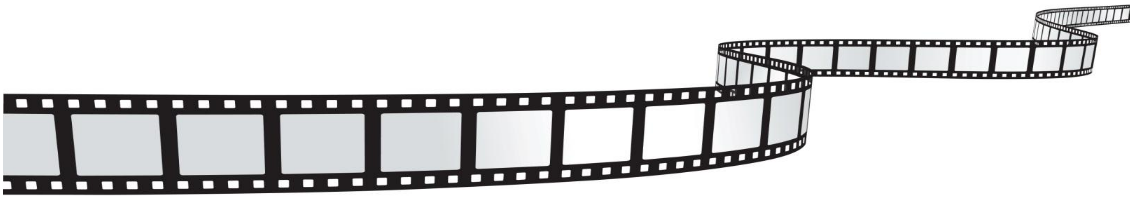
Shave Utility Functions

18.08.10



Contents

1 Introduction **3**

2 Module Index **4**

2.1 Modules 4

3 File Index **5**

3.1 File List 5

4 Module Documentation **6**

4.1 Common Shave API 6

4.1.1 Detailed Description 7

4.1.2 Macro Definition Documentation 7

4.1.3 Function Documentation 15

4.2 Shave LUT 16

4.2.1 Detailed Description 16

4.2.2 Function Documentation 16

4.3 CMXDMA API 19

4.3.1 Detailed Description 19

4.3.2 Function Documentation 19

4.4 SIMD Utilities 22

4.4.1 Detailed Description 25

4.4.2 Function Documentation 25

5 File Documentation **45**

5.1 MDKdox-ShaveUtils-intro.txt File Reference 45

5.2 svuCommonShave.h File Reference 45

5.2.1 Detailed Description 46

5.3 svuCommonShaveLUT.h File Reference 47

5.3.1 Detailed Description 47

5.4 swcCdma.h File Reference 47

5.4.1 Detailed Description 48
5.5 swcSIMDUtils.h File Reference 48
5.5.1 Detailed Description 51

Index **52**

Chapter 1

Introduction

This document describes the Shave Utilities provided with Myriad2.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

CMXDMA API 19

Common Shave API 6

SIMD Utilities 22

Shave LUT 16

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

svuCommonShave.h 45
svuCommonShaveLUT.h 47
swcCdma.h 47
swcSIMDUtils.h 48

Chapter 4

Module Documentation

4.1 Common Shave API

API manipulating shave functionalities.

Macros

- #define SHAVE_HALT

  Shave halt instruction.
- #define scMutexRequest ShDrvMutexRequest

  Mutex Request.
- #define scMutexRequestNoWorkaround ShDrvMutexRequest
- #define scMutexRelease ShDrvMutexRelease

  This will release mutex.
- #define scFifoWrite ShDrvCmxFifoWriteWord

  Write a 32-bit value to a Shave FIFO.
- #define scFifoWriteDword ShDrvCmxFifoWriteDWord

  Write a 64-bit value to a Shave FIFO.
- #define scFifoReadDword ShDrvCmxFifoReadDWord

  Read a 64-bit value from the FIFO of the current shave.
- #define scFifoReadShaveDword ShDrvCmxFifoNReadDWord

  Read a 64-bit value from the FIFO of the current shave.
- #define scFifoWriteDirectDword ShDrvCmxFifoWriteDirectDWord

  Write a 64-bit value to a Shave FIFO directly into the FIFO memory. This permits to alter the values already contained in FIFO without changing the FIFO pointers.
- #define scFifoReadDirectDword ShDrvCmxFifoReadDirectDWord

  Read a 64-bit value from a Shave FIFO directly from the FIFO memory.
- #define scFifoRead ShDrvCmxFifoReadWord

  Read a 32-bit value from the FIFO of the current shave.
- #define scFifoReadShave ShDrvCmxFifoNReadWord

  Read a 32-bit value from the FIFO of the shave given as parameter.
- #define scFifoMonitorSelect ShDrvCmxFifoMonitorSelect

  Configures the FIFO status bit to route through to the SHAVE for direct monitoring.

- #define scFifoWaitElement ShDrvCmxFifoMonitorWaitElement

  Wait for an element in the monitored FIFO.
- #define scFifoRAMControl ShDrvCmxFifoRAMControl

  Control RAM power modes. This permits to switch the FIFO RAM to different power states.
- #define scFifoGetAlmostFullLevel ShDrvCmxFifoGetAlmostFullLevel

  Get the value that was set for the "Almost full" FIFO fill level.
- #define scFifoSetAlmostFullLevel ShDrvCmxFifoSetAlmostFullLevel

  Set the 'Almost full' level for all the FIFOs.
- #define scFifoGetReadPtrValue ShDrvCmxFifoGetReadPtrValue

  Get the read pointer value for a specific Shave.
- #define scFifoGetWritePtrValue ShDrvCmxFifoGetWritePtrValue

  Get the write pointer value for a specific Shave.
- #define scFifoGetFillLevel ShDrvCmxFifoGetFillLevel

  Get the current fill level for a specific shave.
- #define scFifoIsFull ShDrvCmxFifoIsFull

  Check whether a FIFO is full or not.
- #define scFifoIsAlmostFull ShDrvCmxFifoIsAlmostFull

  Check whether a FIFO has reached the 'almost full' level of filling.
- #define scFifoIsEmpty ShDrvCmxFifoIsEmpty

  Check whether a FIFO is empty.
- #define scFifoWriteDirectWord ShDrvCmxFifoWriteDirectWord

  Write a 32-bit value to a Shave FIFO directly into the FIFO memory.
- #define scFifoReadDirectWord ShDrvCmxFifoReadDirectWord

  Read a 32-bit value directly from the FIFO memory without affecting the FIFO pointers.
- #define scFifoReadShaveDwordAtomic ShDrvCmxFifoNReadDWordAtomic

  Atomic read a 64-bit value from the CMX FIFO of the current shave.

Functions

- __asm (".nowarn 32\n"".include svuCommonDefinitions.incl\n"".nowarnend\n")

4.1.1 Detailed Description

API manipulating shave functionalities. Allows shaves manipulating mutexes and other features. For a more detailed understanding please reread the relevant sections in the n the MDKMyriad2Programmer_Guide document.

4.1.2 Macro Definition Documentation

#define scFifoGetAlmostFullLevel ShDrvCmxFifoGetAlmostFullLevel

Get the value that was set for the "Almost full" FIFO fill level.

Returns

the 'Almost full' level which is currently set for all the FIFOs

#define scFifoGetFillLevel ShDrvCmxFifoGetFillLevel

Get the current fill level for a specific shave.

Parameters

`in`	*shaveNr*	- The Shave for which the fill level should be read

Returns

The number of elements currently available in FIFO

#define scFifoGetReadPtrValue ShDrvCmxFifoGetReadPtrValue

Get the read pointer value for a specific Shave.

Parameters

`in`	*shaveNr*	- The Shave for which the pointer value should be read

Returns

The read pointer value.

#define scFifoGetWritePtrValue ShDrvCmxFifoGetWritePtrValue

Get the write pointer value for a specific Shave.

Parameters

`in`	*shaveNr*	- The Shave for which the pointer value should be read

Returns

The write pointer value.

#define scFifoIsAlmostFull ShDrvCmxFifoIsAlmostFull

Check whether a FIFO has reached the 'almost full' level of filling.

Parameters

`in`	*shaveNr*	- The Shave for which the FIFO should be checked

Returns

- 0 - The FIFO doesn't have enough elements in the FIFO to trigger the 'Almost full' bit.
- 1 - The FIFO has triggered the 'Almost full' level.

#define scFifoIsEmpty ShDrvCmxFifoIsEmpty

Check whether a FIFO is empty.

Parameters

in	*shaveNr*	- The Shave for which the FIFO should be checked

Returns

- 0 - The FIFO is not empty(i.e. there is at least one element in FIFO)
- 1 - The FIFO is empty

#define scFifoIsFull ShDrvCmxFifoIsFull

Check whether a FIFO is full or not.

Parameters

in	*shaveNr*	- The Shave for which the FIFO should be checked

Returns

- 0 - The FIFO is full. All the following writes won't have any effect if the values that already reside there wouldn't be read.
- 1 - The FIFO is not full. This doesn't mean that the FIFO is empty.

#define scFifoMonitorSelect ShDrvCmxFifoMonitorSelect

Configures the FIFO status bit to route through to the SHAVE for direct monitoring.

Parameters

in	*shaveNr*	- The Shave for which to enable the direct monitoring.
in	*val*	- Configuration value.

#define scFifoRAMControl ShDrvCmxFifoRAMControl

Control RAM power modes. This permits to switch the FIFO RAM to different power states.

