SIS1100
PCI/PCIe VME Interface

API Reference

SIS GmbH
Harksheider Str. 102A
22399 Hamburg
Germany

Phone: ++49 (0) 40 60 87 305 0
Fax: ++49 (0) 40 60 87 305 20

email: info@struck.de
http://www.struck.de

Version: SIS1100-M-0001-Api_Reference-V100.doc as of 25.10.2011

Revision Table:

Revision	Date	Modification
0.1	12.08.2011	created
1.0	25.10.2011	bump to 1.0

- Table of contents

1 Overview

This section provides an overview over the implemented SIS1100 API functions.

API Function Name	Description
sis1100w_Logging	Enable or disable driver debug logfile
sis1100w_Api_Version	Get sis1100w and Plx Api revisions
sis1100w_Get_Last_Plx_Error	Get last internal error number
sis1100w_Find_No_Of_sis1100	Get amount of installed cards
sis1100w_Get_Handle_And_Open	Get device handle based on device index
sis1100w_Close	Release previously allocated device handle
sis1100w_Init	Initialize device
sis1100w_Get_IdentVersion	Get device ident and version register
sis1100w_Init_sis3100	Initialize remote VME interface
sis1100w_VmeSysreset	Issue VME sysreset
sis1100w_sis1100_Control_Read	Read from local register space
sis1100w_sis1100_Control_Write	Write to local register space
sis1100w_sis3100_Control_Read	Read from remote register space
sis1100w_sis3100_Control_Write	Write to remote register space
sis1100w_Vme_Single_Read	Read via single cycle from VME address
sis1100w_Vme_Single_Write	Write via single cycle to VME address
sis1100w_Vme_Dma_Read	Read via blocktransfer from VME address
sis1100w_Vme_Dma_Write	Write via blocktransfer to VME address
sis1100w_SdramSharc_Single_Read	Read via single cycle from 3100 sdram
sis1100w_SdramSharc_Single_Write	Write via single cycle to 3100 sdram
sis1100w_SdramSharc_Dma_Read	Read via blocktransfer from 3100 sdram
sis1100w_SdramSharc_Dma_Write	Write via blocktransfer to 3100 sdram

1.1 sis1100w_Logging

Syntax:

```
SIS1100W_STATUS
sis1100w_Logging(
       BOOLEAN enable
       );
```

Description:
Enables or disables the debug logging to file.

Note:
Logging is enabled by default due to compaibility reasons.

Arguments:
enable
enable or disable logging

Return Codes:

Code	Description
StatSuccess	The function returned successfully

Usage:

```
SIS1100W_STATUS status;

// disable logging
status = sis1100w_Logging(
                        FALSE
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_Logging': %x\n", status);
}
```

1.2 sis1100w_Api_Version

Syntax:

```
SIS1100W_STATUS
sis1100w_Api_Version(
       PSIS1100W_VERSION version
       );
```

Description:
Returns revision information.

Arguments:

version

Pointer to a SIS1100W_VERSION variable.

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL

Usage:

```
SIS1100W_STATUS status;
SIS1100W_VERSION version;

status = sis1100w_ApiVersion(
                        &version
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_Api_Version': %x\n", status);
}else{
      printf("Revisions: SIS: v%d.%d PLX: v%d.%d\n",
            version.SisMajor,
            version.SisMinor,
            version.PlxMajor,
            version.PlxMinor);
}
```

1.3 sis1100w_Get_Last_Plx_Error

Syntax:

```
SIS1100W_STATUS
sis1100w_Get_Last_Plx_Error(
      int *err
      );
```

Description:
Returns the last internal PlxApi error. This may be used for debugging purposes.

Arguments:

err

Pointer to a signed 32-bit value

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL

Usage:

```
SIS1100W_STATUS status;
int err;

status = sis1100w_Get_Last_Plx_Error(
                        &err
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_Get_Last_Plx_Error': %x\n", status);
}else{
      printf("last Plx error was: %x.\n", err);
}
```

1.4 sis1100w_Find_No_Of_sis1100

Syntax:

```
SIS1100W_STATUS
sis1100w_Find_No_Of_sis1100(
       UINT *no_found
       );
```

Description:
Returns the amount of installed SIS1100 cards.

