RUN, LOG_EVENT_SHAVE_7_-RUN, LOG_EVENT_SHAVE_8_RUN,
LOG_EVENT_SHAVE_9_RUN, LOG_EVENT_SHAVE_10_RUN, LOG_EVENT_SHAVE_-11_RUN, LOG_EVENT_WAIT_FOR_SHAVE_0,
LOG_EVENT_WAIT_FOR_SHAVE_1, LOG_EVENT_WAIT_FOR_SHAVE_2, LOG_EVEN-T_WAIT_FOR_SHAVE_3, LOG_EVENT_WAIT_FOR_SHAVE_4,
LOG_EVENT_WAIT_FOR_SHAVE_5, LOG_EVENT_WAIT_FOR_SHAVE_6, LOG_EVEN-T_WAIT_FOR_SHAVE_7, LOG_EVENT_WAIT_FOR_SHAVE_8,
LOG_EVENT_WAIT_FOR_SHAVE_9, LOG_EVENT_WAIT_FOR_SHAVE_10, LOG_EVE-NT_WAIT_FOR_SHAVE_11, LOG_EVENT_CSS_DIGITAL_POWER,
LOG_EVENT_CSS_ANALOG_POWER, LOG_EVENT_RETENTION, LOG_EVENT_SHA-VE_0_POWER, LOG_EVENT_SHAVE_1_POWER,
LOG_EVENT_SHAVE_2_POWER, LOG_EVENT_SHAVE_3_POWER, LOG_EVENT_SHA-VE_4_POWER, LOG_EVENT_SHAVE_5_POWER,
LOG_EVENT_SHAVE_6_POWER, LOG_EVENT_SHAVE_7_POWER, LOG_EVENT_SHA-VE_8_POWER, LOG_EVENT_SHAVE_9_POWER,
LOG_EVENT_SHAVE_10_POWER, LOG_EVENT_SHAVE_11_POWER, LOG_EVENT_P-MB_POWER, LOG_EVENT_MSS_DIGITAL_POWER,
LOG_EVENT_MSS_ANALOG_POWER, LOG_EVENT_DSS_DIGITAL_POWER, LOG_EV-ENT_DSS_ANALOG_POWER, LOG_EVENT_POWER_M2x5x_BASE = 70,
LOG_EVENT_MSS_CPU_POWER = 86, LOG_EVENT_MSS_AMC_POWER, LOG_EVENT-

_MSS_SIPP_POWER, LOG_EVENT_DSS_POWER, LOG_EVENT_USB_POWER, LOG_EVENT_198_RAIL_BASE = 100, LOG_EVENT_198_RAIL_VDDCV_I_MA = LOG_EVENT_198_RAIL_BASE, LOG_EVENT_198_RAIL_VDDCR_I_MA, LOG_EVENT_198_RAIL_VDDIO_I_MA, LOG_EVENT_198_RAIL_MIPI_VDD_I_MA, LOG_EVENT_198_RAIL_PLL_AVDD_I_MA, LOG_EVENT_198_RAIL_DRAM_MVDDQ_I_MA, LOG_EVENT_198_RAIL_DRAM_MVDDA_I_MA, LOG_EVENT_198_RAIL_DRAM_VDD1_I_MA, LOG_EVENT_198_RAIL_DRAM_VDD2_I_MA, LOG_EVENT_198_RAIL_DRAM_VDDQ_I_MA, LOG_EVENT_198_RAIL_USB_VDD330_I_MA, LOG_EVENT_198_RAIL_USB_VP_VDD_I_MA, LOG_EVENT_198_RAIL_VDDCV_V_MV, LOG_EVENT_198_RAIL_MIPI_VDD_V_MV, LOG_EVENT_198_RAIL_VDDIO_B_I_MUL_I_MA_MA2150, LOG_EVENT_198_TOTAL_CURRENT, LOG_EVENT_198_TOTAL_POWER, LOG_EVENT_198_DDR_CURRENT, LOG_EVENT_198_DDR_POWER, LOG_EVENT_SYS_CLK_CHANGE = 200, LOG_EVENT_LAST_EVENT = 9999 }

8.1.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

8.2 dbgTracerApi.h File Reference

```
#include "logMsg.h"
```

Macros

- #define DEBUG_LOG_LEVEL_LOW LOG_LEVEL_INFO
- #define DEBUG_LOG_LEVEL_MEDIUM LOG_LEVEL_WARNING
- #define DEBUG_LOG_LEVEL_HIGH LOG_LEVEL_ERROR

8.2.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see: common/license.txt

8.3 Fp16Convert.h File Reference

Macros

- #define MOVIDIUS_FP32

- #define F32_NAN_DEFAULT 0xFFC00000
- #define EXTRACT_F16_SIGN(x) ((x >> 15) & 0x1)
- #define EXTRACT_F16_EXP(x) ((x >> 10) & 0x1F)
- #define EXTRACT_F16_FRAC(x) (x & 0x000003FF)
- #define EXTRACT_F32_SIGN(x) ((x >> 31) & 0x1)
- #define EXTRACT_F32_EXP(x) ((x >> 23) & 0xFF)
- #define EXTRACT_F32_FRAC(x) (x & 0x007FFFFF)
- #define RESET_SNAN_BIT(x) x = x | 0x00400000
- #define PACK_F32(x, y, z) ((x << 31) + (y << 23) + z)
- #define PACK_F16(x, y, z) ((x << 15) + (y << 10) + z)
- #define F16_IS_NAN(x) ((x & 0x7FFF)> 0x7C00)
- #define F16_IS_SNAN(x) (((x & 0x7E00) == 0x7C00)&&((x & 0x1FF)> 0))
- #define F32_IS_NAN(x) ((x & 0x7FFFFFFF)> 0x7F800000)
- #define F32_IS_SNAN(x) (((x & 0x7FC00000) == 0x7F800000)&&((x & 0x3FFFFF)> 0))

Rounding modes

- #define F32_RND_NEAREST_EVEN 0
- #define F32_RND_MINUS_INF 1
- #define F32_RND_PLUS_INF 2
- #define F32_RND_TO_ZERO 3

Detect tinyness mode

- #define F32_DETECT_TINY_AFTER_RND 0
- #define F32_DETECT_TINY_BEFORE_RND 1

Exceptions

- #define F32_EX_INEXACT 0x00000001
- #define F32_EX_DIV_BY_ZERO 0x00000002
- #define F32_EX_INVALID 0x00000004
- #define F32_EX_UNDERFLOW 0x00000008
- #define F32_EX_OVERFLOW 0x00000010

Functions

- unsigned int f32Tof16 (float x)

  Convert fp32 to fp16 param[in] x - float(fp32) input to be converted.
- float f16Tof32 (unsigned int x)

  Convert fp16 to fp32 param[in] x - fp16 input to be converted.

8.3.1 Detailed Description

Copyright

8.4 logMsg.h File Reference

```
#include <stdlib.h>
#include <stdint.h>
#include <stdio.h>
#include <dbgLogEvents.h>
#include <stdarg.h>
#include <time.h>
#include <pthread.h>
```

Macros

- #define __PC__

  Trace Logging Header File.
- #define _GNU_SOURCE
- #define SINK_FUNCTION _printf_clone
- #define SINK_BULK _bulk_hexdump
- #define TRACE_BUFFER_SIZE (1024∗1024)
- #define CRITICAL_SECTION_ENTER
- #define CRITICAL_SECTION_EXIT
- #define STR_IMPL_(x) #x
- #define STR(x) STR_IMPL_(x)
- #define LOG_LEVEL_FATAL 1
- #define LOG_LEVEL_ERROR 2
- #define LOG_LEVEL_WARNING 3
- #define LOG_LEVEL_INFO 4
- #define LOG_LEVEL_DEBUG 5
- #define LOG_LEVEL_TRACE 6
- #define DBG_FATAL "<" STR(LOG_LEVEL_FATAL) ">"
- #define DBG_ERROR "<" STR(LOG_LEVEL_ERROR) ">"
- #define DBG_WARNING "<" STR(LOG_LEVEL_WARNING) ">"
- #define DBG_INFO "<" STR(LOG_LEVEL_INFO) ">" /∗ default ∗/
- #define DBG_DEBUG "<" STR(LOG_LEVEL_DEBUG) ">"
- #define DBG_TRACE "<" STR(LOG_LEVEL_TRACE) ">"
- #define DBG_CHAR_LOG_TYPE(lvl) ('0' + lvl)
- #define DBG_MAX_LEVEL DBG_CHAR_LOG_TYPE(LOG_LEVEL_TRACE)
- #define DEFAULT_LOG_LEVEL DBG_CHAR_LOG_TYPE(LOG_LEVEL_INFO)
- #define MAX_STATIC_LOG_LEVEL LOG_LEVEL_TRACE
- #define MV_DBG_FMT_STR_SIZE 256u
- #define MV_UNIT_NAME _
- #define _MV_LOG_LEVEL(UNIT) UNIT ## _traceLogLevel
- #define MV_LOG_LEVEL(UNIT) _MV_LOG_LEVEL(UNIT)
- #define _traceLogLevel MV_LOG_LEVEL(MV_UNIT_NAME)
- #define TIMER_ADDR TIM0_BASE_ADR
- #define FP_TIME_READ() time(NULL)
- #define DBG_PRINT_FILE_LINE "[File: %s, Line: %d]\t", __FILE__, __LINE__

