Spansion® Low Level Driver User Guide

Release 11.3.2

Publication Number Spansion_LLD_User_Guide **Revision** 03 **Issue Date** September 13, 2011

Table of Contents

1. Introduction

The Low Level Driver (LLD) software from Spansion is an API that provides the most basic set of functions required to communicate with a Spansion flash memory device. In most cases, there is a one to one correspondence between commands listed in the data sheet and the LLD Commands. Very little customization is necessary to make the LLD work in your system. The integration of the LLD into your system will greatly reduce your flash driver development time. Please report errors in the documentation so we can improve the LLD code for future users (spansion.solutions@spansion.com).

This document describes the general Spansion flash device functionalities. Not all functions are applicable to your device. Please refer to the device data sheet for applicable functions.

1.1 Files

The LLD consists of the following:

- lld.c – This file contains the Common Commands. You should not need to change this file.
- lld.h – This file contains the external prototypes and the command macros. Include this file wherever you use LLD functions. You should not need to change this file.
- lld_target_specific.h – This file requires changes to work in your system.

Optionally, we provide the trace.c / trace.h modules that allow you to enable the software traces, which helps a lot during debug phases.

1.2 Making the LLD Work in Your Environment

The LLD was written to support various architectures. Changes to lld.c and lld.h should not be necessary.

The file lld_target_specific.h does require modification in order to work in your environment.

In lld_target_specific.h:

1. Select the include header file for the device that you are using. For example, if the device that you are using is S29GL512S, then select S29GLxxxS.h header file.
2. Define the LLD flash chip configuration by setting the LLD_CONFIGURATION to a value that matches your system. For example, if you are using two WS256Ns (interleaved), then set the LLD_CONFIGURATION to X16_AS_X32.
3. Define how the LLD will do memory reads and writes in your system. Define the macros FLASH_RD and FLASH_WR. The default macro should work for most systems.
4. The DelayMicroseconds() functions in lld.c and are based on the macro DELAY_1µs. If you choose to use the default lld.c delay functions, put a value in DELAY_1us that will give a one-microsecond delay.
5. Define the macro PAUSE_BETWEEN_ERASE_SUSPENDS if you are using the erase suspends in your system and the time between suspends is less that 10 milliseconds and the total number of erase suspends can exceed 5000.

2. API Specification

2.1 Nomenclature, Arguments and Typedefs

Bank
A bank (flash bank) is like a separate device. Some Spansion devices have multiple banks, thus allowing for simultaneous read (in one bank), while programming (in another bank).

Cascade
Cascade is a term used to describe a multiple flash configuration where the additional flash devices are used to increase the number of addressable locations.

Command (Cmd)
Command refers to the software implementation of a specific data sheet command.

DYB
Dynamic protection Bit. Volatile protection bit for a sector.

Interleaved
Flash is said to be interleaved when identical multiple devices are used to match the data bus size of a processor. For example, two 16 bit devices are combined to match a 32 bits system bus.

LLD
Low Level Driver. The low level driver is the most basic set of flash functions.

Operation (Op)
An operation is defined as one or more commands combined to provide a more complete capability.

OTP
One Time Programmable. A memory area that can be programmed once and cannot be erased.

Page
The largest programmable unit for Write Buffered Programming. Pages are located on boundaries determined by the size of the page. For devices with 32 word write buffers, the page size is 32 words. In this case, pages start at addresses in which the lower five address bits are zero. Write Buffered Programming can only write to locations within a page.

PPB
Persistent Protection Bit. A non-volatile bit used to protect a sector or a sector group.

Word
Word is used to describe the smallest assessable unit of flash in your system. In a system with a single 16-bit flash, a word would be 16 bits (two bytes). In a system with four interleaved 8 bit flash devices, a word would be 32 bits (four bytes).

2.2 Arguments

base_addr
The base_addr is the starting address of the bank/device being manipulated.

offset
Offset is a measure of distance in words from the beginning of the device. For command cycles defined in the data sheet, it correlates to the “Addr” field.

2.3 Typdefs

ADDRESS

A variable type used in the code to hold addresses and offsets. Defined in lld.h.