Arguments:

no_found

Pointer to an unsigned 32-bit value.

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL

Usage:

```
SIS1100W_STATUS status;
UINT no_found;

status = sis1100w_Get_No_Of_sis1100(
                        &no_found
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_Get_No_Of_sis1100': %x\n", status);
}
```

1.5 sis1100w_Get_Handle_And_Open

Syntax:

```
SIS1100W_STATUS
sis1100w_Get_Handle_And_Open(
       UINT deviceNo,
       struct SIS1100_Device_Struct *handle
       );
```

Description:
Tries to open a device handle for a card with the supplied index 'deviceNo'.
The 'deviceNo' index parameter is 0-based.

Arguments:

deviceNo
: 0-based index of card to be opened

handle
: Pointer to a device handle structure.

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatErrorDeviceOpen	Can't open a handle on supplied index
StatErrorBarMap	Error mapping register space
StatErrorUserHandle	Error allocating memory for user handle

Usage:

```
struct SIS1100_Device_Struct hDevice;
SIS1100W_STATUS status;
UINT card;

card = 0; // use known correct value from 'sis1100w_Find_No_Of_sis1100'

status = sis1100w_Get_Handle_And_Open(
                        card,
                        &hDevice
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_Get_Handle_And_Open': %x\n", status);
}
```

1.6 sis1100w_Close

Syntax:

```
SIS1100W_STATUS
sis1100w_Close(
       struct SIS1100_Device_Struct *handle
       );
```

Description:
Releases a previously allocated device handle.

Arguments:
handle
Pointer to an opened device handle structure.

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatErrorBarUnmap	Error unmapping register space
StatErrorDeviceClose	Error closing PLX handle

Usage:

```
SIS1100W_STATUS status;
struct SIS1100_Device_Struct hDevice; // previously opened

status = sis1100w_Close(
                        &hDevice
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_Close': %x\n", status);
}
```

1.7 sis1100w_Init

Syntax:

```
SIS1100W_STATUS
sis1100w_Init(
       struct SIS1100_Device_Struct *handle
       );
```

Description:
Initializes the local interface. Should be called before using any local control read or write calls.

Arguments:

handle

Pointer to an opened device handle structure.

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatErrorDeviceReset	Error resetting PLX chip
StatErrorRegisterAccess	Error accessing PLX config registers

Usage:

```
SIS1100W_STATUS status;
struct SIS1100_Device_Struct hDevice; // previously opened

status = sis1100w_Init(
                        &hDevice
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_Init: %x\n", status);
}
```

1.8 sis1100w_Get_IdentVersion

Syntax:

```
SIS1100W_STATUS
sis1100w_Get_IdentVersion(
       struct SIS1100_Device_Struct *handle,
       UINT *ident
       );
```

Description:
Returns the local device ident and revision register.

Arguments:

handle
: Pointer to an opened device handle structure.

ident
: Pointer to an unsigned 32-bit value.

Return Codes:

Code	Description
`ApiSuccess`	The function returned successfully
`ApiNullArgument`	At least 1 pointer argument is NULL

Usage:

```
SIS1100W_STATUS status;
UINT ident;

status = sis1100w_Get_IdentVersion(
                        &hDevice,
                        &ident
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_Get_IdentVersion': %x\n", status);
}else{
      printf("SIS1100 ident: %08X\n", ident);
}
```

1.9 sis1100w_Init_sis3100

Syntax:

```
SIS1100W_STATUS
sis1100w_Init_sis31000(
       struct SIS1100_Device_Struct *hande
       );
```

Description:
Initializes the remote side of the VME interface. Must be called before issuing any VME access calls.

Arguments:

handle

Pointer to an opened device handle structure.