- #define DBG_PRINT_MODULE_NAME "[Module: " STR(MV_UNIT_NAME) "]\t"
- #define DBG_PRINT_TIMESTAMP "[Timestamp: %lld]\t", FP_TIME_READ()
- #define DBG_PRINT_LOG_LEVEL "[Severity: %c]\t", _traceLogLevel
- #define DBG_PRINT_THREAD "[Thread: %s, Id: 0x%lx]\t", getMyThreadName(), pthread_-self()
- #define DBG_PRINT_CORE_ID "[CPU: x86]\t"
- #define _FIRST_ARG(a,...) a
- #define FIRST_ARG(...) _FIRST_ARG(__VA_ARGS__)
- #define _SECOND_ARG(a,...) , ##__VA_ARGS__
- #define SECOND_ARG(...) _SECOND_ARG(__VA_ARGS__)
- #define FL_STR1
- #define FL_ARG1
- #define FL_STR2
- #define FL_ARG2
- #define FL_STR3
- #define FL_ARG3
- #define FL_STR4
- #define FL_ARG4
- #define FL_STR5
- #define FL_ARG5
- #define FL_STR6
- #define FL_ARG6
- #define FL_STR FL_STR1 FL_STR2 FL_STR3 FL_STR4 FL_STR5 FL_STR6
- #define FL_ARGS FL_ARG1 FL_ARG2 FL_ARG3 FL_ARG4 FL_ARG5 FL_ARG6
- #define LOG_TRACE(fmt,...) logMsg(DBG_TRACE FL_STR fmt FL_ARGS, ##__VA_ARGS-__)
- #define LOG_BULK_TRACE(data, size) logBulk(LOG_LEVEL_TRACE, data, size)
- #define LOG_TRACE_EVENT(id, data) dbgLogEvent(id, data, LOG_LEVEL_TRACE)
- #define LOG_DEBUG(fmt,...) logMsg(DBG_DEBUG FL_STR fmt FL_ARGS, ##__VA_ARG-S__)
- #define LOG_BULK_DEBUG(data, size) logBulk(LOG_LEVEL_DEBUG, data, size)
- #define LOG_DEBUG_EVENT(id, data) dbgLogEvent(id, data, LOG_LEVEL_DEBUG)
- #define LOG_INFO(fmt,...) logMsg(DBG_INFO FL_STR fmt FL_ARGS, ##__VA_ARGS__)
- #define LOG_BULK_INFO(data, size) logBulk(LOG_LEVEL_INFO, data, size)
- #define LOG_INFO_EVENT(id, data) dbgLogEvent(id, data, LOG_LEVEL_INFO)
- #define LOG_WARNING(fmt,...) logMsg(DBG_WARNING FL_STR fmt FL_ARGS, ##__VA-_ARGS__)
- #define LOG_BULK_WARNING(data, size) logBulk(LOG_LEVEL_WARNING, data, size)
- #define LOG_WARNING_EVENT(id, data) dbgLogEvent(id, data, LOG_LEVEL_WARNING)
- #define LOG_ERROR(fmt,...) logMsg(DBG_ERROR FL_STR fmt FL_ARGS, ##__VA_ARGS-__)
- #define LOG_BULK_ERROR(data, size) logBulk(LOG_LEVEL_ERROR, data, size)
- #define LOG_ERROR_EVENT(id, data) dbgLogEvent(id, data, LOG_LEVEL_ERROR)
- #define LOG_FATAL(fmt,...) logMsg(DBG_FATAL FL_STR fmt FL_ARGS, ##__VA_ARGS-__)
- #define LOG_BULK_FATAL(data, size) logBulk(LOG_LEVEL_FATAL, data, size)
- #define LOG_FATAL_EVENT(id, data) dbgLogEvent(id, data, LOG_LEVEL_FATAL)
- #define dbgLogEvent(a, b, c) (void)(a);(void)(b);(void)(c)
- #define _dbgLogPlainMessage(a, b) (void)(a);(void)(b)

Functions

- void SINK_FUNCTION (const char ∗__restrict msg)
- void SINK_BULK (void ∗__restrict data, size_t size)
- __attribute__ ((weak)) int MV_LOG_LEVEL(MV_UNIT_NAME)
- __attribute__ ((no_instrument_function)) static inline char ∗getMyThreadName(void)
- void _printf_clone (const char ∗__restrict msg)
- void logBulk (const int level, void ∗__restrict data, size_t size)
- __attribute__ ((weak, no_instrument_function)) void logBulk(const int level

  Log bulk data.
- printf ("\n")
- __attribute__ ((format(printf, 1, 2), nonnull(1), no_instrument_function)) static inline void logMsg(const char ∗__restrict format
- va_start (ap, format)
- if (nbBytes< 0)
- va_end (ap)
- SINK_FUNCTION (buffer)
- SINK_BULK (data, size)

Variables

- size_t size
- char buffer [MV_DBG_FMT_STR_SIZE]
- int nbBytes = __builtin_vsnprintf(buffer, MV_DBG_FMT_STR_SIZE, format, ap)
- void ∗__restrict data

8.4.1 Macro Definition Documentation

#define __PC__

Trace Logging Header File.

== Sample usage:

// print a simple string LOG_INFO("This is a info string with default level 4\n");

// print formatted strings as well. Uses printf-like syntax LOG_ERROR("Now we are printing error: %d\n", errno);

// dump binary data LOG_BULK_TRACE(data, size);

// send an event LOG_TRACE_EVENT(id, data);

Configuration: Add the following defines to the compiler using -D switch or simply #re-define in code

MV_UNIT_NAME - Used to set the dynamic log level per unit basis. Define it before including this header. Default is empty When the unit name is defined, a weak int will be generated with <unit name>="">_traceLogLevel name otherwise will be simply `int _traceLogLevel;`

DEFAULT_LOG_LEVEL - Is the default log level. If not set it is LOG_LEVEL_INFO(4) Everything above this value will be ignored

MAX_STATIC_LOG_LEVEL - Compile out all messages above this value

SINK_FUNCTION - the function used to print printf-like strings void SINK_FUNCTION(const char∗ __restrict msg); Predefined values:

- _printf_clone (default) - use the standard printf function
- _trace_print - use the TraceProfiler

SINK_BULK - function dumping out bulk messages void SINK_BULK(const int level, void ∗ __restrict data, size_t size); Predefined values:

- _bulk_hexdump (default) - write using printf the data as hex values

TRACE_BUFFER_SIZE - size in bytes for the TraceProfiler buffer

MV_DBG_FMT_STR_SIZE - size in bytes of the formatted string for printf-like functions The default value is 256

DBG_ARGx (x=1..6) - prefix used in DBG_LOG()/DBG_ERROR()/... The values are a printf-like argument, and several predefined values can be used The number x represent the position in string. Gaps are allowed. Numbers out-of-range will be silently ignored Predefined values: DBG_PRINT_FILE_LINE DBG_PRINT_MODULE_NAME DBG_PRINT_TIMESTAMP DBG_PRINT_LOG_LEVEL DBG_PRINT_THREAD DBG_PRINT_CORE_ID