This can be achieved by passing an 8-bit value, where the most significant three bits trigger one of the three power states.

- bit 5 - When set, this bit shuts down power to periphery and memory core, no memory data retention.
- bit 6 - When set, this bit triggers the 'Light Sleep' mode: the memory goes into low leakage mode, there is no change in the output state.
- bit 7 - When set, this bit triggers the 'Deep Sleep' mode: it shuts down power to periphery and maintains memory contents. The outputs of the memory are pulled low.

Parameters

in	*value*	- The input value

#define scFifoRead ShDrvCmxFifoReadWord

Read a 32-bit value from the FIFO of the current shave.

Returns

The 32-bit value read from FIFO

#define scFifoReadDirectDword ShDrvCmxFifoReadDirectDWord

Read a 64-bit value from a Shave FIFO directly from the FIFO memory.

This permits to read the values from FIFO without changing the FIFO pointers.

Parameters

in		- The shave number which FIFO should be read
in		- The entry index from where the value should be read. There is a total of 16 x 64-bit entries for EACH shave, so this number shouldn't be bigger than 15.

Returns

The 64-bit value read directly from FIFO.

#define scFifoReadDirectWord ShDrvCmxFifoReadDirectWord

Read a 32-bit value directly from the FIFO memory without affecting the FIFO pointers.

Parameters

in	*shaveNr*	- The Shave for which the FIFO should be read.
in	*index*	- The entry index from where the value should be read. There is a total of 16 x 64-bit entries for EACH shave, so this number shouldn't be bigger than 15.

Returns

The 32-bit value read directly from FIFO

#define scFifoReadDword ShDrvCmxFifoReadDWord

Read a 64-bit value from the FIFO of the current shave.

Returns

The 64-bit value read from FIFO

#define scFifoReadShave ShDrvCmxFifoNReadWord

Read a 32-bit value from the FIFO of the shave given as parameter.

Parameters

in	*shaveNr*	- the shave number which FIFO should be read

Returns

The 32-bit value read from FIFO

#define scFifoReadShaveDword ShDrvCmxFifoNReadDWord

Read a 64-bit value from the FIFO of the current shave.

Parameters

in	*shaveNr*	- the shave number which FIFO should be read

Returns

The 64-bit value read from FIFO.

#define scFifoReadShaveDwordAtomic ShDrvCmxFifoNReadDWordAtomic

Atomic read a 64-bit value from the CMX FIFO of the current shave.

Parameters

in	*shaveNr*	- the shave number which FIFO should be read

Returns

The 64-bit value read from FIFO. The upper byte of the return value will be set to 0x00 in case of success, or to 0xFF if the FIFO was empty

#define scFifoSetAlmostFullLevel ShDrvCmxFifoSetAlmostFullLevel

Set the ’Almost full’ level for all the FIFOs.

Parameters

in	*level*	- The number of elements which can be written in FIFO until the ’Almost full’ bit is triggered.

#define scFifoWaitElement ShDrvCmxFifoMonitorWaitElement

Wait for an element in the monitored FIFO.

Returns

#define scFifoWrite ShDrvCmxFifoWriteWord

Write a 32-bit value to a Shave FIFO.

Parameters

in	*shaveNr*	- the shave number which FIFO should be written
in	*val*	- The u32 value that should be written to FIFO

#define scFifoWriteDirectDword ShDrvCmxFifoWriteDirectDWord

Write a 64-bit value to a Shave FIFO directly into the FIFO memory. This permits to alter the values already contained in FIFO without changing the FIFO pointers.

Parameters

in	*shaveNr*	- The shave number which FIFO should be written
in	*index*	- The entry index where the value should be written to. There is a total of 16 x 64-bit entries for EACH shave, so this number shouldn't be bigger than 15.
in	*val*	- The u64 value that should be written to FIFO

Returns

void

#define scFifoWriteDirectWord ShDrvCmxFifoWriteDirectWord

Write a 32-bit value to a Shave FIFO directly into the FIFO memory.

This permits to alter the values already contained in FIFO without changing the FIFO pointers.

Parameters

in	*shaveNr*	- The shave number which FIFO should be written
in	*index*	- The entry index where the value should be written to. There is a total of 16 x 64-bit entries for EACH shave, so this number shouldn't be bigger than 15.
in	*val*	- The u32 value that should be written to FIFO

#define scFifoWriteDword ShDrvCmxFifoWriteDWord

Write a 64-bit value to a Shave FIFO.

Parameters

in	*shaveNr*	- the shave number which FIFO should be written
in	*val*	- The u64 value that should be written to FIFO

Returns

void

#define scMutexRelease ShDrvMutexRelease

This will release mutex.

Note

For a detailed explanation, please see the "Mutexes" section in the MDKMyriad2Programmer_-Guide

Parameters

in	*mutex_num*	- mutex number that will be released

Returns

void

#define scMutexRequest ShDrvMutexRequest

Mutex Request.

Note

For a detailed explanation, please see the "Mutexes" section in the MDKMyriad2Programmer_-Guide

Parameters

in	*mutex_num*	- mutex number requested

Returns

void

#define scMutexRequestNoWorkaround ShDrvMutexRequest

#define SHAVE_HALT

Value:

```
{ __asm volatile ( \
    "// 'SHAVE_HALT' defined in svuCommonShave.h used in " __FILE__ "\n\t" \
    "NOP"                "\n\t" \
    "BRU.swih 0x001F"    "\n\t" \
    "NOP 6"              "\n\t" \
    ::: "memory"); __builtin_unreachable (); }
```

Shave halt instruction.

4.1.3 Function Documentation

__asm (".nowarn 32\n"".include svuCommonDefinitions.incl\n"".nowarnend\n")

4.2 Shave LUT

Shave Look up table module functions API.

Functions

- ushort8 svuGet16BitVals16BitLUT (ushort8 in_values, u16 *lut_memory)

  Function that reads 8 ushort values into a ushort8 vector from LUT.
- uchar8 svuGet8BitVals16BitLUT (ushort8 in_values, u16 *lut_memory)

  Function that reads 8 uchar values into a ushort8 vector from LUT.
- uchar8 svuGet8BitVals8BitLUT (uchar8 in_values, u8 *lut_memory)

  Function that reads 8 uchar values into a uchar8 vector from LUT.
- uchar16 svuGet16_8BitVals8BitLUT (uchar16 in_values, u8 *lut_memory)

  Function that reads 16 uchar values into a uchar16 vector from LUT.
- ushort8 svuGetu16BitVals16BitLUT (ushort8 in_values, u16 *lut_memory)

  Function that reads 8 ushort values into a ushort8 vector from LUT.

4.2.1 Detailed Description

Shave Look up table module functions API. Used for inserting values into vectors from LUT.

4.2.2 Function Documentation

uchar16 svuGet16_8BitVals8BitLUT (uchar16 in_values, u8 * lut_memory)

Function that reads 16 uchar values into a uchar16 vector from LUT.

Parameters

in	*in_values*	- vector type to read input for performing LUT
in	*lut_memory*	- pointer to the LUT memory

Returns

vectorized LUT seek results

ushort8 svuGet16BitVals16BitLUT (ushort8 in_values, u16 * lut_memory)

Function that reads 8 ushort values into a ushort8 vector from LUT.

Parameters

in	*in_values*	- vector type to read input for performing LUT
in	*lut_memory*	- pointer to the LUT memory

Returns

vectorized LUT seek results

uchar8 svuGet8BitVals16BitLUT (ushort8 in_values, u16 * lut_memory)

Function that reads 8 uchar values into a ushort8 vector from LUT.

Parameters

in	*in_values*	- vector type to read input for performing LUT
in	*lut_memory*	- pointer to the LUT memory

Returns

vectorized LUT seek results

uchar8 svuGet8BitVals8BitLUT (uchar8 in_values, u8 * lut_memory)

Function that reads 8 uchar values into a uchar8 vector from LUT.

Parameters

in	*in_values*	- vector type to read input for performing LUT
in	*lut_memory*	- pointer to the LUT memory

Returns

vectorized LUT seek results

ushort8 svuGetu16BitVals16BitLUT (ushort8 in_values, u16 * lut_memory)

Function that reads 8 ushort values into a ushort8 vector from LUT.

Parameters

in	*in_values*	- vector type to read input for performing LUT
in	*lut_memory*	- pointer to the LUT memory

Returns

vectorized LUT seek results

4.3 CMXDMA API

CMXDMA driver for Shave processors.