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatErrorNoRemoteDevice	No remote device present (link down)

Usage:

```
SIS1100W_STATUS status;

status = sis1100w_Init_sis3100(
                        &hDevice
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_Init_sis3100': %x\n", status);
}
```

1.10 sis1100w_VmeSysreset

Syntax:

```
SIS1100W_STATUS
sis1100w_VmeSysreset(
      struct SIS1100_Device_Struct *handle
      );
```

Description:
Issues 300ms VME sysreset pulse.

Arguments:

handle

Pointer to an opened device handle structure.

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatLE<..>	Local error

Usage:

```
SIS1100W_STATUS status;

status = sis1100w_VmeSysreset(
                        &hDevice
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_VmeSysreset': %x\n", status);
}
```

1.11 sis1100w_sis1100_Control_Read

Syntax:

```
SIS1100W_STATUS
sis1100w_sis1100_Control_Read(
       struct SIS1100_Device_Struct *handle,
       UINT offset,
       UINT *data
       );
```

Description:
Reads from local control register.

Arguments:

handle
: Pointer to an opened device handle structure.

offset
: Register address

data
: Pointer to an unsigned 32-bit buffer

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL

Usage:

```
SIS1100W_STATUS status;
UINT offset;
UINT data;

offset = 0x4;

status = sis1100w_sis1100_Control_Read(
                        &hDevice,
                        offset,
                        &data
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_sis1100_Control_Read': %x\n", status);
}else{
      printf("address: %X holds: %08X\n", offset, data);
}
```

1.12 sis1100w_sis1100_Control_Write

Syntax:

```
SIS1100W_STATUS
sis1100w_sis1100_Control_Write(
       struct SIS1100_Device_Struct *handle,
       UINT offset,
       UINT data
       );
```

Description:
Writes to local control register.

Arguments:

handle
: Pointer to an opened device handle structure.

offset
: Register address

data
: Unsigned 32-bit buffer

Return Codes:

Code	Description
ApiSuccess	The function returned successfully
ApiNullArgument	At least 1 pointer argument is NULL

Usage:

```
SIS1100W_STATUS status;
UINT offset;
UINT data;

offset = 0x4;
data = 0xDEADBEEF;

status = sis1100w_sis1100_Control_Write(
                        &hDevice,
                        offset,
                        data
                        );

if(status != StatSuccess){
       printf("Error in 'sis1100w_sis1100_Control_Write': %x\n", status);
}
```

1.13 sis1100w_sis3100_Control_Read

Syntax:

```
SIS1100W_STATUS
sis1100w_sis3100_Control_Read(
       struct SIS1100_Device_Struct *handle,
       UINT offset,
       UINT *data
       );
```

Description:
Reads from remote control register.

Arguments:

handle
: Pointer to an opened device handle structure.

offset
: Register address

data
: Pointer to an unsigned 32-bit buffer

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatLE<..>	Local Error
StatRE<..>	Remote Error

Usage:

```
SIS1100W_STATUS status;
UINT offset;
UINT data;

offset = 0x4;

status = sis1100w_sis3100_Control_Read(
                        &hDevice,
                        offset,
                        &data
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_sis3100_Control_Read': %x\n", status);
}else{
      printf("address: %X holds: %08X\n", offset, data);
}
```

1.14 sis1100w_sis3100_Control_Write

Syntax:

```
SIS1100W_STATUS
sis1100w_sis3100_Control_Write(
       struct SIS1100_Device_Struct *handle,
       UINT offset,
       UINT data
       );
```

Description:
Writes to remote control register.

Arguments:

handle
: Pointer to an opened device handle structure.

offset
: Register address

data
: Usigned 32-bit value

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatLE<..>	Local Error
StatRE<..>	Remote Error

Usage:

```
SIS1100W_STATUS status;
UINT offset;
UINT data;

offset = 0x4;
data = 0xDEADBEEF;

status = sis1100w_sis3100_Control_Write(
                        &hDevice,
                        offset,
                        data
                        );

if(status != StatSuccess){
      printf("Error in 'sis100w_sis3100_Control_Write': %x\n", status);
}
```

1.15 sis1100w_Vme_Single_Read

Syntax:

```
SIS1100W_STATUS
sis100w_Vme_Single_Read(
       struct SIS1100_Device_Struct *handle,
       UINT addr,
       UINT am,
       UINT size,
       UINT *data
       );
```

Description:
Issues a single cycle read request to VME space.