DEVSTATUS

A variable type used in the code to describe the state of the flash. It is defined in lld.h.

```
typedef enum {
 DEV_STATUS_UNKNOWN = 0,
 DEV_NOT_BUSY,
 DEV_BUSY,
 DEV_EXCEEDED_TIME_LIMITS,
 DEV_SUSPEND,
 DEV_WRITE_BUFFER_ABORT,
 DEV_STATUS_GET_PROBLEM,
 DEV_VERIFY_ERROR,
 DEV_BYTES_PER_OP_WRONG,
DEV_SECTOR_LOCK,
DEV_PROGRAM_SUSPEND,
DEV_PROGRAM_SUSPEND_ERROR,
DEV_ERASE_SUSPEND,
DEV_ERASE_SUSPEND_ERROR,
DEV_BUSY_IN_OTHER_BANK
} DEVSTATUS;
```

FLASHDATA

A variable type used in the code to hold the smallest unit of data in your system. Its size is determined by the macro LLD_CONFIGURATION (in lld_target_specific.h. FLASHDATA is defined in lld.h).

POLLING_TYPE

POLLING_TYPE is a type of variable used to identify the operation to the polling routine.

```
typedef enum
{
LLD_P_POLL_PGM = 1,
LLD_P_POLL_WRT_BUF_PGM,
LLD_P_POLL_SEC_ERS,
LLD_P_POLL_CHIP_ERS,
LLD_P_POLL_RESUME
}POLLING_TYPE;
```

2.4 Common APIs

Notice some of the APIs listed below share the same names, but with different parameters. For instance, lld_GetDeviceID() has two forms of parameter list. The first one requires a base address while the second one needs to pass both a base address and offset address. To decide which form of APIs to use, the users need to refer to the data sheet of the specific device or related documents for more details.

2.4.1 Basic Operations

The Command API is a set of functions common to all Spansion flash devices. As we mentioned earlier, there is basically a one to one correlation between Common API functions and the commands listed in the flash data sheet. This API consists of a set of basic functions (Basic Operations) and a set of building blocks (Basic Commands).

The Basic Operations are a set of functions that provide a level of operation one step above the Basic Commands. The operation performs the desired function and poll for completion. The return value is used to determine the status of the operation.

Most of the function names of the Basic Operations end with “Op”.

Note: In systems that cannot wait for programming or erasing to finish, you will need to either implement another solution or develop non-blocking code based on our Basic Commands. In LLD, two support functions, DelayMilliseconds() and DelayMicroseconds(), have been implemented as examples. Users need to re-examine or re-implemented the functions based on their own particular platforms so more accurate time delay can be achieved.

2.4.1.1 lld_GetVersion

Description:

This command is used to return LLD version number.

Returns: Version number returned in given array.

Parameters:

Name	Type	Description
versionStr[]	LLD_CHAR *	Empty char array for receiving LLD version number.

Behavior:

Note: The size of the given char array has to be at least 9 in order to avoid buffer overflow.

Related Commands: n/a

Example Code:

```
LLD_ChAR versionStr[9];
lld_GetVersion(versionStr);
printf(" LLD Release Version: %s", versionStr);
```

2.4.1.2 lld_Poll – Using DQ Toggling

Description:

This function is used to poll the status of the flash after program and erase operations. In the event of a device error, this function will record the error, reset the flash (software reset) and return the status to the caller.

Returns: DEVSTATUS

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.
Offset	ADDRESS	Offset to the location being manipulated.
exp_data_ptr	FLASHDATA *	A pointer to a variable containing the expected data.
act_data_ptr	FLASHDATA *	A pointer to a variable to store the actual data.
polling_type	POLLING_TYPE	An indication of the type of operation being performed.

Behavior:

This function will continue to poll until the operation completes or an error is detected.

Related Commands: lld_StatusGet

Example Code:

```
lld_ProgramBufferToFlashCmd(base_addr, last_loaded_addr);

status = lld_Poll(base_addr, last_loaded_addr, &write_data,
                  &read_data, LLD_P_POLL_WRT_BUF_PGM);
return(status);
```

2.4.1.3 lld_Poll – Using Status Register

Description:

This function is used to poll the status of the flash after program and erase operations. It will return the value of the status register. The caller routine need to check the status register bit to determine the operation result is succeed or failed.

Returns: DEVSTATUS

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.
Offset	ADDRESS	Offset to the location being manipulated.

Behavior:

This function will continue to poll until the operation completes or an error is detected.

Related Commands:

Example Code:

```
lld_ProgramBufferToFlashCmd(base_addr, offset);

status_reg = lld_Poll(base_addr, offset);
```

2.4.1.4 lld_StatusGetReg

Description:

This function writes the status register read command sequence to flash and reads the current value of the status register.