Functions

- dmaRequesterId dmaInitRequesterWithAgent (int priority, int agentToAssign)

  Initialize a requester ID which will be used to properly initialize and distinguish single tasks or groups of tasks.
- void dmaSetUsedAgents (u8 nrOfUsedAgents, u8 startingFrom)

  Set up the number of link agents the driver will use in order to start new tasks.
- u32 dmaSolveRelAddr (u32 inAddr, u32 shaveNumber)

  Translate windowed address into real physical address. Non-windowed address are passed through.
- dmaTransactionList ∗ dmaCreateTransactionExt (u32 Type, dmaRequesterId ReqId, dmaTransactionList ∗NewTransaction, u8 ∗Src, u8 ∗Dst, u32 ByteLength, u32 SrcLineWidth, u32 DstLineWidth, s32 SrcStride, s32 DstStride, u8 BurstLength)

  CMXDMA task structure initialization extension, allowing the user to set a custom burst length.

4.3.1 Detailed Description

CMXDMA driver for Shave processors. This driver lets you perform fast data transfers using CMXDMA hardware

4.3.2 Function Documentation

dmaTransactionList∗ dmaCreateTransactionExt (u32 Type, dmaRequesterId ReqId, dmaTransactionList ∗ NewTransaction, u8 ∗ Src, u8 ∗ Dst, u32 ByteLength, u32 SrcLineWidth, u32 DstLineWidth, s32 SrcStride, s32 DstStride, u8 BurstLength)

CMXDMA task structure initialization extension, allowing the user to set a custom burst length.

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

Parameters

in	*Type*	Transaction type
in	*ReqId*	A requester ID returned by function #dmaInitRequester used to set the task priority and the task ID
in	*NewTransaction*	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
`in`	*BurstLength*	Number of transactions in a burst (1 - 16)

Returns

Pointer to initialized CMXDMA structure

dmaRequesterId dmaInitRequesterWithAgent (int priority, int agentToAssign)

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
`in`	*agentToAssign*	- The link agent to be used by CMXDMA while performing transfers initiated with generated requester ID.

Returns

a new requester ID

void dmaSetUsedAgents (u8 nrOfUsedAgents, u8 startingFrom)

Set up the number of link agents the driver will use in order to start new tasks.

If this function is not called, the default configuration will be to use all 4 link agents. If an invalid configuration will be provided(e.g too many link agents to use), the configuration will be rounded to the first appropriate.

Parameters

`in`	*nrOfUsedAgents*	- How many agents to use
`in`	*startingFrom*	- the first agent which will be used in the current configuration

Returns

void

u32 dmaSolveRelAddr (u32 inAddr, u32 shaveNumber)

Translate windowed address into real physical address. Non-windowed address are passed through.

Parameters

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

Returns

Resolved address

4.4 SIMD Utilities

SIMD utility functions API.

Functions

- float4 swcSIMDAbs4F32 (float4 in_vec)

  This will compute the absolute value of vector.
- float2 swcSIMDAbs2F32 (float2 in_vec)

  This will compute the absolute value of vector.
- half8 swcSIMDAbs8F16 (half8 in_vec)

  This will compute the absolute value of vector.
- half4 swcSIMDAbs4F16 (half4 in_vec)

  This will compute the absolute value of vector.
- half2 swcSIMDAbs2F16 (half2 in_vec)

  This will compute the absolute value of vector.
- int4 swcSIMDAbs4I32 (int4 in_vec)

  This will compute the absolute value of vector.
- int3 swcSIMDAbs3I32 (int3 in_vec)

  This will compute the absolute value of vector.
- int2 swcSIMDAbs2I32 (int2 in_vec)

  This will compute the absolute value of vector.
- short8 swcSIMDAbs8I16 (short8 in_vec)

  This will compute the absolute value of vector.
- short4 swcSIMDAbs4I16 (short4 in_vec)

  This will compute the absolute value of vector.
- short2 swcSIMDAbs2I16 (short2 in_vec)

  This will compute the absolute value of vector.
- char16 swcSIMDAbs16I8 (char16 in_vec)

  This will compute the absolute value of vector.
- char8 swcSIMDAbs8I8 (char8 in_vec)

  This will compute the absolute value of vector.
- char4 swcSIMDAbs4I8 (char4 in_vec)

  This will compute the absolute value of vector.
- char2 swcSIMDAbs2I8 (char2 in_vec)

  This will compute the absolute value of vector.
- float4 swcSIMDMin4F32 (float4 in_vec1, float4 in_vec2)

  This will compute the minumum value of vector.
- float2 swcSIMDMin2F32 (float2 in_vec1, float2 in_vec2)

  This will compute the minumum value of vector.
- int4 swcSIMDMin4I32 (int4 in_vec1, int4 in_vec2)

  This will compute the minumum value of vector.
- int3 swcSIMDMin3I32 (int3 in_vec1, int3 in_vec2)

  This will compute the minumum value of vector.
- int2 swcSIMDMin2I32 (int2 in_vec1, int2 in_vec2)

This will compute the minumum value of vector.

- uint4 swcSIMDMin4U32 (uint4 in_vec1, uint4 in_vec2)

  This will compute the minumum value of vector.
- uint3 swcSIMDMin3U32 (uint3 in_vec1, uint3 in_vec2)

  This will compute the minumum value of vector.
- uint2 swcSIMDMin2U32 (uint2 in_vec1, uint2 in_vec2)

  This will compute the minumum value of vector.
- half8 swcSIMDMin8F16 (half8 in_vec1, half8 in_vec2)

  This will compute the minumum value of vector.
- half4 swcSIMDMin4F16 (half4 in_vec1, half4 in_vec2)

  This will compute the minumum value of vector.
- half2 swcSIMDMin2F16 (half2 in_vec1, half2 in_vec2)

  This will compute the minumum value of vector.
- short8 swcSIMDMin8I16 (short8 in_vec1, short8 in_vec2)

  This will compute the minumum value of vector.
- short4 swcSIMDMin4I16 (short4 in_vec1, short4 in_vec2)

  This will compute the minumum value of vector.
- short2 swcSIMDMin2I16 (short2 in_vec1, short2 in_vec2)

  This will compute the minumum value of vector.
- ushort8 swcSIMDMin8U16 (ushort8 in_vec1, ushort8 in_vec2)

  This will compute the minumum value of vector.
- ushort4 swcSIMDMin4U16 (ushort4 in_vec1, ushort4 in_vec2)

  This will compute the minumum value of vector.
- ushort2 swcSIMDMin2U16 (ushort2 in_vec1, ushort2 in_vec2)

  This will compute the minumum value of vector.
- char16 swcSIMDMin16I8 (char16 in_vec1, char16 in_vec2)

  This will compute the minumum value of vector.
- char8 swcSIMDMin8I8 (char8 in_vec1, char8 in_vec2)

  This will compute the minumum value of vector.
- char4 swcSIMDMin4I8 (char4 in_vec1, char4 in_vec2)

  This will compute the minumum value of vector.
- char2 swcSIMDMin2I8 (char2 in_vec1, char2 in_vec2)

  This will compute the minumum value of vector.
- uchar16 swcSIMDMin16U8 (uchar16 in_vec1, uchar16 in_vec2)

  This will compute the minumum value of vector.
- uchar8 swcSIMDMin8U8 (uchar8 in_vec1, uchar8 in_vec2)

  This will compute the minumum value of vector.
- uchar4 swcSIMDMin4U8 (uchar4 in_vec1, uchar4 in_vec2)

  This will compute the minumum value of vector.
- uchar2 swcSIMDMin2U8 (uchar2 in_vec1, uchar2 in_vec2)

  This will compute the minumum value of vector.
- float4 swcSIMDMax4F32 (float4 in_vec1, float4 in_vec2)

  This will compute the maximum value of vector.
- float2 swcSIMDMax2F32 (float2 in_vec1, float2 in_vec2)

  This will compute the maximum value of vector.