Note:
This function provides raw VME access.
It is recommended to use the supplied VME wrapper calls in 'sis3100_vme_calls.c'.

Arguments:

handle
: Pointer to an opened device handle structure.

addr
: VME address

am
: VME address modifier

size
: 1, 2 or 4 byte wide access

data
: Pointer to unsigned 32-bit value

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatLE<..>	Local error
StatRE<..>	Remote error

Usage:

```
SIS1100W_STATUS status;
UINT addr;
UINT data;

addr = 0;

status = sis1100w_Vme_Single_Read(
                        &hDevice,
                        offset,
                        0x9, // A32D32
                        4, // 4byte wide
                        &data
```

```
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_Vme_Single_Read': %x\n", status);
}
```

1.16 sis1100w_Vme_Single_Write

Syntax:

```
SIS1100W_STATUS
sis100w_Vme_Single_Write(
       struct SIS1100_Device_Struct *handle,
       UINT addr,
       UINT am,
       UINT size,
       UINT data
       );
```

Description:
Issues a single cycle write request to VME space.

Note:
This function provides raw VME access.
It is recommended to use the supplied VME wrapper calls in 'sis3100_vme_calls.c'.

Arguments:

handle
: Pointer to an opened device handle structure.

addr
: VME address

am
: VME address modifier

size
: 1, 2 or 4 byte wide access

data
: Unsigned 32-bit value

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatLE<..>	Local error
StatRE<..>	Remote error

Usage:

```
SIS1100W_STATUS status;
UINT addr;
UINT data = 0xDEADBEEF;

addr = 0;

status = sis1100w_Vme_Single_Write(
                        &hDevice,
                        offset,
                        0x9, // A32D32
                        4, // 4byte wide
```

```
                        data
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_Vme_Single_Write': %x\n", status);
}
```

1.17 sis1100w_Vme_Dma_Read

Syntax:

```
SIS1100W_STATUS
sis100w_Vme_Dma_Read(
       struct SIS1100_Device_Struct *handle,
       UINT addr,
       UINT am,
       UINT size,
       UINT fifo_mode,
       UINT *dmabufs,
       UINT req_num_data,
       UINT *got_num_data
       );
```

Description:
Issues a blocktransfer read request to VME space.

Note:
This function provides raw VME access.
It is recommended to use the supplied VME wrapper calls in 'sis3100_vme_calls.c'.

Arguments:

handle
: Pointer to an opened device handle structure.

addr
: VME address

am
: VME address modifier

size
: 1, 2 or 4 byte wide access

fifo_mode
: holds VME address constant for fifo readout

dmabufs
: Pointer to unsigned 32-bit buffer

req_num_data
: Requested amount of 32-bit words

got_num_data
: Actual amount of transferred 32-bit words

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatLE<..>	Local error
StatRE<..>	Remote error

Usage:

```
SIS1100W_STATUS status;
```

```
UINT addr;
UINT data[1000];
UINT got_data_amount;

addr = 0;

status = sis1100w_Vme_Dma_Read(
                        &hDevice,
                        offset,
                        0x9, // A32D32
                        4, // 4byte wide
                        0, // no fifo_mode
                        &data,
                        1000, // request 1000 32-bit words
                        &got_data_amount // how much data has been received
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_Vme_Dma_Read': %x\n", status);
}
```

1.18 sis1100w_Vme_Dma_Write

Syntax:

```
SIS1100W_STATUS
sis100w_Vme_Dma_Write(
       struct SIS1100_Device_Struct *handle,
       UINT addr,
       UINT am,
       UINT size,
       UINT fifo_mode,
       UINT *dmabufs,
       UINT req_num_data,
       UINT *put_num_data
       );
```

Description:
Issues a blocktransfer write request to VME space.