Example: #define DBG_ARG4 DBG_PRINT_THREAD #define DBG_ARG2 DBG_PRINT_TIMESTAMP gcc -DDBG_ARG6=DBG_PRINT_FILE_LINE

SYNC_LOG_MSG - Put sink function in a critical section (mutex for shave/disable interrupts for rtems/set max interrupt level for bm) Unset by default for speed purposes. Define this if you see scrambled messages

For C++ there are extra two functions: static inline void logMessage(int logLevel, const char∗ __restrict format, ...); static inline void logMessage(int logLevel, const char∗ __restrict format, va_list args);

For convenience, also can be ostreams like cout (mv_fatal, mv_err, mv_warn, etc): mv_info << "This is info message with value " << 101 << "\n";

#define _dbgLogPlainMessage(a, b) (void)(a);(void)(b)

#define _FIRST_ARG(a, ...) a

#define _GNU_SOURCE

#define _MV_LOG_LEVEL(UNIT) UNIT ## **_traceLogLevel**

#define _SECOND_ARG(a, ...) , ##__VA_ARGS__

#define _traceLogLevel **MV_LOG_LEVEL**(**MV_UNIT_NAME**)

#define CRITICAL_SECTION_ENTER

Value:

```
do { \
    ShDrvMutexRequest(4);
```

#define CRITICAL_SECTION_EXIT

Value:

```
ShDrvMutexRelease(4); \
    } while (0);
```

#define DBG_CHAR_LOG_TYPE(lvl) ('0' + lvl)

#define DBG_DEBUG "<" **STR**(**LOG_LEVEL_DEBUG**) ">"

#define DBG_ERROR "<" **STR**(**LOG_LEVEL_ERROR**) ">"

#define DBG_FATAL "<" **STR**(**LOG_LEVEL_FATAL**) ">"

#define DBG_INFO "<" **STR**(**LOG_LEVEL_INFO**) ">" /* default */

#define DBG_MAX_LEVEL **DBG_CHAR_LOG_TYPE**(**LOG_LEVEL_TRACE**)

#define DBG_PRINT_CORE_ID "[CPU: x86]\t"

#define DBG_PRINT_FILE_LINE "[File: %s, Line: %d]\t", __FILE__, __LINE__

#define DBG_PRINT_LOG_LEVEL "[Severity: %c]\t", _traceLogLevel

#define DBG_PRINT_MODULE_NAME "[Module: " STR(**MV_UNIT_NAME**) "]\t"

#define DBG_PRINT_THREAD "[Thread: %s, Id: 0x%lx]\t", getMyThreadName(), pthread_self()

#define DBG_PRINT_TIMESTAMP "[Timestamp: %lld]\t", FP_TIME_READ()

#define DBG_TRACE "<" **STR**(**LOG_LEVEL_TRACE**) ">"

#define DBG_WARNING "<" **STR**(**LOG_LEVEL_WARNING**) ">"

#define dbgLogEvent(a, b, c) (void)(a);(void)(b);(void)(c)

#define DEFAULT_LOG_LEVEL **DBG_CHAR_LOG_TYPE**(**LOG_LEVEL_INFO**)

#define FIRST_ARG(...) **_FIRST_ARG**(__VA_ARGS__)

#define FL_ARG1

#define FL_ARG2

#define FL_ARG3

#define FL_ARG4

#define FL_ARG5

#define FL_ARG6

#define FL_ARGS **FL_ARG1 FL_ARG2 FL_ARG3 FL_ARG4 FL_ARG5 FL_ARG6**

#define FL_STR **FL_STR1 FL_STR2 FL_STR3 FL_STR4 FL_STR5 FL_STR6**

#define FL_STR1

#define FL_STR2

#define FL_STR3

#define FL_STR4

#define FL_STR5

#define FL_STR6

#define FP_TIME_READ( ) time(NULL)

#define LOG_BULK_DEBUG(**data**, **size**) **logBulk**(**LOG_LEVEL_DEBUG**, **data**, **size**)

#define LOG_BULK_ERROR(**data**, **size**) **logBulk**(**LOG_LEVEL_ERROR**, **data**, **size**)

#define LOG_BULK_FATAL(**data**, **size**) **logBulk**(**LOG_LEVEL_FATAL**, **data**, **size**)

#define LOG_BULK_INFO(**data**, **size**) **logBulk**(**LOG_LEVEL_INFO**, **data**, **size**)

#define LOG_BULK_TRACE(**data**, **size**) **logBulk**(**LOG_LEVEL_TRACE**, **data**, **size**)

#define LOG_BULK_WARNING(**data**, **size**) **logBulk**(**LOG_LEVEL_WARNING**, **data**, **size**)

#define LOG_DEBUG(fmt, ...) logMsg(**DBG_DEBUG FL_STR** fmt **FL_ARGS**, ##__VA_ARGS__)

#define LOG_DEBUG_EVENT(id, **data**) **dbgLogEvent**(id, **data**, **LOG_LEVEL_DEBUG**)

#define LOG_ERROR(fmt, ...) logMsg(**DBG_ERROR FL_STR** fmt **FL_ARGS**, ##__VA_ARGS__)

#define LOG_ERROR_EVENT(id, **data**) **dbgLogEvent**(id, **data**, **LOG_LEVEL_ERROR**)

#define LOG_FATAL(fmt, ...) logMsg(**DBG_FATAL FL_STR** fmt **FL_ARGS**, ##__VA_ARGS__)

#define LOG_FATAL_EVENT(id, **data**) **dbgLogEvent**(id, **data**, **LOG_LEVEL_FATAL**)

#define LOG_INFO(fmt, ...) logMsg(**DBG_INFO FL_STR** fmt **FL_ARGS**, ##__VA_ARGS__)

#define LOG_INFO_EVENT(id, **data**) **dbgLogEvent**(id, **data**, **LOG_LEVEL_INFO**)

#define LOG_LEVEL_DEBUG 5

#define LOG_LEVEL_ERROR 2

#define LOG_LEVEL_FATAL 1

#define LOG_LEVEL_INFO 4

#define LOG_LEVEL_TRACE 6

#define LOG_LEVEL_WARNING 3

#define LOG_TRACE(fmt, ...) logMsg(**DBG_TRACE FL_STR** fmt **FL_ARGS**, ##__VA_ARGS__)

#define LOG_TRACE_EVENT(id, **data**) **dbgLogEvent**(id, **data**, **LOG_LEVEL_TRACE**)

#define LOG_WARNING(fmt, ...) logMsg(**DBG_WARNING FL_STR** fmt **FL_ARGS**, ##__VA_ARGS__)

#define LOG_WARNING_EVENT(id, **data**) **dbgLogEvent**(id, **data**, **LOG_LEVEL_WARNING**)

#define MAX_STATIC_LOG_LEVEL **LOG_LEVEL_TRACE**

#define MV_DBG_FMT_STR_SIZE 256u

#define MV_LOG_LEVEL(UNIT) **_MV_LOG_LEVEL**(UNIT)

#define MV_UNIT_NAME _

#define SECOND_ARG(...) **_SECOND_ARG**(__VA_ARGS__)

#define SINK_BULK _bulk_hexdump

#define SINK_FUNCTION **_printf_clone**

#define STR(x) **STR_IMPL_**(x)

#define STR_IMPL_(x) #x

#define TIMER_ADDR TIM0_BASE_ADR

#define TRACE_BUFFER_SIZE (1024∗1024)

8.4.2 Function Documentation

__attribute__ ((weak))

__attribute__ ((no_instrument_function))

__attribute__ ((weak, no_instrument_function)) const

Log bulk data.