- int4 swcSIMDMax4I32 (int4 in_vec1, int4 in_vec2)

  This will compute the maximum value of vector.
- int3 swcSIMDMax3I32 (int3 in_vec1, int3 in_vec2)

  This will compute the maximum value of vector.
- int2 swcSIMDMax2I32 (int2 in_vec1, int2 in_vec2)

  This will compute the maximum value of vector.
- uint4 swcSIMDMax4U32 (uint4 in_vec1, uint4 in_vec2)

  This will compute the maximum value of vector.
- uint3 swcSIMDMax3U32 (uint3 in_vec1, uint3 in_vec2)

  This will compute the maximum value of vector.
- uint2 swcSIMDMax2U32 (uint2 in_vec1, uint2 in_vec2)

  This will compute the maximum value of vector.
- half8 swcSIMDMax8F16 (half8 in_vec1, half8 in_vec2)

  This will compute the maximum value of vector.
- half4 swcSIMDMax4F16 (half4 in_vec1, half4 in_vec2)

  This will compute the maximum value of vector.
- half2 swcSIMDMax2F16 (half2 in_vec1, half2 in_vec2)

  This will compute the maximum value of vector.
- short8 swcSIMDMax8I16 (short8 in_vec1, short8 in_vec2)

  This will compute the maximum value of vector.
- short4 swcSIMDMax4I16 (short4 in_vec1, short4 in_vec2)

  This will compute the maximum value of vector.
- short2 swcSIMDMax2I16 (short2 in_vec1, short2 in_vec2)

  This will compute the maximum value of vector.
- ushort8 swcSIMDMax8U16 (ushort8 in_vec1, ushort8 in_vec2)

  This will compute the maximum value of vector.
- ushort4 swcSIMDMax4U16 (ushort4 in_vec1, ushort4 in_vec2)

  This will compute the maximum value of vector.
- ushort2 swcSIMDMax2U16 (ushort2 in_vec1, ushort2 in_vec2)

  This will compute the maximum value of vector.
- char16 swcSIMDMax16I8 (char16 in_vec1, char16 in_vec2)

  This will compute the maximum value of vector.
- char8 swcSIMDMax8I8 (char8 in_vec1, char8 in_vec2)

  This will compute the maximum value of vector.
- char4 swcSIMDMax4I8 (char4 in_vec1, char4 in_vec2)

  This will compute the maximum value of vector.
- char2 swcSIMDMax2I8 (char2 in_vec1, char2 in_vec2)

  This will compute the maximum value of vector.
- uchar16 swcSIMDMax16U8 (uchar16 in_vec1, uchar16 in_vec2)

  This will compute the maximum value of vector.
- uchar8 swcSIMDMax8U8 (uchar8 in_vec1, uchar8 in_vec2)

  This will compute the maximum value of vector.
- uchar4 swcSIMDMax4U8 (uchar4 in_vec1, uchar4 in_vec2)

  This will compute the maximum value of vector.
- uchar2 swcSIMDMax2U8 (uchar2 in_vec1, uchar2 in_vec2)

This will compute the maximum value of vector.

- float swcSIMDSum4F32 (float4 in_vec)

  Computes the Horizontal vector sum to scalar.
- float swcSIMDSumAbs4F32 (float4 in_vec)

  Computes the Horizontal vector sum to scalar.
- half swcSIMDSum8F16 (half8 in_vec)

  Computes the Horizontal vector sum to scalar.
- half swcSIMDSumAbs8F16 (half8 in_vec)

  Computes the Horizontal vector sum to scalar.
- u32 swcSIMDSum4U32 (uint4 in_vec)

  Computes the Horizontal vector sum to scalar.
- s32 swcSIMDSum4I32 (int4 in_vec)

  Computes the Horizontal vector sum to scalar.
- u32 swcSIMDSum8U16 (ushort8 in_vec)

  Computes the Horizontal vector sum to scalar.
- s32 swcSIMDSum8I16 (short8 in_vec)

  Computes the Horizontal vector sum to scalar.
- u32 swcSIMDSum16U8 (uchar16 in_vec)

  Computes the Horizontal vector sum to scalar.
- s32 swcSIMDSum16I8 (char16 in_vec)

  Computes the Horizontal vector sum to scalar.

4.4.1 Detailed Description

SIMD utility functions API. Single Instruction Multiple Data utilities that operate with VRFs

4.4.2 Function Documentation

char16 swcSIMDAbs16I8 (char16 in_vec)

This will compute the absolute value of vector.

Parameters

`in`	*in_vec*	- char16 vector received as input

Returns

char16 vector

half2 swcSIMDAbs2F16 (half2 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- half2 vector received as input

Returns

half2 vector

float2 swcSIMDAbs2F32 (float2 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- float2 vector received as input

Returns

float2 vector

short2 swcSIMDAbs2I16 (short2 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- short2 vector received as input

Returns

short2 vector

int2 swcSIMDAbs2I32 (int2 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- int2 vector received as input

Returns

int2 vector

char2 swcSIMDAbs2I8 (char2 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- char2 vector received as input

Returns

char2 vector

int3 swcSIMDAbs3I32 (int3 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- int3 vector received as input

Returns

int3 vector

half4 swcSIMDAbs4F16 (half4 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- half4 vector received as input

Returns

half4 vector

float4 swcSIMDAbs4F32 (float4 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- float4 vector received as input

Returns

float4 vector

short4 swcSIMDAbs4I16 (short4 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- short4 vector received as input

Returns

short4 vector

int4 swcSIMDAbs4I32 (int4 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- int4 vector received as input

Returns

int4 vector

char4 swcSIMDAbs4I8 (char4 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- char4 vector received as input

Returns

char4 vector

half8 swcSIMDAbs8F16 (half8 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- half8 vector received as input

Returns

half8 vector

short8 swcSIMDAbs8I16 (short8 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- short8 vector received as input

Returns

short8 vector

char8 swcSIMDAbs8I8 (char8 in_vec)

This will compute the absolute value of vector.

Parameters

in	*in_vec*	- char8 vector received as input

Returns

char8 vector

char16 swcSIMDMax16I8 (char16 in_vec1, char16 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- char16 vector received as input
in	*in_vec2*	- char16 vector received as input

Returns

char16 vector

uchar16 swcSIMDMax16U8 (uchar16 in_vec1, uchar16 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- uchar16 vector received as input
in	*in_vec2*	- uchar16 vector received as input

Returns

uchar16 vector

half2 swcSIMDMax2F16 (half2 in_vec1, half2 in_vec2)

This will compute the maximum value of vector.

Parameters

`in`	*in_vec1*	- half2 vector received as input
`in`	*in_vec2*	- half2 vector received as input

Returns

half2 vector

float2 swcSIMDMax2F32 (float2 in_vec1, float2 in_vec2)

This will compute the maximum value of vector.

Parameters

`in`	*in_vec1*	- float2 vector received as input
`in`	*in_vec2*	- float2 vector received as input

Returns

float2 vector

short2 swcSIMDMax2I16 (short2 in_vec1, short2 in_vec2)

This will compute the maximum value of vector.

Parameters

`in`	*in_vec1*	- short2 vector received as input
`in`	*in_vec2*	- short2 vector received as input

Returns

short2 vector

int2 swcSIMDMax2I32 (int2 in_vec1, int2 in_vec2)

This will compute the maximum value of vector.

Parameters

`in`	*in_vec1*	- int2 vector received as input
`in`	*in_vec2*	- int2 vector received as input

Returns

int2 vector

char2 swcSIMDMax2I8 (char2 in_vec1, char2 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- char2 vector received as input
in	*in_vec2*	- char2 vector received as input

Returns

char2 vector

ushort2 swcSIMDMax2U16 (ushort2 in_vec1, ushort2 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- ushort2 vector received as input
in	*in_vec2*	- ushort2 vector received as input

Returns

ushort2 vector

uint2 swcSIMDMax2U32 (uint2 in_vec1, uint2 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- uint2 vector received as input
in	*in_vec2*	- uint2 vector received as input

Returns

uint2 vector

uchar2 swcSIMDMax2U8 (uchar2 in_vec1, uchar2 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- uchar2 vector received as input
in	*in_vec2*	- uchar2 vector received as input

Returns

uchar2 vector

int3 swcSIMDMax3I32 (int3 in_vec1, int3 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- int3 vector received as input
in	*in_vec2*	- int3 vector received as input

Returns

int3 vector

uint3 swcSIMDMax3U32 (uint3 in_vec1, uint3 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- uint3 vector received as input
in	*in_vec2*	- uint3 vector received as input

Returns

uint3 vector

half4 swcSIMDMax4F16 (half4 in_vec1, half4 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- half4 vector received as input
in	*in_vec2*	- half4 vector received as input

Returns

half4 vector

float4 swcSIMDMax4F32 (float4 in_vec1, float4 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- float4 vector received as input
in	*in_vec2*	- float4 vector received as input

Returns

float4 vector

short4 swcSIMDMax4I16 (short4 in_vec1, short4 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- short4 vector received as input
in	*in_vec2*	- short4 vector received as input

Returns

short4 vector

int4 swcSIMDMax4I32 (int4 in_vec1, int4 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- int4 vector received as input
in	*in_vec2*	- int4 vector received as input

Returns

int4 vector

char4 swcSIMDMax4I8 (char4 in_vec1, char4 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- char4 vector received as input
in	*in_vec2*	- char4 vector received as input

Returns

char4 vector

ushort4 swcSIMDMax4U16 (ushort4 in_vec1, ushort4 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- ushort4 vector received as input
in	*in_vec2*	- ushort4 vector received as input

Returns

ushort8 vector

uint4 swcSIMDMax4U32 (uint4 in_vec1, uint4 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- uint4 vector received as input
in	*in_vec2*	- uint4 vector received as input

Returns

uint4 vector

uchar4 swcSIMDMax4U8 (uchar4 in_vec1, uchar4 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- uchar4 vector received as input
in	*in_vec2*	- uchar4 vector received as input

Returns

uchar4 vector

half8 swcSIMDMax8F16 (half8 in_vec1, half8 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- half8 vector received as input
in	*in_vec2*	- half8 vector received as input

Returns

half8 vector

short8 swcSIMDMax8I16 (short8 in_vec1, short8 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- short8 vector received as input
in	*in_vec2*	- short8 vector received as input

Returns

short8 vector

char8 swcSIMDMax8I8 (char8 in_vec1, char8 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- char8 vector received as input
in	*in_vec2*	- char8 vector received as input

Returns

char8 vector

ushort8 swcSIMDMax8U16 (ushort8 in_vec1, ushort8 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- ushort8 vector received as input
in	*in_vec2*	- ushort8 vector received as input

Returns

ushort8 vector

uchar8 swcSIMDMax8U8 (uchar8 in_vec1, uchar8 in_vec2)

This will compute the maximum value of vector.