Note:
This function provides raw VME access.
It is recommended to use the supplied VME wrapper calls in 'sis3100_vme_calls.c'.

Arguments:

handle
: Pointer to an opened device handle structure.

addr
: VME address

am
: VME address modifier

size
: 1, 2 or 4 byte wide access

fifo_mode
: holds VME address constant for fifo readout

dmabufs
: Pointer to unsigned 32-bit buffer

req_num_data
: Requested amount of 32-bit words

put_num_data
: Actual amount of transferred 32-bit words

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatLE<..>	Local error
StatRE<..>	Remote error

Usage:

```
SIS1100W_STATUS status;
```

```
UINT addr;
UINT data[1000];
UINT put_data_amount;

addr = 0;
// fill data with sane values

status = sis1100w_Vme_Dma_Write(
                        &hDevice,
                        offset,
                        0x9, // A32D32
                        4, // 4byte wide
                        0, // no fifo_mode
                        &data,
                        1000, // write 1000 32-bit words
                        &put_data_amount // how much data has been written
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_Vme_Dma_Write': %x\n", status);
}
```

1.19 sis1100w_SdramSharc_Single_Read

Syntax:

```
SIS1100W_STATUS
sis100w_SdramSharcSingle_Read(
       struct SIS1100_Device_Struct *handle,
       UINT addr,
       UINT *data
       );
```

Description:
Issues a single cycle read request to the SIS3100 onboard SDram (if populated).

Arguments:

handle
: Pointer to an opened device handle structure.

addr
: Memory address

data
: Pointer to unsigned 32-bit value

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatLE<..>	Local error
StatRE<..>	Remote error

Usage:

```
SIS1100W_STATUS status;
UINT addr;
UINT data;

addr = 0;

status = sis1100w_SdramSharc_Single_Read(
                        &hDevice,
                        offset,
                        &data
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_SdramSharc_Single_Read': %x\n", status);
}
```

1.20 sis1100w_SdramSharc_Single_Write

Syntax:

```
SIS1100W_STATUS
sis100w_SdramSharc_Single_Write(
       struct SIS1100_Device_Struct *handle,
       UINT addr,
       UINT data
       );
```

Description:
Issues a single cycle write request to the SIS3100 onboard SDram (if populated).

Arguments:

handle
: Pointer to an opened device handle structure.

addr
: Memory address

data
: Unsigned 32-bit value

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatLE<..>	Local error
StatRE<..>	Remote error

Usage:

```
SIS1100W_STATUS status;
UINT addr;
UINT data = 0xDEADBEEF;

addr = 0;

status = sis1100w_SdramSharc_Single_Write(
                        &hDevice,
                        offset,
                        data
                        );

if(status != StatSuccess){
      printf("Error in 'sis1100w_SdramSharc_Single_Write': %x\n", status);
}
```

1.21 sis1100w_SdramSharc_Dma_Read

Syntax:

```
SIS1100W_STATUS
sis100w_Vme_Dma_Read(
       struct SIS1100_Device_Struct *handle,
       UINT addr,
       UINT *dmabufs,
       UINT req_num_data,
       UINT *got_num_data
       );
```

Description:
Issues a blocktransfer read request to the SIS3100 onboard SDram (if populated).

Arguments:

handle
: Pointer to an opened device handle structure.

addr
: Memory address

dmabufs
: Pointer to unsigned 32-bit buffer

req_num_data
: Requested amount of 32-bit words

got_num_data
: Actual amount of transferred 32-bit words

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatLE<..>	Local error
StatRE<..>	Remote error

Usage:

```
SIS1100W_STATUS status;
UINT addr;
UINT data[1000];
UINT got_data_amount;

addr = 0;

status = sis1100w_SdramSharc_Dma_Read(
                        &hDevice,
                        offset,
                        &data,
                        1000, // request 1000 32-bit words
                        &got_data_amount // how much data has been received
                        );

if(status != StatSuccess){
```

```
      printf("Error in 'sis1100w_SdramSharc_Dma_Read': %x\n", status);
}
```

1.22 sis1100w_SdramSharc_Dma_Write

Syntax:

```
SIS1100W_STATUS
sis100w_SdramSharc_Dma_Write(
       struct SIS1100_Device_Struct *handle,
       UINT addr,
       UINT *dmabufs,
       UINT req_num_data,
       UINT *put_num_data
       );
```

Description:
Issues a blocktransfer write request to the SIS3100 onboard SDram (if populated).