Returns: value of status register

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.
offset	ADDRESS	Offset to the location being manipulated.

Behavior:

n/a

Related Commands: lld_StatusGetReg

Example Code:

```
lld_StatusGetReg (base_addr, offset);
```

2.4.1.5 lld_StatusGet

Description:

Unlike lld_Poll, lld_StatusGet tests the status and returns immediately. This function would be a good choice in situations where non-blocking functions were required.

Returns: DEVSTATUS

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being manipulated.
offset	ADDRESS	An index into the flash of the location to program.

Behavior:

n/a

Related Commands: lld_Poll

Example Code:

```
do
{
    dev_status = lld_StatusGet(base_addr, offset);
}
while(dev_status == DEV_BUSY);
```

2.4.1.6 lld_StatusClear (CMD1)

Description:

This function clears the status register of the flash.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.

Behavior:

n/a

Related Commands: lld_StatusGetReg

Example Code:

```
lld_StatusClear (base_addr);
```

2.4.1.7 lld_StatusClear (CMD2)

Description:

This function clears the status register of the flash.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.
offset	ADDRESS	Offset to the location being manipulated.

Behavior:

n/a

Related Commands: lld_StatusGetReg

Example Code:

```
lld_StatusClear (base_addr, offset);
```

2.4.1.8 lld_ProgramOp

Description:

This function programs a single word in flash and poll the status for completion.

Returns: DEVSTATUS

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being manipulated.
offset	ADDRESS	An index into the flash of the location to program.
write_data	FLASHDATA	The value to program into flash.

Behavior:

Program suspend will not work with this function, since this function will not return until the process is finished. Returns the device to read array mode.

Related Commands: lld_ProgramCmd, lld_Poll

Example Code:

```
addr       = (FLASHDATA *)   strtoul(argv[1], 0, input_radix);
offset     = (ADDRESS)    strtoul(argv[2], 0, input_radix);
write_data = (FLASHDATA) strtoul(argv[3], 0, input_radix);

status = lld_ProgramOp(addr, offset, write_data);
printf("status = %s\n", get_status_str(status));
```

2.4.1.9 lld_WriteBufferProgramOp

Description:

This function programs words in the specified flash page and polls status for completion.

Returns: DEVSTATUS

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being programmed.
offset	ADDRESS	An index into the flash page.
word_count	WORDCOUNT	Number of words (not bytes) to program.
data_buf	FLASHDATA *	Pointer to the data to program to flash.

Behavior:

You must be familiar enough with your platform to know what the page boundaries and page sizes are for this function. Page sizes are based on the maximum number of words that can be written in Write Buffered Programming (check the data sheet) and based on your architecture's flash interleaving (check with the designer/schematics).

Restrictions: Each Write Buffered Programming operation can only write data within a single page and can only write a maximum of LLD_BUFFER_SIZE words.

Program suspend will not work with this function, since this function will not return until the process is finished.

Related Commands: lld_WriteToBufferCmd, lld_ProgramBufferToFlashCmd

Example Code:

```
addr     = (FLASHDATA *)   strtoul(argv[1], 0, input_radix);
offset   = (ADDRESS)   strtoul(argv[2], 0, input_radix);
word_cnt = (WORDCOUNT) strtoul(argv[3], 0, input_radix);
source   = (ADDRESS)   strtoul(argv[4], 0, input_radix);

status = lld_WriteBufferProgramOp(addr, offset, word_cnt,
                                    (FLASHDATA *)source);
printf("status = %s\n", get_status_str(status));
```

2.4.1.10 lld_ChipEraseOp

Description:

This function erases the entire chip and polls for completion. In the case of interleaved devices, all are erased.

Returns: DEVSTATUS

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be erased.

Behavior:

Returns the device to read array mode.

Note: This command takes a long time (minutes) to complete!

Related Commands: lld_ChipEraseCmd, lld_Poll

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

status = lld_ChipEraseOp(addr);
printf("status = %s\n", get_status_str(status));
```

2.4.1.11 lld_SectorEraseOp

Description:

This command erases the specified sector and waits for the process to end.

Returns: DEVSTATUS

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being erased.
Offset	ADDRESS	An index into the flash sector to be erased.

Behavior:

This command takes some time to complete (seconds).

Related Commands: lld_SectorEraseCmd, lld_Poll

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
offset = (ADDRESS) strtoul(argv[2], 0, input_radix);

status = lld_SectorEraseOp(addr, offset);
printf("status = %s\n", get_status_str(status));
```

2.4.1.12 lld_ReadOp

Description:

This function reads the specified word.