Parameters

in	*in_vec1*	- uchar8 vector received as input
in	*in_vec2*	- uchar8 vector received as input

Returns

uchar8 vector

char16 swcSIMDMin16I8 (char16 in_vec1, char16 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- char16 vector received as input
in	*in_vec2*	- char16 vector received as input

Returns

char16 vector

uchar16 swcSIMDMin16U8 (uchar16 in_vec1, uchar16 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- uchar16 vector received as input
in	*in_vec2*	- uchar16 vector received as input

Returns

uchar16 vector

half2 swcSIMDMin2F16 (half2 in_vec1, half2 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- half2 vector received as input
in	*in_vec2*	- half2 vector received as input

Returns

half2 vector

float2 swcSIMDMin2F32 (float2 in_vec1, float2 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- float2 vector received as input
in	*in_vec2*	- float2 vector received as input

Returns

float2 vector

short2 swcSIMDMin2I16 (short2 in_vec1, short2 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- short2 vector received as input
in	*in_vec2*	- short2 vector received as input

Returns

short2 vector

int2 swcSIMDMin2I32 (int2 in_vec1, int2 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- int2 vector received as input
in	*in_vec2*	- int2 vector received as input

Returns

int2 vector

char2 swcSIMDMin2I8 (char2 in_vec1, char2 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- char2 vector received as input
in	*in_vec2*	- char2 vector received as input

Returns

char2 vector

ushort2 swcSIMDMin2U16 (ushort2 in_vec1, ushort2 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- ushort2 vector received as input
in	*in_vec2*	- ushort2 vector received as input

Returns

ushort2 vector

uint2 swcSIMDMin2U32 (uint2 in_vec1, uint2 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- uint2 vector received as input
in	*in_vec2*	- uint2 vector received as input

Returns

uint2 vector

uchar2 swcSIMDMin2U8 (uchar2 in_vec1, uchar2 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- uchar2 vector received as input
in	*in_vec2*	- uchar2 vector received as input

Returns

uchar2 vector

int3 swcSIMDMin3I32 (int3 in_vec1, int3 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- int3 vector received as input
in	*in_vec2*	- int3 vector received as input

Returns

int3 vector

uint3 swcSIMDMin3U32 (uint3 in_vec1, uint3 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- uint3 vector received as input
in	*in_vec2*	- uint3 vector received as input

Returns

uint3 vector

half4 swcSIMDMin4F16 (half4 in_vec1, half4 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- half4 vector received as input
in	*in_vec2*	- half4 vector received as input

Returns

half4 vector

float4 swcSIMDMin4F32 (float4 in_vec1, float4 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- float4 vector received as input
in	*in_vec2*	- float4 vector received as input

Returns

float4 vector

short4 swcSIMDMin4I16 (short4 in_vec1, short4 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- short4 vector received as input
in	*in_vec2*	- short4 vector received as input

Returns

short4 vector

int4 swcSIMDMin4I32 (int4 in_vec1, int4 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- int4 vector received as input
in	*in_vec2*	- int4 vector received as input

Returns

int4 vector

char4 swcSIMDMin4I8 (char4 in_vec1, char4 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- char4 vector received as input
in	*in_vec2*	- char4 vector received as input

Returns

char4 vector

ushort4 swcSIMDMin4U16 (ushort4 in_vec1, ushort4 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- ushort4 vector received as input
in	*in_vec2*	- ushort4 vector received as input

Returns

ushort4 vector

uint4 swcSIMDMin4U32 (uint4 in_vec1, uint4 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- uint4 vector received as input
in	*in_vec2*	- uint4 vector received as input

Returns

uint4 vector

uchar4 swcSIMDMin4U8 (uchar4 in_vec1, uchar4 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- uchar4 vector received as input
in	*in_vec2-*	uchar4 vector received as input

Returns

uchar4 vector

half8 swcSIMDMin8F16 (half8 in_vec1, half8 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- half8 vector received as input
in	*in_vec2*	- half8 vector received as input

Returns

half8 vector

short8 swcSIMDMin8I16 (short8 in_vec1, short8 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- short8 vector received as input
in	*in_vec2*	- short8 vector received as input

Returns

short8 vector

char8 swcSIMDMin8I8 (char8 in_vec1, char8 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- char8 vector received as input
in	*in_vec2*	- char8 vector received as input

Returns

char8 vector

ushort8 swcSIMDMin8U16 (ushort8 in_vec1, ushort8 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- ushort8 vector received as input
in	*in_vec2*	- ushort8 vector received as input

Returns

ushort8 vector

uchar8 swcSIMDMin8U8 (uchar8 in_vec1, uchar8 in_vec2)

This will compute the minumum value of vector.

Parameters

in	*in_vec1*	- uchar8 vector received as input
in	*in_vec2*	- uchar8 vector received as input

Returns

uchar8 vector

s32 swcSIMDSum16I8 (char16 in_vec)

Computes the Horizontal vector sum to scalar.

Parameters

in	*in_vec*	- vector received as input

Returns

s32 vector as output

u32 swcSIMDSum16U8 (uchar16 in_vec)

Computes the Horizontal vector sum to scalar.

Parameters

in	*in_vec*	- vector received as input

Returns

u32 vector as output

float swcSIMDSum4F32 (float4 in_vec)

Computes the Horizontal vector sum to scalar.

Parameters

in	*in_vec*	- vector received as input

Returns

float vector as output

s32 swcSIMDSum4I32 (int4 in_vec)

Computes the Horizontal vector sum to scalar.

Parameters

in	*in_vec*	- vector received as input

Returns

s32 vector as output

u32 swcSIMDSum4U32 (uint4 in_vec)

Computes the Horizontal vector sum to scalar.

Parameters

in	*in_vec*	- vector received as input

Returns

u32 vector as output

half swcSIMDSum8F16 (half8 in_vec)

Computes the Horizontal vector sum to scalar.

Parameters

in	*in_vec*	- vector received as input

Returns

half vector as output

s32 swcSIMDSum8I16 (short8 in_vec)

Computes the Horizontal vector sum to scalar.

Parameters

in	*in_vec*	- vector received as input

Returns

s32 vector as output

u32 swcSIMDSum8U16 (ushort8 in_vec)

Computes the Horizontal vector sum to scalar.

Parameters

in	*in_vec*	- vector received as input

Returns

u32 vector as output

float swcSIMDSumAbs4F32 (float4 in_vec)

Computes the Horizontal vector sum to scalar.

Parameters

in	*in_vec*	- vector received as input

Returns

float vector as output

half swcSIMDSumAbs8F16 (half8 in_vec)

Computes the Horizontal vector sum to scalar.