Arguments:

handle
: Pointer to an opened device handle structure.

addr
: Memory address

dmabufs
: Pointer to unsigned 32-bit buffer

req_num_data
: Requested amount of 32-bit words

put_num_data
: Actual amount of transferred 32-bit words

Return Codes:

Code	Description
StatSuccess	The function returned successfully
StatNullArgument	At least 1 pointer argument is NULL
StatLE<..>	Local error
StatRE<..>	Remote error

Usage:

```
SIS1100W_STATUS status;
UINT addr;
UINT data[1000];
UINT put_data_amount;

addr = 0;
// fill data with sane values

status = sis1100w_SdramSharc_Dma_Write(
                        &hDevice,
                        offset,
                        &data,
                        1000, // write 1000 32-bit words
                        &put_data_amount // how much data has been written
                        );
```

```
if(status != StatSuccess){
      printf("Error in 'sis1100w_SdramSharc_Dma_Write': %x\n", status);
}
```

2 Data Structures

This section provides an overview over the available data structures.
See sis1100wType.h and/or sis1100wStat.h for reference.

2.1 struct SIS1100_Device_Struct

Syntax:

```
struct SIS100_Device_Struct{
	VOID *plx_Va;
	VOID *devObj;
};
```

Description:
Userspace card handle to be able to handle more than one card simultaneously.
This structure needs to be passed to all function which will access the hardware.
The members of this type should never be written directly.

2.2 SIS1100W_VERSION

Syntax:

```
typedef struct _SIS1100W_VERSION{
	UCHAR SisMajor;
	UCHAR SisMinor;
	UCHAR PlxMajor;
	UCHAR PlxMinor;
}SIS1100W_VERSION, *PSIS1100W_VERSION;
```

Description:
Struct used to retrieve Api revision information via the sis1100w_Api_Version call.

2.3 SIS1100W_STATUS

Syntax:

```
#define API_RETURNCODE_START 0x0
#define API_RETURNCODE_LOCAL 0x100
#define API_RETURNCODE_REMOTE 0x200

typedef enum _SIS1100W_STATUS{
	// 0: general return codes
	StatSuccess = API_RETURNCODE_START,
	StatNullArgument,
	StatInvalidDeviceIndex,
	StatErrorDeviceOpen,
	StatErrorDeviceReset,
	StatErrorDeviceClose,
	StatErrorBarMap,
	StatErrorBarUnmap,
	StatErrorAllocUserHandle,
	StatErrorRegisterAccess,
	StatErrorFifoFlush,
	StatErrorNoRemoteDevice,
```

```
        StatInvalidSizeValue,
        // 0x100: local side errors
        StatLESynch = API_RETURNCODE_LOCAL + 0x1, // starts at 0x101
        StatLENrdy,
        StatLEXoff,
        StatLEResource,
        StatLEDlock,
        StatLETimeout = API_RETURNCODE_LOCAL + 0x7,
        // 0x200: remote side errors
        StatRENrdy = API_RETURNCODE_REMOTE + 0x2, // starts at 0x202
        StatREProt = API_RETURNCODE_REMOTE + 0x6,
        StatRETimeout,
        StatREBerr,
        StatREFerr,
        StatREBusErr = API_RETURNCODE_REMOTE + 0x11 // SIS310x Berr at 0x211
}SIS1100W_STATUS;
```

Description:
A enum type to have a readable representation of the possible return code from each API function.

3 Index