All bulk data will be dumped in a file on disk by moviProf when an appropriate sink is used logBulk symol is weak in order for clients to be able to replace it with their own implementation

Parameters

level	log level of the data
data	is the binary message to be logged
size	data size

__attribute__ ((format(**printf**, 1, 2), nonnull(1), no_instrument_function)) const

Log formatted string messages

Parameters

format	
...	static because __builtin_va_arg_pack()

void _printf_clone (const char ∗__restrict msg)

if ( )

void logBulk (const int level, void ∗__restrict data, size_t size)

printf ("\n")

Referenced by __attribute__().

void SINK_BULK (void ∗__restrict data, size_t size)

SINK_BULK (**data** , **size**)

void SINK_FUNCTION (const char ∗__restrict msg)

SINK_FUNCTION (**buffer**)

va_end (ap)

va_start (ap , format)

8.4.3 Variable Documentation

char buffer[**MV_DBG_FMT_STR_SIZE**]

void∗ __restrict data

int nbBytes = __builtin_vsnprintf(**buffer**, **MV_DBG_FMT_STR_SIZE**, format, ap)

void ∗__restrict size_t size

Initial value:

```
{
    for (unsigned i=0; i<size; i++) {
        uint8_t _byte = ((uint8_t*)data)[i];
        printf("0x%x%c", _byte, i%10==9?'\n':' ');
        UNUSED(_byte);
    }
```

8.5 MDKdox-LeonUtils-intro.txt File Reference

8.6 swcCdmaCommon.h File Reference

```
#include "swcCdmaCommonDefines.h"
```

Functions

- dmaRequesterId dmaInitRequester (int priority)

  Initialize a requester ID which will be used to properly initialize and distinguish single tasks or groups of tasks.
- dmaTransactionList ∗ dmaCreateTransactionFullOptions (dmaRequesterId ReqId, dmaTransactionList ∗NewTransaction, u8 ∗Src, u8 ∗Dst, u32 ByteLength, u32 SrcLineWidth, u32 DstLineWidth, s32 SrcStride, s32 DstStride)

  Initialize a new CMXDMA task structure which can be used to realize a DMA data transfer using source and destination strides.
- dmaTransactionList ∗ dmaCreateTransaction (dmaRequesterId ReqId, dmaTransactionList ∗NewTransaction, u8 ∗Src, u8 ∗Dst, u32 ByteLength)

  Initialize a new CMXDMA task structure which can be used to realize a simple DMA data transfer.
- dmaTransactionList ∗ dmaCreateTransactionSrcStride (dmaRequesterId ReqId, dmaTransactionList ∗NewTransaction, u8 ∗Src, u8 ∗Dst, u32 ByteLength, u32 LineWidth, s32 SrcStride)

  Initialize a new CMXDMA task structure which can be used to realize a DMA data transfer using source stride only.
- dmaTransactionList ∗ dmaCreateTransactionDstStride (dmaRequesterId ReqId, dmaTransactionList ∗NewTransaction, u8 ∗Src, u8 ∗Dst, u32 ByteLength, u32 LineWidth, s32 DstStride)

  Initialize a new CMXDMA task structure which can be used to realize a DMA data transfer using destination stride only.
- dmaTransactionList ∗ dmaCreate3DTransaction (dmaRequesterId ReqId, dmaTransactionList ∗NewTransaction, u8 ∗Src, u8 ∗Dst, u32 ByteLength, u32 SrcLineWidth, u32 DstLineWidth, s32 SrcStride, s32 DstStride, u32 NumPlanes, s32 SrcPlaneStride, s32 DstPlaneStride)

  Creates a new 3D transaction.
- void dmaLinkTasks (dmaTransactionList ∗listHead, u32 nmbTasks,...)

  Link multiple tasks in a single linked list. Please note that this function allows linking just for single tasks.
- int dmaStartListTask (dmaTransactionList ∗ListPtr)

  Set-up CMXDMA to execute the given list of tasks.
- void dmaWaitTask (dmaTransactionList ∗ListPtr)

  Wait in a blocking way for a given task to finish.
- int dmaIsTaskFinished (dmaTransactionList ∗ListPtr)

  Check whether a task finished it's execution or is still running/pending.

8.6.1 Detailed Description

Copyright

8.7 swcCdmaCommonDefines.h File Reference

```
#include "CmxDma.h"
```

Data Structures

- struct configBits
  Bit field for fine-grained configuration of CMXDMA transaction.
- struct dmaTransactionList_t
  2D transaction type

Macros

- #define ALIGNED8 __attribute__ ((aligned (8)))
- #define SVU_SLICE_OFFSET 0x10000
- #define SWC_CMX_DMA_DEFAULT_NUM_PLANE (0)
- #define SWC_CMX_DMA_DEFAULT_PLANE_STRIDE (0)
- #define MIN_NUM_PLANES (1)
- #define MAX_NUM_PLANES (256)

Typedefs

- typedef dmaTransactionList_t dmaTransactionList
- typedef void(∗ dmaIrqHandler)(dmaTransactionList ∗ListPtr, void ∗userContext)

8.7.1 Detailed Description

Copyright

8.8 swcCrc.h File Reference

```
#include "mv_types.h"
#include <swcLeonUtils.h>
```

Functions

- u32 swcCalcCrc32 (u8 ∗pBuffer, u32 byteLength, pointer_type pt)

  Calculate simple CRC32 over a byte buffer of byteLength.

8.8.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see-: common/license.txt

8.9 swcFifo.h File Reference

```
#include <stdint.h>
```

Data Structures

- struct swcFifo_t

Typedefs

- typedef struct swcFifo_t swcFifo_t

Functions

- int32_t swcFifoGetBasePtr (struct swcFifo_t ∗hndl, void ∗∗ptr)
- int32_t swcFifoInit (struct swcFifo_t ∗hndl, void ∗buffer, int32_t size)
- int32_t swcFifoPush32Bit (struct swcFifo_t ∗hndl, uint32_t data)
- int32_t swcFifoPush16Bit (struct swcFifo_t ∗hndl, uint16_t data)
- int32_t swcFifoPush8Bit (struct swcFifo_t ∗hndl, uint8_t data)
- int32_t swcFifoPop32Bit (struct swcFifo_t ∗hndl, uint32_t ∗data)
- int32_t swcFifoPop16Bit (struct swcFifo_t ∗hndl, uint16_t ∗data)
- int32_t swcFifoPop8Bit (struct swcFifo_t ∗hndl, uint8_t ∗data)
- int32_t swcFifoGetWritePtr (struct swcFifo_t ∗hndl, void ∗∗ptr, uint32_t reqLen)
- int32_t swcFifoMarkWriteDone (struct swcFifo_t ∗hndl)
- int32_t swcFifoGetReadPtr (struct swcFifo_t ∗hndl, void ∗∗ptr, uint32_t reqLen)
- int32_t swcFifoMarkReadDone (struct swcFifo_t ∗hndl)
- uint32_t swcFifoAvailable (struct swcFifo_t ∗hndl)
- uint32_t swcFifoContigAvailable (struct swcFifo_t ∗hndl)
- uint32_t swcFifoLength (struct swcFifo_t ∗hndl)

8.9.1 Typedef Documentation

typedef struct **swcFifo_t** **swcFifo_t**

8.9.2 Function Documentation

uint32_t swcFifoAvailable (struct **swcFifo_t** ∗ hndl)

uint32_t swcFifoContigAvailable (struct **swcFifo_t** ∗ hndl)

int32_t swcFifoGetBasePtr (struct **swcFifo_t** ∗ hndl, void ∗∗ ptr)

int32_t swcFifoGetReadPtr (struct **swcFifo_t** ∗ hndl, void ∗∗ ptr, uint32_t reqLen)

int32_t swcFifoGetWritePtr (struct **swcFifo_t** ∗ hndl, void ∗∗ ptr, uint32_t reqLen)

int32_t swcFifoInit (struct **swcFifo_t** ∗ hndl, void ∗ buffer, int32_t size)

uint32_t swcFifoLength (struct **swcFifo_t** ∗ hndl)

int32_t swcFifoMarkReadDone (struct **swcFifo_t** ∗ hndl)

int32_t swcFifoMarkWriteDone (struct **swcFifo_t** ∗ hndl)

int32_t swcFifoPop16Bit (struct **swcFifo_t** ∗ hndl, uint16_t ∗ data)

int32_t swcFifoPop32Bit (struct **swcFifo_t** ∗ hndl, uint32_t ∗ data)

int32_t swcFifoPop8Bit (struct **swcFifo_t** ∗ hndl, uint8_t ∗ data)

int32_t swcFifoPush16Bit (struct **swcFifo_t** ∗ hndl, uint16_t data)

int32_t swcFifoPush32Bit (struct **swcFifo_t** ∗ hndl, uint32_t data)

int32_t swcFifoPush8Bit (struct **swcFifo_t** ∗ hndl, uint8_t data)

8.10 swcLeonMath.h File Reference

```
#include <mv_types.h>
```

Functions

- float swcMathSinf (float angle)

  Utility trigonometric function to calculate the sine of an angle.
- float swcMathCosf (float angle)
- u32 swcIPow (u32 base, u32 exp)

  Utility Integer function to raise base^exp.
- double swcLongLongToDouble (unsigned long long longVal)

  Utility function to cast a 64 bit int to a double.