Parameters

in	*in_vec*	- vector received as input

Returns

half vector as output

Chapter 5

File Documentation

5.1 MDKdox-ShaveUtils-intro.txt File Reference

5.2 svuCommonShave.h File Reference

```
#include <mv_types.h>
#include "svuCommonShaveDefines.h"
#include "ShDrvMutex.h"
#include "ShDrvCmxFifo.h"
```

Macros

- #define SHAVE_HALT

  Shave halt instruction.
- #define scMutexRequest ShDrvMutexRequest

  Mutex Request.
- #define scMutexRequestNoWorkaround ShDrvMutexRequest
- #define scMutexRelease ShDrvMutexRelease

  This will release mutex.
- #define scFifoWrite ShDrvCmxFifoWriteWord

  Write a 32-bit value to a Shave FIFO.
- #define scFifoWriteDword ShDrvCmxFifoWriteDWord

  Write a 64-bit value to a Shave FIFO.
- #define scFifoReadDword ShDrvCmxFifoReadDWord

  Read a 64-bit value from the FIFO of the current shave.
- #define scFifoReadShaveDword ShDrvCmxFifoNReadDWord

  Read a 64-bit value from the FIFO of the current shave.
- #define scFifoWriteDirectDword ShDrvCmxFifoWriteDirectDWord

  Write a 64-bit value to a Shave FIFO directly into the FIFO memory. This permits to alter the values already contained in FIFO without changing the FIFO pointers.
- #define scFifoReadDirectDword ShDrvCmxFifoReadDirectDWord

  Read a 64-bit value from a Shave FIFO directly from the FIFO memory.
- #define scFifoRead ShDrvCmxFifoReadWord

Read a 32-bit value from the FIFO of the current shave.

- #define scFifoReadShave ShDrvCmxFifoNReadWord

  Read a 32-bit value from the FIFO of the shave given as parameter.
- #define scFifoMonitorSelect ShDrvCmxFifoMonitorSelect

  Configures the FIFO status bit to route through to the SHAVE for direct monitoring.
- #define scFifoWaitElement ShDrvCmxFifoMonitorWaitElement

  Wait for an element in the monitored FIFO.
- #define scFifoRAMControl ShDrvCmxFifoRAMControl

  Control RAM power modes. This permits to switch the FIFO RAM to different power states.
- #define scFifoGetAlmostFullLevel ShDrvCmxFifoGetAlmostFullLevel

  Get the value that was set for the "Almost full" FIFO fill level.
- #define scFifoSetAlmostFullLevel ShDrvCmxFifoSetAlmostFullLevel

  Set the 'Almost full' level for all the FIFOs.
- #define scFifoGetReadPtrValue ShDrvCmxFifoGetReadPtrValue

  Get the read pointer value for a specific Shave.
- #define scFifoGetWritePtrValue ShDrvCmxFifoGetWritePtrValue

  Get the write pointer value for a specific Shave.
- #define scFifoGetFillLevel ShDrvCmxFifoGetFillLevel

  Get the current fill level for a specific shave.
- #define scFifoIsFull ShDrvCmxFifoIsFull

  Check whether a FIFO is full or not.
- #define scFifoIsAlmostFull ShDrvCmxFifoIsAlmostFull

  Check whether a FIFO has reached the 'almost full' level of filling.
- #define scFifoIsEmpty ShDrvCmxFifoIsEmpty

  Check whether a FIFO is empty.
- #define scFifoWriteDirectWord ShDrvCmxFifoWriteDirectWord

  Write a 32-bit value to a Shave FIFO directly into the FIFO memory.
- #define scFifoReadDirectWord ShDrvCmxFifoReadDirectWord

  Read a 32-bit value directly from the FIFO memory without affecting the FIFO pointers.
- #define scFifoReadShaveDwordAtomic ShDrvCmxFifoNReadDWordAtomic

  Atomic read a 64-bit value from the CMX FIFO of the current shave.

Functions

- __asm (".nowarn 32\n"".include svuCommonDefinitions.incl\n"".nowarnend\n")

5.2.1 Detailed Description

Copyright

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

5.3 svuCommonShaveLUT.h File Reference

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

Functions

- ushort8 svuGet16BitVals16BitLUT (ushort8 in_values, u16 ∗lut_memory)

  Function that reads 8 ushort values into a ushort8 vector from LUT.
- uchar8 svuGet8BitVals16BitLUT (ushort8 in_values, u16 ∗lut_memory)

  Function that reads 8 uchar values into a ushort8 vector from LUT.
- uchar8 svuGet8BitVals8BitLUT (uchar8 in_values, u8 ∗lut_memory)

  Function that reads 8 uchar values into a uchar8 vector from LUT.
- uchar16 svuGet16_8BitVals8BitLUT (uchar16 in_values, u8 ∗lut_memory)

  Function that reads 16 uchar values into a uchar16 vector from LUT.
- ushort8 svuGetu16BitVals16BitLUT (ushort8 in_values, u16 ∗lut_memory)

  Function that reads 8 ushort values into a ushort8 vector from LUT.

5.3.1 Detailed Description

Copyright

5.4 swcCdma.h File Reference

```
#include "swcCdmaCommon.h"
#include <DrvRegUtilsDefines.h>
#include <stdarg.h>
#include <DrvIcbDefines.h>
#include <swcWhoAmI.h>
```

Functions

- dmaRequesterId dmaInitRequesterWithAgent (int priority, int agentToAssign)

  Initialize a requester ID which will be used to properly initialize and distinguish single tasks or groups of tasks.
- void dmaSetUsedAgents (u8 nrOfUsedAgents, u8 startingFrom)

  Set up the number of link agents the driver will use in order to start new tasks.
- u32 dmaSolveRelAddr (u32 inAddr, u32 shaveNumber)

  Translate windowed address into real physical address. Non-windowed address are passed through.
- dmaTransactionList ∗ dmaCreateTransactionExt (u32 Type, dmaRequesterId ReqId, dmaTransactionList ∗NewTransaction, u8 ∗Src, u8 ∗Dst, u32 ByteLength, u32 SrcLineWidth, u32 DstLineWidth, s32 SrcStride, s32 DstStride, u8 BurstLength)

  CMXDMA task structure initialization extension, allowing the user to set a custom burst length.

5.4.1 Detailed Description

Copyright

5.5 swcSIMDUtils.h File Reference

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

Functions

- float4 swcSIMDAbs4F32 (float4 in_vec)

  This will compute the absolute value of vector.
- float2 swcSIMDAbs2F32 (float2 in_vec)

  This will compute the absolute value of vector.
- half8 swcSIMDAbs8F16 (half8 in_vec)

  This will compute the absolute value of vector.
- half4 swcSIMDAbs4F16 (half4 in_vec)

  This will compute the absolute value of vector.
- half2 swcSIMDAbs2F16 (half2 in_vec)

  This will compute the absolute value of vector.
- int4 swcSIMDAbs4I32 (int4 in_vec)

  This will compute the absolute value of vector.
- int3 swcSIMDAbs3I32 (int3 in_vec)

  This will compute the absolute value of vector.
- int2 swcSIMDAbs2I32 (int2 in_vec)

  This will compute the absolute value of vector.
- short8 swcSIMDAbs8I16 (short8 in_vec)

  This will compute the absolute value of vector.
- short4 swcSIMDAbs4I16 (short4 in_vec)

  This will compute the absolute value of vector.
- short2 swcSIMDAbs2I16 (short2 in_vec)

  This will compute the absolute value of vector.
- char16 swcSIMDAbs16I8 (char16 in_vec)

  This will compute the absolute value of vector.
- char8 swcSIMDAbs8I8 (char8 in_vec)

  This will compute the absolute value of vector.
- char4 swcSIMDAbs4I8 (char4 in_vec)

  This will compute the absolute value of vector.
- char2 swcSIMDAbs2I8 (char2 in_vec)

  This will compute the absolute value of vector.
- float4 swcSIMDMin4F32 (float4 in_vec1, float4 in_vec2)

This will compute the minumum value of vector.

- float2 swcSIMDMin2F32 (float2 in_vec1, float2 in_vec2)

  This will compute the minumum value of vector.
- int4 swcSIMDMin4I32 (int4 in_vec1, int4 in_vec2)

  This will compute the minumum value of vector.
- int3 swcSIMDMin3I32 (int3 in_vec1, int3 in_vec2)

  This will compute the minumum value of vector.
- int2 swcSIMDMin2I32 (int2 in_vec1, int2 in_vec2)

  This will compute the minumum value of vector.
- uint4 swcSIMDMin4U32 (uint4 in_vec1, uint4 in_vec2)

  This will compute the minumum value of vector.
- uint3 swcSIMDMin3U32 (uint3 in_vec1, uint3 in_vec2)

  This will compute the minumum value of vector.
- uint2 swcSIMDMin2U32 (uint2 in_vec1, uint2 in_vec2)

  This will compute the minumum value of vector.
- half8 swcSIMDMin8F16 (half8 in_vec1, half8 in_vec2)

  This will compute the minumum value of vector.
- half4 swcSIMDMin4F16 (half4 in_vec1, half4 in_vec2)

  This will compute the minumum value of vector.
- half2 swcSIMDMin2F16 (half2 in_vec1, half2 in_vec2)

  This will compute the minumum value of vector.
- short8 swcSIMDMin8I16 (short8 in_vec1, short8 in_vec2)

  This will compute the minumum value of vector.
- short4 swcSIMDMin4I16 (short4 in_vec1, short4 in_vec2)