8.10.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see-: common/license.txt

8.11 swcLeonUtils.h File Reference

```
#include "swcLeonUtilsDefines.h"
#include <mv_types.h>
```

Macros

- #define NATIVE_POINTER_TYPE le_pointer
- #define swcLeonSwapU32(value)

  Swaps endianness of a 32-bit integer (usefull when sharing data between Leon and Shave)
- #define swcLeonSwapU16(value)

  Swaps endianness of a 16-bit integer (usefull when sharing data between Leon and Shave)
- #define swcLeonReadNoCacheU8(addr)

  Reads data bypassing leon LRAM cache.
- #define swcLeonReadNoCacheI8(addr)

  Reads data bypassing leon LRAM cache.
- #define swcLeonReadNoCacheU16(addr)

  Reads data bypassing leon LRAM cache.
- #define swcLeonReadNoCacheI16(addr)

  Reads data bypassing leon LRAM cache.
- #define swcLeonReadNoCacheU32(addr)

  Reads data bypassing leon LRAM cache.
- #define swcLeonReadNoCacheI32(addr) ((int)swcLeonReadNoCacheU32(addr))

  Reads data bypassing leon LRAM cache.
- #define swcLeonReadNoCacheU64(addr)

  Reads data bypassing leon L1 cache.
- #define swcLeonReadNoCacheI64(addr) ((s64)swcLeonReadNoCacheU64(addr))

  Reads data bypassing leon L1 cache.
- #define swcLeonWriteNoCache8(addr, data)

  Writes data bypassing leon LRAM cache.
- #define swcLeonWriteNoCache16(addr, data)

  Writes data bypassing leon LRAM cache.
- #define swcLeonWriteNoCache32(addr, data)

  Writes data bypassing leon LRAM cache.
- #define swcLeonWriteNoCache64(addr, data)

  Writes data bypassing leon L1 cache.
- #define swcLeonFlushCaches() asm volatile("flush" ::: "memory")

  Flush Leon Instruction and Data Caches.

- #define swcLeonDataCacheFlush()

  Flush Leon Data Cache.
- #define swcLeonFlushDcache() swcLeonDataCacheFlush()
- #define swcLeonDataCacheFlushNoWait() swcLeonDataCacheFlush()
- #define swcLeonInstructionCacheFlush()

  Flush Leon Instruction Cache.
- #define swcLeonFlushIcache() swcLeonInstructionCacheFlush()
- #define swcLeonIsCacheFlushPending()

  Check if Leon cache flush is pending.
- #define swcLeonEnableCaches(flush)

  Enable Leon Instruction and Data Caches.
- #define swcLeonEnableIcache(flush)

  Enable Leon Instruction Cache.
- #define swcLeonEnableDcache(flush)

  Enable Leon Data Cache.
- #define swcLeonDisableCaches() asm volatile("sta %%g0, [%%g0] 2" ::: "memory")

  Disable Leon Instruction and Data Caches.
- #define swcLeonDisableDcache()

  Disable Leon Data Cache.
- #define swcLeonDisableIcache()

  Disable Leon Instruction Cache.
- #define swcLeonDisableTraps()

  Disable traps.
- #define swcLeonEnableTraps()

  Enable traps.
- #define swcLeonL1DForceCacheLineMiss(addr) swcRead32Asi01(addr)

  Force a Leon L1 data cache miss.

Enumerations

- enum pointer_type { be_pointer, le_pointer }

  Pointer type.

Functions

- void swcLeonDataCacheFlushBlockWhilePending (void)

  Flushes Leon data cache, and wait while the flush is pending. (DO NOT USE)
- void swcLeonHalt (void)

  Stops Leon.
- int swcLeonSetPIL (u32 pil)

  Sets the Processor Interrupt Level atomically.
- void swcLeonFlushWindows (void)

  Flushes all the interrupt windows before the caller's to the stack.
- void swcLeonMemCpy (void ∗dst, pointer_type dst_pt, const void ∗src, pointer_type src_pt, u32 count)

Generic memory copying function to copy le/be buffers to le/be buffers.

- void swcLeonMemMove (void ∗dst, pointer_type dst_pt, const void ∗src, pointer_type src_pt, u32 count)

  Same as swcLeonMemCpy, except buffers may overlap.

- void swcLeonSwapBuffer (void ∗buf, pointer_type pt, u32 count)

  Swap the endianness of a buffer in place.

8.11.1 Detailed Description

Copyright

8.12 swcLeonUtilsDefines.h File Reference

Macros

- #define MASK_PSR_impl 0xf0000000
- #define POS_PSR_impl 28
- #define MASK_PSR_ver 0x0f000000
- #define POS_PSR_ver 24
- #define MASK_PSR_icc 0x00f00000
- #define POS_PSR_icc 20
- #define PSR_N 0x00800000
- #define PSR_Z 0x00400000
- #define PSR_V 0x00200000
- #define PSR_C 0x00100000
- #define PSR_EC 0x00002000
- #define PSR_EF 0x00001000
- #define MASK_PSR_PIL 0x00000f00
- #define POS_PSR_PIL 8
- #define PSR_PIL0 0x00000000
- #define PSR_PIL1 0x00000100
- #define PSR_PIL2 0x00000200
- #define PSR_PIL3 0x00000300
- #define PSR_PIL4 0x00000400
- #define PSR_PIL5 0x00000500
- #define PSR_PIL6 0x00000600
- #define PSR_PIL7 0x00000700
- #define PSR_PIL8 0x00000800
- #define PSR_PIL9 0x00000900
- #define PSR_PIL10 0x00000a00
- #define PSR_PIL11 0x00000b00
- #define PSR_PIL12 0x00000c00
- #define PSR_PIL13 0x00000d00
- #define PSR_PIL14 0x00000e00

- #define PSR_PIL15 0x00000f00
- #define PSR_S 0x00000080
- #define PSR_PS 0x00000040
- #define PSR_ET 0x00000020
- #define MASK_PSR_CWP 0x0000001f
- #define POS_PSR_CWP 0
- #define PSR_CWP0 0x00000000
- #define PSR_CWP1 0x00000001
- #define PSR_CWP2 0x00000002
- #define PSR_CWP3 0x00000003
- #define PSR_CWP4 0x00000004
- #define PSR_CWP5 0x00000005
- #define PSR_CWP6 0x00000006
- #define PSR_CWP7 0x00000007
- #define MASK_WIM_BITS 0x000000ff
- #define WIM_INVD0 0x00000001
- #define WIM_INVD1 0x00000002
- #define WIM_INVD2 0x00000004
- #define WIM_INVD3 0x00000008
- #define WIM_INVD4 0x00000010
- #define WIM_INVD5 0x00000020
- #define WIM_INVD6 0x00000040
- #define WIM_INVD7 0x00000080
- #define MASK_TBR_tba 0xfffff000
- #define POS_TBR_tba 12
- #define MASK_TBR_tt 0x00000ff0
- #define POS_TBR_tt 4
- #define TBR_tt_reset 0x000
- #define TBR_tt_instr_access_exception 0x010
- #define TBR_tt_illegal_instr 0x020
- #define TBR_tt_privileged_instr 0x030
- #define TBR_tt_fp_disabled 0x040
- #define TBR_tt_window_overflow 0x050
- #define TBR_tt_window_underflow 0x060
- #define TBR_tt_mem_address_not_aligned 0x070
- #define TBR_tt_fp_exception 0x080
- #define TBR_tt_data_access_exception 0x090
- #define TBR_tt_tag_overflow 0x0A0
- #define TBR_tt_watchpoint 0x0B0
- #define TBR_tt_IRQ1 0x110
- #define TBR_tt_IRQ2 0x120
- #define TBR_tt_IRQ3 0x130
- #define TBR_tt_IRQ4 0x140
- #define TBR_tt_IRQ5 0x150
- #define TBR_tt_IRQ6 0x160
- #define TBR_tt_IRQ7 0x170
- #define TBR_tt_IRQ8 0x180

- #define TBR_tt_IRQ9 0x190
- #define TBR_tt_IRQ10 0x1A0
- #define TBR_tt_IRQ11 0x1B0
- #define TBR_tt_IRQ12 0x1C0
- #define TBR_tt_IRQ13 0x1D0
- #define TBR_tt_IRQ14 0x1E0
- #define TBR_tt_IRQ15 0x1F0
- #define TBR_tt_r_register_access_error 0x200
- #define TBR_tt_instr_access_error 0x210
- #define TBR_tt_cp_disabled 0x240
- #define TBR_tt_unimplemented_FLUSH 0x250
- #define TBR_tt_cp_exception 0x280
- #define TBR_tt_data_access_error 0x290
- #define TBR_tt_division_by_0 0x2A0
- #define TBR_tt_data_store_error 0x2B0
- #define TBR_tt_data_access_MMU_miss 0x2C0
- #define TBR_tt_instr_access_MMU_miss 0x3C0
- #define TBR_tt_user_trap_0 0x800
- #define TBR_tt_user_trap_127 0xFF0
- #define MASK_FSR_RD 0xC0000000
- #define POS_FSR_RD 30
- #define FSR_RD_NEAREST 0x00000000
- #define FSR_RD_ZERO 0x40000000
- #define FSR_RD_INF 0x80000000
- #define FSR_RD_NINF 0xC0000000
- #define MASK_FSR_TEM 0x0f800000
- #define POS_FSR_TEM 25
- #define FSR_NVM 0x08000000
- #define FSR_OFM 0x04000000
- #define FSR_UFM 0x02000000
- #define FSR_DZM 0x01000000
- #define FSR_NXM 0x00800000
- #define FSR_NS 0x00400000
- #define MASK_FSR_ver 0x000E0000
- #define POS_FSR_ver 17
- #define MASK_FSR_tt 0x0001C000
- #define POS_FSR_rrm 14
- #define FSR_tt_NONE 0x00000000
- #define FSR_tt_IEEE 0x00004000
- #define FSR_tt_UNF 0x00008000
- #define FSR_tt_SEQUENCE 0x00010000
- #define FSR_QNE 0x00002000
- #define MASK_FSR_fcc 0x00000C00
- #define POS_FSR_fcc 10
- #define FSR_EQ 0x00000000
- #define FSR_LT 0x00000400
- #define FSR_GT 0x00000800

- #define FSR_UNORDERED 0x00000C00
- #define MASK_FSR_AEXC 0x000003E0
- #define POS_FSR_AEXC 5
- #define FSR_NVA 0x00000200
- #define FSR_OFA 0x00000100
- #define FSR_UFA 0x00000080
- #define FSR_DFA 0x00000040
- #define FSR_NXA 0x00000020
- #define MASK_FSR_CEXC 0x0000001F
- #define POS_FSR_CEXC 0
- #define FSR_NVC 0x00000010
- #define FSR_OFC 0x00000008
- #define FSR_UFC 0x00000004
- #define FSR_DFC 0x00000002
- #define FSR_NXC 0x00000001
- #define MASK_HBRK_ADDR 0xC0000000
- #define LEON_PROCESSOR_INDEX_MASK (1 << 28)
- #define ASR17_DWT (0x00004000)
- #define ASR17_SVT (0x00002000)
- #define __CCR_ASI 0x02
- #define __CCR_OFS 0x00000000
- #define CACHE_CONTROL_REG_OFS (0x00000000)
- #define ICACHE_CONFIG_REG_OFS (0x00000008)
- #define DCACHE_CONFIG_REG_OFS (0x0000000C)
- #define CCR_FI (1<<21)
- #define CCR_FD (1<<22)
- #define POS_CCR_IP 15
- #define CCR_IP (1<<POS_CCR_IP)
- #define POS_CCR_DP 14
- #define CCR_DP (1<<POS_CCR_DP)
- #define CCR_DS (1<<23)
- #define CCR_DF (1<<5)
- #define CCR_IF (1<<4)
- #define MASK_CCR_DCS (3<<2)
- #define CCR_DCS_ENABLED (3<<2)
- #define CCR_DCS_FROZEN (1<<2)
- #define CCR_DCS_DISABLED (0<<2)
- #define MASK_CCR_ICS (3<<0)
- #define CCR_ICS_ENABLED (3<<0)
- #define CCR_ICS_FROZEN (1<<0)
- #define CCR_ICS_DISABLED (0<<0)
- #define CCR_IB (1<<16)
- #define __NOCACHE_ASI 0x01
- #define __ICACHE_TAGS_ASI 0x0C
- #define __ICACHE_DATA_ASI 0x0D
- #define __DCACHE_TAGS_ASI 0x0E
- #define __DCACHE_DATA_ASI 0x0F

- #define __ICACHE_FLUSH_ASI_DO_NOT_USE 0x10
- #define __DCACHE_FLUSH_ASI 0x11
- #define _ASM __asm__ __volatile__
- #define NOP _ASM("nop;":::"memory")

8.12.1 Detailed Description

Copyright

8.13 swcMemoryTransfer.h File Reference

Functions

- void swcU32memcpy (u32 ∗dst, u32 ∗src, u32 len)

  Function that copies from source to destination.
- void swcU32memsetU32 (u32 ∗addr, u32 lenB, u32 val)

  Function that sets memory with a givven value.

8.13.1 Detailed Description

Copyright

8.14 swcRandom.h File Reference

```
#include "swcRandomDefines.h"
```

Functions

- void swcRandInit (u64 initValue)

  Reset the base seed of the PRNG.
- u64 swcRandGetRandValue (void)

  Get next 64 bit random value in sequence defined by the global seed which was set using swcRandInit().
- u64 swcRandGetRandValue_r (u64 ∗seed)

  Get next 64 bit random value in sequence defined by seed.
- int swcRandBufferOp (tyRandOperation operation, void ∗targetAddress, u32 len, u64 seed)

8.14.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see-: common/license.txt

8.15 swcRandomDefines.h File Reference

Macros

- #define RAND_MAX ((u64)(-1))

Enumerations

- enum tyRandOperation { RAND_WRITE, RAND_VERIFY, RAND_WRITE_32, RAND_VERIFY_32 }

8.15.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see-: common/license.txt

8.16 swcShaveLoader.h File Reference

```
#include <stdarg.h>
#include "mv_types.h"
#include <DrvIcb.h>
#include <swcDmaTypes.h>
#include "DrvCommon.h"
#include "theDynContext.h"
#include "swcLeonUtils.h"
```

Macros

- #define ADDR_DDRL2(x) (((u32)(x)) & 0xF0FFFFFF)

  use DDR address through L2 cache. Force it's use.
- #define ACCEPT_ALTERNATIVE_SHAVE_START_METHOD FALSE
- #define SHAVE_INTERRUPT_LEVEL 3

Shave dummy wrappers

- #define SVU(x) x
- #define IRF(x) x
- #define SRF(x) x
- #define VRF(x) x

Typedefs

- typedef u32 swcShaveUnit_t

Enumerations

- enum context_t { SHVXDATA = 0, SHVZDATA, SHVDLIB }

Functions

- void swcSetAbsoluteDefaultStack (u32 shave_num)

  Set absolute default stack for a specific shave.
- void swcStateConsideredShaveStackSize (u32 shaveNumber, u32 size)

  Allows the user to assert a stack size against which checks may be implemented. This does not represent a guarantee that the system will allocate this stack it only allows users to specify how much space they themselves have considered and made available through other means for the application. Calling this function allows the system to perform checks which would detect if this size was overrun at any stage.
- u32 swcGetShaveStackSize (u32 shaveNumber)

  Reads back the stack size for a specified shave. When calling either swcSetAbsoluteDefaultStack or swcSetWindowedDefaultStack the stack size set to register i20 will be stored and can be read back with the help of this function.
- u32 swcGetUnusedShaveFreeStack (u32 shaveNumber, u32 canaryValue)