  This will compute the minumum value of vector.
- short2 swcSIMDMin2I16 (short2 in_vec1, short2 in_vec2)

  This will compute the minumum value of vector.
- ushort8 swcSIMDMin8U16 (ushort8 in_vec1, ushort8 in_vec2)

  This will compute the minumum value of vector.
- ushort4 swcSIMDMin4U16 (ushort4 in_vec1, ushort4 in_vec2)

  This will compute the minumum value of vector.
- ushort2 swcSIMDMin2U16 (ushort2 in_vec1, ushort2 in_vec2)

  This will compute the minumum value of vector.
- char16 swcSIMDMin16I8 (char16 in_vec1, char16 in_vec2)

  This will compute the minumum value of vector.
- char8 swcSIMDMin8I8 (char8 in_vec1, char8 in_vec2)

  This will compute the minumum value of vector.
- char4 swcSIMDMin4I8 (char4 in_vec1, char4 in_vec2)

  This will compute the minumum value of vector.
- char2 swcSIMDMin2I8 (char2 in_vec1, char2 in_vec2)

  This will compute the minumum value of vector.
- uchar16 swcSIMDMin16U8 (uchar16 in_vec1, uchar16 in_vec2)

  This will compute the minumum value of vector.
- uchar8 swcSIMDMin8U8 (uchar8 in_vec1, uchar8 in_vec2)

  This will compute the minumum value of vector.

- uchar4 swcSIMDMin4U8 (uchar4 in_vec1, uchar4 in_vec2)

  This will compute the minumum value of vector.
- uchar2 swcSIMDMin2U8 (uchar2 in_vec1, uchar2 in_vec2)

  This will compute the minumum value of vector.
- float4 swcSIMDMax4F32 (float4 in_vec1, float4 in_vec2)

  This will compute the maximum value of vector.
- float2 swcSIMDMax2F32 (float2 in_vec1, float2 in_vec2)

  This will compute the maximum value of vector.
- int4 swcSIMDMax4I32 (int4 in_vec1, int4 in_vec2)

  This will compute the maximum value of vector.
- int3 swcSIMDMax3I32 (int3 in_vec1, int3 in_vec2)

  This will compute the maximum value of vector.
- int2 swcSIMDMax2I32 (int2 in_vec1, int2 in_vec2)

  This will compute the maximum value of vector.
- uint4 swcSIMDMax4U32 (uint4 in_vec1, uint4 in_vec2)

  This will compute the maximum value of vector.
- uint3 swcSIMDMax3U32 (uint3 in_vec1, uint3 in_vec2)

  This will compute the maximum value of vector.
- uint2 swcSIMDMax2U32 (uint2 in_vec1, uint2 in_vec2)

  This will compute the maximum value of vector.
- half8 swcSIMDMax8F16 (half8 in_vec1, half8 in_vec2)

  This will compute the maximum value of vector.
- half4 swcSIMDMax4F16 (half4 in_vec1, half4 in_vec2)

  This will compute the maximum value of vector.
- half2 swcSIMDMax2F16 (half2 in_vec1, half2 in_vec2)

  This will compute the maximum value of vector.
- short8 swcSIMDMax8I16 (short8 in_vec1, short8 in_vec2)

  This will compute the maximum value of vector.
- short4 swcSIMDMax4I16 (short4 in_vec1, short4 in_vec2)

  This will compute the maximum value of vector.
- short2 swcSIMDMax2I16 (short2 in_vec1, short2 in_vec2)

  This will compute the maximum value of vector.
- ushort8 swcSIMDMax8U16 (ushort8 in_vec1, ushort8 in_vec2)

  This will compute the maximum value of vector.
- ushort4 swcSIMDMax4U16 (ushort4 in_vec1, ushort4 in_vec2)

  This will compute the maximum value of vector.
- ushort2 swcSIMDMax2U16 (ushort2 in_vec1, ushort2 in_vec2)

  This will compute the maximum value of vector.
- char16 swcSIMDMax16I8 (char16 in_vec1, char16 in_vec2)

  This will compute the maximum value of vector.
- char8 swcSIMDMax8I8 (char8 in_vec1, char8 in_vec2)

  This will compute the maximum value of vector.
- char4 swcSIMDMax4I8 (char4 in_vec1, char4 in_vec2)

  This will compute the maximum value of vector.
- char2 swcSIMDMax2I8 (char2 in_vec1, char2 in_vec2)

This will compute the maximum value of vector.

- uchar16 swcSIMDMax16U8 (uchar16 in_vec1, uchar16 in_vec2)

  This will compute the maximum value of vector.
- uchar8 swcSIMDMax8U8 (uchar8 in_vec1, uchar8 in_vec2)

  This will compute the maximum value of vector.
- uchar4 swcSIMDMax4U8 (uchar4 in_vec1, uchar4 in_vec2)

  This will compute the maximum value of vector.
- uchar2 swcSIMDMax2U8 (uchar2 in_vec1, uchar2 in_vec2)

  This will compute the maximum value of vector.
- float swcSIMDSum4F32 (float4 in_vec)

  Computes the Horizontal vector sum to scalar.
- float swcSIMDSumAbs4F32 (float4 in_vec)

  Computes the Horizontal vector sum to scalar.
- half swcSIMDSum8F16 (half8 in_vec)

  Computes the Horizontal vector sum to scalar.
- half swcSIMDSumAbs8F16 (half8 in_vec)

  Computes the Horizontal vector sum to scalar.
- u32 swcSIMDSum4U32 (uint4 in_vec)

  Computes the Horizontal vector sum to scalar.
- s32 swcSIMDSum4I32 (int4 in_vec)

  Computes the Horizontal vector sum to scalar.
- u32 swcSIMDSum8U16 (ushort8 in_vec)

  Computes the Horizontal vector sum to scalar.
- s32 swcSIMDSum8I16 (short8 in_vec)

  Computes the Horizontal vector sum to scalar.
- u32 swcSIMDSum16U8 (uchar16 in_vec)

  Computes the Horizontal vector sum to scalar.
- s32 swcSIMDSum16I8 (char16 in_vec)

  Computes the Horizontal vector sum to scalar.

5.5.1 Detailed Description

Copyright

Index

__asm
  Common Shave API, 15

CMXDMA API, 19
  dmaCreateTransactionExt, 19
  dmaInitRequesterWithAgent, 20
  dmaSetUsedAgents, 20
  dmaSolveRelAddr, 20
Common Shave API, 6
  __asm, 15
  SHAVE_HALT, 15
  scFifoGetAlmostFullLevel, 7
  scFifoGetFillLevel, 7
  scFifoGetReadPtrValue, 9
  scFifoGetWritePtrValue, 9
  scFifoIsAlmostFull, 9
  scFifoIsEmpty, 9
  scFifoIsFull, 10
  scFifoMonitorSelect, 10
  scFifoRAMControl, 10
  scFifoRead, 11
  scFifoReadDirectDword, 11
  scFifoReadDirectWord, 11
  scFifoReadDword, 11
  scFifoReadShave, 11
  scFifoReadShaveDword, 13
  scFifoReadShaveDwordAtomic, 13
  scFifoSetAlmostFullLevel, 13
  scFifoWaitElement, 13
  scFifoWrite, 13
  scFifoWriteDirectDword, 14
  scFifoWriteDirectWord, 14
  scFifoWriteDword, 14
  scMutexRelease, 14
  scMutexRequest, 15
  scMutexRequestNoWorkaround, 15

dmaCreateTransactionExt
  CMXDMA API, 19
dmaInitRequesterWithAgent
  CMXDMA API, 20
dmaSetUsedAgents
  CMXDMA API, 20
dmaSolveRelAddr
  CMXDMA API, 20

MDKdox-ShaveUtils-intro.txt, 45

SHAVE_HALT
  Common Shave API, 15
SIMD Utilities, 22
  swcSIMDAbs16I8, 25
  swcSIMDAbs2F16, 25
  swcSIMDAbs2F32, 26
  swcSIMDAbs2I16, 26
  swcSIMDAbs2I32, 26
  swcSIMDAbs2I8, 26
  swcSIMDAbs3I32, 27
  swcSIMDAbs4F16, 27
  swcSIMDAbs4F32, 27
  swcSIMDAbs4I16, 27
  swcSIMDAbs4I32, 28
  swcSIMDAbs4I8, 28
  swcSIMDAbs8F16, 28
  swcSIMDAbs8I16, 28
  swcSIMDAbs8I8, 29
  swcSIMDMax16I8, 29
  swcSIMDMax16U8, 29
  swcSIMDMax2F16, 29
  swcSIMDMax2F32, 30
  swcSIMDMax2I16, 30
  swcSIMDMax2I32, 30
  swcSIMDMax2I8, 30
  swcSIMDMax2U16, 31
  swcSIMDMax2U32, 31
  swcSIMDMax2U8, 31
  swcSIMDMax3I32, 31
  swcSIMDMax3U32, 32
  swcSIMDMax4F16, 32
  swcSIMDMax4F32, 32
  swcSIMDMax4I16, 32
  swcSIMDMax4I32, 33
  swcSIMDMax4I8, 33
  swcSIMDMax4U16, 33
  swcSIMDMax4U32, 33
  swcSIMDMax4U8, 34
  swcSIMDMax8F16, 34
  swcSIMDMax8I16, 34

swcSIMDMax8I8, 34
swcSIMDMax8U16, 35
swcSIMDMax8U8, 35
swcSIMDMin16I8, 35
swcSIMDMin16U8, 35
swcSIMDMin2F16, 36
swcSIMDMin2F32, 36
swcSIMDMin2I16, 36
swcSIMDMin2I32, 36
swcSIMDMin2I8, 37
swcSIMDMin2U16, 37
swcSIMDMin2U32, 37
swcSIMDMin2U8, 37
swcSIMDMin3I32, 38
swcSIMDMin3U32, 38
swcSIMDMin4F16, 38
swcSIMDMin4F32, 38
swcSIMDMin4I16, 39
swcSIMDMin4I32, 39
swcSIMDMin4I8, 39
swcSIMDMin4U16, 39
swcSIMDMin4U32, 40
swcSIMDMin4U8, 40
swcSIMDMin8F16, 40
swcSIMDMin8I16, 40
swcSIMDMin8I8, 41
swcSIMDMin8U16, 41
swcSIMDMin8U8, 41
swcSIMDSum16I8, 41
swcSIMDSum16U8, 42
swcSIMDSum4F32, 42
swcSIMDSum4I32, 42
swcSIMDSum4U32, 42
swcSIMDSum8F16, 43
swcSIMDSum8I16, 43
swcSIMDSum8U16, 43
swcSIMDSumAbs4F32, 43
swcSIMDSumAbs8F16, 44
scFifoGetAlmostFullLevel
Common Shave API, 7
scFifoGetFillLevel
Common Shave API, 7
scFifoGetReadPtrValue
Common Shave API, 9
scFifoGetWritePtrValue
Common Shave API, 9
scFifoIsAlmostFull
Common Shave API, 9
scFifoIsEmpty
Common Shave API, 9
scFifoIsFull
Common Shave API, 10
scFifoMonitorSelect
Common Shave API, 10
scFifoRAMControl
Common Shave API, 10
scFifoRead
Common Shave API, 11
scFifoReadDirectDword
Common Shave API, 11
scFifoReadDirectWord
Common Shave API, 11
scFifoReadDword
Common Shave API, 11
scFifoReadShave
Common Shave API, 11
scFifoReadShaveDword
Common Shave API, 13
scFifoReadShaveDwordAtomic
Common Shave API, 13
scFifoSetAlmostFullLevel
Common Shave API, 13
scFifoWaitElement
Common Shave API, 13
scFifoWrite
Common Shave API, 13
scFifoWriteDirectDword
Common Shave API, 14
scFifoWriteDirectWord
Common Shave API, 14
scFifoWriteDword
Common Shave API, 14
scMutexRelease
Common Shave API, 14
scMutexRequest
Common Shave API, 15
scMutexRequestNoWorkaround
Common Shave API, 15
Shave LUT, 16
svuGet16_8BitVals8BitLUT, 16
svuGet16BitVals16BitLUT, 16
svuGet8BitVals16BitLUT, 16
svuGet8BitVals8BitLUT, 18
svuGetu16BitVals16BitLUT, 18
svuCommonShave.h, 45
svuCommonShaveLUT.h, 47
svuGet16_8BitVals8BitLUT
Shave LUT, 16
svuGet16BitVals16BitLUT
Shave LUT, 16
svuGet8BitVals16BitLUT
Shave LUT, 16

svuGet8BitVals8BitLUT
Shave LUT, 18
svuGetu16BitVals16BitLUT
Shave LUT, 18
swcCdma.h, 47
swcSIMDAbs16I8
SIMD Utilities, 25
swcSIMDAbs2F16
SIMD Utilities, 25
swcSIMDAbs2F32
SIMD Utilities, 26
swcSIMDAbs2I16
SIMD Utilities, 26
swcSIMDAbs2I32
SIMD Utilities, 26
swcSIMDAbs2I8
SIMD Utilities, 26
swcSIMDAbs3I32
SIMD Utilities, 27
swcSIMDAbs4F16
SIMD Utilities, 27
swcSIMDAbs4F32
SIMD Utilities, 27
swcSIMDAbs4I16
SIMD Utilities, 27
swcSIMDAbs4I32
SIMD Utilities, 28
swcSIMDAbs4I8
SIMD Utilities, 28
swcSIMDAbs8F16
SIMD Utilities, 28
swcSIMDAbs8I16
SIMD Utilities, 28
swcSIMDAbs8I8
SIMD Utilities, 29
swcSIMDMax16I8
SIMD Utilities, 29
swcSIMDMax16U8
SIMD Utilities, 29
swcSIMDMax2F16
SIMD Utilities, 29
swcSIMDMax2F32
SIMD Utilities, 30
swcSIMDMax2I16
SIMD Utilities, 30
swcSIMDMax2I32
SIMD Utilities, 30
swcSIMDMax2I8
SIMD Utilities, 30
swcSIMDMax2U16
SIMD Utilities, 31
swcSIMDMax2U32
SIMD Utilities, 31
swcSIMDMax2U8
SIMD Utilities, 31
swcSIMDMax3I32
SIMD Utilities, 31
swcSIMDMax3U32
SIMD Utilities, 32
swcSIMDMax4F16
SIMD Utilities, 32
swcSIMDMax4F32
SIMD Utilities, 32
swcSIMDMax4I16
SIMD Utilities, 32
swcSIMDMax4I32
SIMD Utilities, 33
swcSIMDMax4I8
SIMD Utilities, 33
swcSIMDMax4U16
SIMD Utilities, 33
swcSIMDMax4U32
SIMD Utilities, 33
swcSIMDMax4U8
SIMD Utilities, 34
swcSIMDMax8F16
SIMD Utilities, 34
swcSIMDMax8I16
SIMD Utilities, 34
swcSIMDMax8I8
SIMD Utilities, 34
swcSIMDMax8U16
SIMD Utilities, 35
swcSIMDMax8U8
SIMD Utilities, 35
swcSIMDMin16I8
SIMD Utilities, 35
swcSIMDMin16U8
SIMD Utilities, 35
swcSIMDMin2F16
SIMD Utilities, 36
swcSIMDMin2F32
SIMD Utilities, 36
swcSIMDMin2I16
SIMD Utilities, 36
swcSIMDMin2I32
SIMD Utilities, 36
swcSIMDMin2I8
SIMD Utilities, 37
swcSIMDMin2U16
SIMD Utilities, 37
swcSIMDMin2U32

SIMD Utilities, 37
swcSIMDMin2U8
SIMD Utilities, 37
swcSIMDMin3I32
SIMD Utilities, 38
swcSIMDMin3U32
SIMD Utilities, 38
swcSIMDMin4F16
SIMD Utilities, 38
swcSIMDMin4F32
SIMD Utilities, 38
swcSIMDMin4I16
SIMD Utilities, 39
swcSIMDMin4I32
SIMD Utilities, 39
swcSIMDMin4I8
SIMD Utilities, 39
swcSIMDMin4U16
SIMD Utilities, 39
swcSIMDMin4U32
SIMD Utilities, 40
swcSIMDMin4U8
SIMD Utilities, 40
swcSIMDMin8F16
SIMD Utilities, 40
swcSIMDMin8I16
SIMD Utilities, 40
swcSIMDMin8I8
SIMD Utilities, 41
swcSIMDMin8U16
SIMD Utilities, 41
swcSIMDMin8U8
SIMD Utilities, 41
swcSIMDSum16I8
SIMD Utilities, 41
swcSIMDSum16U8
SIMD Utilities, 42
swcSIMDSum4F32
SIMD Utilities, 42
swcSIMDSum4I32
SIMD Utilities, 42
swcSIMDSum4U32
SIMD Utilities, 42
swcSIMDSum8F16
SIMD Utilities, 43
swcSIMDSum8I16
SIMD Utilities, 43
swcSIMDSum8U16
SIMD Utilities, 43
swcSIMDSumAbs4F32
SIMD Utilities, 43
swcSIMDSumAbs8F16
SIMD Utilities, 44
swcSIMDUtils.h, 48