  If stack painter was used, this function searches for the size of unused stack given pattern checks N-OTE!: this function does nothing relevant if user did not call swcStateConsideredShaveStackSize and swcStackPainter before running a shave application.
- void swcStackPainter (u32 shaveNumber, u32 canaryValue)

  Paint stack with a specific canary value. NOTE: one must have called the swcStateConsideredShave-StackSize on the shaveNumber used here in advance of calling this function.
- void swcGetShaveWindowRegs (u32 shaveNumber, u32 ∗windows)

  Get Shave window register values.
- void swcSetShaveWindow (u32 shave_num, u32 window_num, u32 window_addr)

  Set a specific window register with a target window base address.
- void swcSetShaveWindows (u32 shaveNumber, u32 windowA, u32 windowB, u32 windowC, u32 windowD)

  Set each window register with the corresponding window base address.
- void swcSetShaveWindowsToDefault (u32 shaveNumber)

  Reset windows to default values in case they are rewritten by other shaves param[in] shaveNumber - shave number for which default value will be set.
- u32 swcShaveRunning (u32 svu)

  Check if a specific Shave is running or it is stopped.
- void swcRunShave (u32 shave_nr, u32 entry_point)

  Start shave shave_nr from entry_point.
- void swcStartShave (u32 shave_nr, u32 entry_point)

  Starts non blocking execution of a shave.
- void swcDynStartShave (u32 shave_nr, u32 Context)

  Starts non blocking execution of a shave using dynamic sub module alocator.
- void swcShaveStartAsync (u32 shave_nr, u32 entry_point, irq_handler function)

  Starts non blocking execution of a shave.

- void swcStartShaveAsync (u32 shave_nr, u32 entry_point, irq_handler function) __Deprecated_-_("Please use swcShaveStartAsync instead.")
- void swcDynShaveStartAsync (u32 shave_nr, u32 Context, irq_handler function)

  Starts dynamic non blocking execution of a shave. A master entry point is executed prior to jumping into shave entry point.
- void swcAssignShaveCallback (u32 shave_nr, irq_handler function)

  Assigns a callback to a shave for end of execution. Alternative way to the swcStartShaveAsync way of working.
- void swcSetRegsCC (u32 shave_num, const char ∗fmt, va_list a_list)
- void swcStartShaveCC (u32 shave_num, u32 pc, const char ∗fmt,...)

  Write the value to a IRF/SRF/VRF Registers from a specific Shave.
- void swcDisableShaveCallback (u32 shave_nr)

  Disables the interrupt for shave end. Useful for cases where the shave needs to be run for a few times in Async mode with interrupts but then the same shave needs to stop triggering interrupts.
- void swcStartShaveAsyncCC (u32 shave_num, u32 pc, irq_handler function, const char ∗fmt,...)

  Write the value to a IRF/SRF/VRF Registers from a specific Shave.
- void swcSetupShaveCC (u32 shave_num, const char ∗fmt,...)

  Write the value to a IRF/SRF/VRF Registers from a specific Shave.
- void swcSetRounding (u32 shave_no, u32 roundingBits)

  Function that starts one shave but at the same time also sets rounding bits.
- void swcResetShave (u32 shaveNumber)

  Reset shave.
- void swcResetShaveLite (u32 shaveNumber)

  Reset shave without resetting the fifo.
- int swcWaitShaves (u32 no_of_shaves, swcShaveUnit_t ∗shave_list)

  Function that waits for the shaves used to finish.
- int swcWaitShave (u32 shave_nr)

  Wait for a specific shave to finish execution.
- u32 swcShavesRunning (u32 first, u32 last)

  Check if a list of shaves is running or not.
- u32 swcShavesRunningArr (u32 arr[ ], u32 N)

  Check if a list of shaves stored in an array is running or not.
- u32 swcSolveShaveRelAddr (u32 vAddr, u32 shaveNumber)

  Translate windowed address into real physical address.
- void swcLoadMbin (u8 ∗sAddr, u32 targetS)

  Load a mbin file to a specific target address on shave.
- void swcSetWindowedDefaultStack (u32 shave_num)

  Sets a default value for stack.
- void swcLoadshvdlib (u8 ∗sAddr, u32 targetS)

  Dynamically load shvdlib file - These are elf object files stripped of any symbols.
- void swcLoadDynLibraryCacheAware (u8 ∗sAddr, u32 targetS, context_t contextType, u32 ∗vpProgrammedMemAddress, u32 ∗flushLength)

  Dynamically load library file and return start memory address and length that need to be flushed - These are elf object files stripped of any symbols.
- void swcLoadDynLibrary (u8 ∗sAddr, u32 targetS, context_t contextType)

  Dynamically load library file - These are elf object files stripped of any symbols.

- s32 swcRunShaveAlgo (DynamicContext_t ∗moduleStData, int ∗const shaveNumber)

    Sets up and launches one dynamic application instance. Uses the shaves preliminary assigned by user via function swcSetupDynShaveApps(). Allocates all necessary memory, loads the dynamic library, then starts the shave.

- s32 swcRunShaveAlgoCC (DynamicContext_t ∗moduleStData, int ∗const shaveNumber, const char ∗fmt,...)

    Sets up and launches one dynamic application instance. Uses the shaves preliminary assigned by user via function swcSetupDynShaveApps(). Allocates all necessary memory, loads the dynamic library, then starts the shave.

- s32 swcRunShaveAlgoOnAssignedShave (DynamicContext_t ∗moduleStData, u32 shaveNumber)

    Sets up and launches one dynamic application instance on a specifically requested SHAVE Uses the shaves preliminary assigned by user via function swcSetupDynShaveApps(). Allocates all necessary memory, loads the dynamic library, then starts the shave. Checks if the requested shave has bee configured in advance and if it is not running.

- s32 swcRunShaveAlgoOnAssignedShaveCC (DynamicContext_t ∗moduleStData, u32 shaveNumber, const char ∗fmt,...)

    Sets up and launches one dynamic application instance on a specifically requested SHAVE Uses the shaves preliminary assigned by user via function swcSetupDynShaveApps(). Allocates all necessary memory, loads the dynamic library, then starts the shave. Checks if the requested shave has bee configured in advance and if it is not running.

- s32 swcSetupDynShaveApps (DynamicContext_t ∗moduleStData, const swcShaveUnit_t ∗svuList, const uint32_t instances)

    This function allocates heap and group data memory for all configured instances of one application. It must be called prior to using swcRunShaveAlgo(). Can be used from both Leons. svuList below is not copied internally, instead just the pointer is assigned to an internal structure. Please ensure the svuList memory is alive until the call of swcCleanupDynShaveApps. Note: be careful about stack declared svuList.

- s32 swcCleanupDynShaveApps (DynamicContext_t ∗moduleStData)

    This function frees the heap and group data memory for all configured instances of one application. It can be called after usage of swcRunShaveAlgo(). Can be used from both Leons.

- s32 swcDynShaveAppSetWindows (DynamicContext_t ∗moduleStData, u32 cmxCriticalCodeSize)

    This function allows hinting how much code/data is desired to be allocated TODO: add functionality to precompute these sizes based on .textCrit size.

- u32 swcCheckFreeAndValidShave (DynamicContext_t ∗moduleStData, u32 shaveNumber)

    This function is used to check if the user has called a correct shave. We define "correct" as: configured to be used by the current dyncontext and not currently running.

- s32 swcRequestUnallocatedShaves (swcShaveUnit_t ∗svuList, u32 shavesNumber)

    This functions gives a list of unallocated shaves in the system.

- s32 swcGetUnallocatedShavesNumber (void)

    This function return the number of unallocated shave in the system.

- s32 swcCleanupDynShaveListApps (DynamicContext_t ∗mData, swcShaveUnit_t ∗svuList, uint32_t elementsNumber)

    This function frees the heap and group data memory for the specified instances of one application. Can be used from both Leons.

- void swcSetNewHeapLocation (DynamicContext_t ∗mData, unsigned char ∗newAddress, swcShaveUnit_t shaveNumber)

    This function set a new heap location for a specific shave. Can be used from both Leons.

- void swcSetNewAppDynDataLocation (DynamicContext_t ∗mData, unsigned char ∗newAddress, swcShaveUnit_t shaveNumber)

  This function set a new memory location where to load the application dynamic data. Can be used from both Leons.

- void swcSetGrpDynDataLocation (DynamicContext_t ∗mData, unsigned char ∗newAddress, swcShaveUnit_t shaveNumber)

  This function set a new memory location where to load the grup dynamic data. Can be used from both Leons.

- int swcIsoSetupShaveApplication (DynamicContext_t ∗moduleStData, swcShaveUnit_t ∗svuList, uint32_t shavesNumber, MISA_PARADIGM_TYPE paradigmType)

  This function allocates heap and group data memory for all configured instances of one application and loads the dynamic library. It must be called prior to using swcRunShaveAlgo(). Can be used from both Leons. Please ensure the svuList memory is alive until the call of swcCleanupDynShaveApps. Note: be careful about stack declared svuList.

- int swcStartEntryPointDC (DynamicContext_t ∗moduleStData, uint32_t shaveNumber, const char ∗functionName)

  This function launch a shave application with a specific function as entry point. Can be used from both Leons.

- int swcStartEntryPointDCCC (DynamicContext_t ∗moduleStData, uint32_t shaveNumber, const char ∗functionName, const char ∗fmt,...)

  This function launch a shave application with a specific function as entry point. Can be used from both Leons.

- int swcStartFC (DynamicContext_t ∗moduleStData, uint32_t shaveNumber)

  This function launch a shave application by calling the main function. Can be used from both Leons.

- int swcIsoCleanShaveApplication (DynamicContext_t ∗moduleStData, swcShaveUnit_t ∗svuList, uint32_t shavesNumber, MISA_PARADIGM_TYPE paradigmType)

  This function frees the heap and group data memory for all configured instances of one application. It can be called after usage of swcRunShaveAlgo(). Can be used from both Leons.

8.16.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see- : common/license.txt

8.17 swcSliceUtils.h File Reference

```
#include <swcDmaTypes.h>
#include <mv_types.h>
#include <DrvIcb.h>
```

Functions

- void swcSliceReleaseMutex (unsigned int mutexNo)

  Function that releases a certain hardware mutex.

- int swcSliceRequestMutex (unsigned int mutexNo, int requestOption)

Function that requests a certain hardware mutex.

- void swcSetMutexInterrupt (irq_handler mutexHandler, int intMask)

  Function that requests a certain hardware mutex.
- int swcSliceIsMutexFree (unsigned int mutexNo)

  Checks if a mutex is free.

8.17.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see-: common/license.txt

8.18 swcTestUtils.h File Reference

```
#include "swcTestUtilsDefines.h"
#include "mv_types.h"
```

Functions

- tyProcessorType swcGetProcessorType (void)

  This function recognizes the processor type the simulations are running on.
- void swcShaveProfInit (performanceStruct ∗perfStruct)

  Function that initializes the benchmark structure's elements.
- void swcShaveProfStartGathering (u32 shaveNumber, performanceStruct ∗perfStruct)

  Function that starts the counters for structure's members.
- int swcShaveProfGatheringDone (performanceStruct ∗perfStruct)

  Function that verifies if all the structure's parameters are updated with the values from the counters.
- void swcShaveProfStopGathering (u32 shaveNumber, performanceStruct ∗perfStruct)

  Function that reads the values from the counters.
- void swcShaveProfPrint (u32 shaveNumber, performanceStruct ∗perfStruct)

  Function that prints the values that were read from the counters.
- void swcShaveProfStartGatheringFields (u32 shaveNumber, performanceCounterDef perfDefines)

  Function that starts one counter at the time, finding information about one possible field only.
- void swcShaveProfStopFieldsGathering (u32 shaveNumber, performanceCounterDef perfDefines)

  Function that prints and reads values from counters.
- void swcShaveProfStopFieldsGatehering (u32 shaveNumber, performanceCounterDef perfDefines) __Deprecated__("Use swcShaveProfStopFieldsGathering instead")

  Function that prints and reads values from counters.
- void swcShaveProfileCyclesStart (u32 shaveNumber)

  Function that start gathering information about cycles, stalls and instructions.
- void swcShaveProfileCyclesStop (u32 shaveNumber)

  Function that prints and reads values from counters.

8.18.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see- : common/license.txt

8.19 swcTestUtilsDefines.h File Reference

```
#include "DrvTimer.h"
#include <mv_types.h>
```

Data Structures

- struct performanceStruct

Enumerations

- enum tyProcessorType {
  MVI_UNKNOWN, MVI_IC, MVI_VCS, MVI_FSIM,
  MVI_FPGA }
- enum performanceCounterDef {
  PERF_STALL_COUNT, PERF_INSTRUCT_COUNT, PERF_CLK_CYCLE_COUNT, PERF_-BRANCH_COUNT,
  PERF_TIMER_COUNT }

8.19.1 Detailed Description

Copyright

> All code copyright Movidius Ltd 2012, all rights reserved For License Warranty see- : common/license.txt

8.20 theDynContext.h File Reference

```
#include "DrvRegUtilsDefines.h"
#include "DrvSvuDefines.h"
#include "stdlib.h"
#include "sys/shave_system.h"
#include "sys/shave_exitcodes.h"
```

Data Structures

- struct DynamicContextInstances_elm
- struct DynamicContext_elm
- struct DynamicContextInfo_elm

- struct DynamicContextGlobal_elm

Macros

- #define TOKEN_PASTE_INTERN(APP) APP ## X_ModuleData
- #define MODULE_DATA_INTERN(APP) TOKEN_PASTE_INTERN(APP)
- #define TOKEN_PASTE(APP) APP ## X_ModuleData
- #define MODULE_DATA(APP) TOKEN_PASTE(APP)

Typedefs

- typedef u32 swcShaveUnit_t
- typedef u32 ParadigmSpecificEntry
- typedef struct DynamicContextInstances_elm ∗ DynamicContextInstancesPtr
- typedef struct DynamicContext_elm DynamicContext_t
- typedef struct DynamicContextInfo_elm DynamicContextInfo_t
- typedef struct DynamicContextGlobal_elm DynamicContextGlobal_t

Enumerations

- enum DYNCONTEXT_HEAP_ACTION_TYPE { DYNCONTEXT_HEAP_NOINIT = 0, DYNCONTEXT_HEAP_INIT = 1, DYNCONTEXT_HEAP_INVALID_VAL = 3 }
- enum DYNCONTEXT_APP_REENTRANT_TYPE { DYNCONTEXT_APP_NOT_RENTRANT = 0, DYNCONTEXT_APP_REENTRANT = 1 }
- enum MISA_PARADIGM_TYPE { MISA_DECOUPLED = 5001, MISA_FULLY_COUPLED }

Variables

- DynamicContextGlobal_t GlobalContextData

8.20.1 Detailed Description

Copyright

All code copyright Movidius Ltd 2016, all rights reserved For License Warranty see-: common/license.txt

8.20.2 Macro Definition Documentation

#define MODULE_DATA(APP) **TOKEN_PASTE**(APP)

#define MODULE_DATA_INTERN(APP) **TOKEN_PASTE_INTERN**(APP)

#define TOKEN_PASTE(APP) APP ## X_ModuleData

#define TOKEN_PASTE_INTERN(APP) APP ## X_ModuleData

8.20.3 Typedef Documentation

typedef struct **DynamicContext_elm DynamicContext_t**

typedef struct **DynamicContextGlobal_elm DynamicContextGlobal_t**

typedef struct **DynamicContextInfo_elm DynamicContextInfo_t**

typedef struct **DynamicContextInstances_elm*** **DynamicContextInstancesPtr**

typedef u32 **ParadigmSpecificEntry**

typedef u32 **swcShaveUnit_t**

8.20.4 Enumeration Type Documentation

enum **DYNCONTEXT_APP_REENTRANT_TYPE**

Enumerator

DYNCONTEXT_APP_NOT_RENTRANT
DYNCONTEXT_APP_REENTRANT

enum **DYNCONTEXT_HEAP_ACTION_TYPE**

Enumerator

DYNCONTEXT_HEAP_NOINIT
DYNCONTEXT_HEAP_INIT
DYNCONTEXT_HEAP_INVALID_VAL

enum **MISA_PARADIGM_TYPE**

Enumerator

MISA_DECOUPLED
MISA_FULLY_COUPLED

8.20.5 Variable Documentation

DynamicContextGlobal_t GlobalContextData

Index