Since the flash is usually memory mapped, you can read it without any special commands (its just memory). However, by funneling all the reads through this function a more consistent code base is developed. Also, some higher-level Spansion layers may require it.

Returns: FLASHDATA (Word read)

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being read.
offset	ADDRESS	An index to the location to be read.

Behavior:

No special behavior.

Related Commands: n/a

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
offset = (ADDRESS) strtoul(argv[2], 0, input_radix);

data_read = lld_ReadOp(addr, offset);
printf("%8.8X\n", data_read);
```

2.4.1.13 lld_EraseSuspendOp (CMD1)

Description:

This function suspends the erase operation. The erase resume command will resume the erase operation.

Returns: DEVSTATUS

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be erased.

Behavior:

The device will be in erase suspend mode.

Related Commands: lld_ProgramSuspendCmd, lld_EraseResumeCmd, lld_Poll

Example Code:

```
status = lld_EraseSuspendOp(base_addr);
printf("status = %s\n", get_status_str(status));
```

2.4.1.14 lld_EraseSuspendOp (CMD2)

Description:

This function suspends the erase operation. The erase resume command will resume the erase operation.

Returns: DEVSTATUS

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be erased.
offset	ADDRESS	An index to the flash sector to be suspended.

Behavior:

The device will be in erase suspend mode.

Related Commands: lld_ProgramSuspendCmd, lld_EraseResumeCmd, lld_Poll

Example Code:

```
  status = lld_EraseSuspendOp( base_addr, offset);
  printf ( "status = %s\n", get_status_str(status) );
```

2.4.1.15 lld_ProgramSuspendOp (CMD1)

Description:

This function suspends the program operation. The program resume command will resume the program operation.

Returns: DEVSTATUS

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be suspended.

Behavior:

The device will be in program suspend mode.

Related Commands: lld_EraseSuspendCmd, lld_ProgramResumeCmd, lld_Poll

Example Code:

```
status = lld_ProgramSuspendOp(base_addr);
printf("status = %s\n", get_status_str(status));
```

2.4.1.16 lld_ProgramSuspendOp (CMD2)

Description:

This function suspends the program operation. The program resume command will resume the program operation.

Returns: DEVSTATUS

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be checked.
offset	ADDRESS	An index to the flash sector to be suspended.

Behavior:

The device will be in program suspend mode.

Related Commands: lld_ProgramSuspendCmd, lld_EraseResumeCmd, lld_Poll

Example Code:

```
status = lld_ProgramSuspendOp( base_addr, offset);
printf ( "status = %s\n", get_status_str(status) );
```

2.4.1.17 lld_BlankCheckOp

Description:

This function checks the if specified sector is blank.

Returns: DEVSTATUS

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being erased.
offset	ADDRESS	An index to the flash sector to be checked.

Behavior:

This command takes some time to complete.

Related Commands: lld_SectorEraseCmd, lld_Poll

Example Code:

```
status = lld_BlankCheckOp(base_addr, offset);
printf("status = %s\n", get_status_str(status));
```

2.4.1.18 lld_memcpy

Description:

The lld_memcpy function was added to simplify Write Buffer Programming. It is used to program memory like the lld_WriteBufferProgramOp, but the caller does not have to understand flash page sizes, boundaries, etc.

Returns: DEVSTATUS

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being programmed.
Offset	ADDRESS	An index into the flash of the first location to be programmed.
word_cnt	WORDCOUNT	Number of words to program.
data_buf	FLASHDATA *	The location of the source data.

Behavior:

The device is put into read array mode when finished. This can take a long time when there is a great deal of data. This command cannot span banks/devices.

Related Commands: lld_WriteBufferProgramOp

Example Code:

```
addr       = (FLASHDATA *)     strtoul(argv[1], 0, input_radix);
offset     = (ADDRESS)     strtoul(argv[2], 0, input_radix);
word_count = (WORDCOUNT)   strtoul(argv[3], 0, input_radix);
source     = (FLASHDATA *) strtoul(argv[4], 0, input_radix);

status = lld_memcpy(addr, offset, word_count, source);
printf("status = %s\n", get_status_str(status));
```

2.4.1.19 lld_GetDeviceId

Description:

This function reads the device ID from CFI region.

Returns: unsigned int   deviceID

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being read.

Behavior:

No special behavior.

Related Commands: n/a

Example Code:

```
  data_read = lld_GetDeviceId(base_addr );
  printf("%8.8X\n", data_read);
```

2.4.1.20 lld_GetDeviceId (Device with Address Space Overlay Mode)

Description:

This function reads the device ID from CFI region.

Returns: unsigned int deviceID

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being read.
offset	ADDRESS	Specify sector offset for ASO (Address Space Overlay).

Behavior:

No special behavior.

Related Commands: n/a

Example Code:

```
data_read = lld_GetDeviceId(base_addr, offset);
printf("%8.8X\n", data_read);
```

2.4.2 Basic Commands

2.4.2.1 lld_ResetCmd

Description:

This command is used to return the flash to the read array mode. It is normally not necessary after programming or erase, since the flash returns to read array mode automatically when there are no problems. However, if a program or erase operation encounters an error, it will be necessary to issue an lld_ResetCmd to return the device to read array mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being reset.

Behavior:

Note: The flash Reset command, implemented by lld_ResetCmd, does not invoke a hardware reset of the flash.

Related Commands: n/a

Example Code:

```
FLASH_WR(base_addr, LLD_UNLOCK_ADDR1, NOR_CFI_QUERY_CMD); /*CFI mode*/
data  = FLASH_RD(base_addr, offset);  /* Read CFI data */
lld_ResetCmd(base_addr); /* return flash to read array mode */
return(data);
```

2.4.2.2 lld_ProgramCmd

Description:

This command is used to program a single word.

Note: On devices that support Write Buffer Programming, you are expected to use Write Buffered Programming. It is possible that future Spansion flash devices will not support the data sheet Program command. If you are developing code to run on future parts AND the current part supports Write Buffered Programming, you should program the flash with the Write Buffered Program commands.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being programmed.
offset	ADDRESS	An index into the flash that correlates to the flash array element to be programmed.
pgm_data_ptr	FLASHDATA *	A pointer to the data to be used for programming.

Behavior:

When issued, this command will begin the programming process. The flash will no longer be in read array mode during programming. Typically, this command is followed by a status polling routine to determine the state of the flash.

Related Commands: lld_Poll, lld_StatusGet

Example Code:

```
lld_ProgramCmd(base_addr, offset, &write_data);
```

2.4.2.3 lld_WriteToBufferCmd

Description:

This command is used to start the Write Buffered Program sequence. It must be followed by other commands to perform Write Buffered Programming.

Note: Write Buffered Programming is faster than the legacy lld_ProgramCmd programming method, and it is the recommended way to program flash in devices that support this feature.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being programmed.
offset	ADDRESS	An index into the flash that correlates to the Addr. field in the Command Table of the data sheet.

Behavior:

Write Buffered Programming is more complicated to use than the older lld_ProgramCmd. Make sure you read the data sheet section on Write Buffered Programming before coding. The lld_WriteBufferProgramOp or the lld_memcpy might be easier, since they are complete implementations of Write Buffered Programming.

Restrictions: Each Write Buffered Programming operation can only write data within a single page and can only write a maximum of LLD_BUFFER_SIZE words.

Related Commands: lld_ProgramBufferToFlashCmd, lld_WriteBufferProgramOp, lld_memcpy, lld_Poll, lld_StatusGet

Example Code:

```
/* Issue Load Write Buffer Command Sequence */
lld_WriteToBufferCmd(base_addr, offset);

/* Write # of locations to program */
wcount *= LLD_DEV_MULTIPLIER;

FLASH_WR(base_addr, offset, wcount);
```

2.4.2.4 lld_ProgramBufferToFlashCmd

Description:

This command is used in conjunction with the lld_WriteToBufferCmd. It is the last command issued in the Write Buffer Programming sequence.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being programmed.
offset	ADDRESS	An index into the flash that correlates to the Addr. field in the Command Table of the data sheet.

Behavior:

The device will no longer be in read array mode during the execution of Write Buffered Programming.

Related Commands: lld_ProgramBufferToFlashCmd, lld_WriteBufferProgramOp, lld_memcpy, lld_Poll, lld_StatusGet

Example Code:

```
/* Issue Program Buffer to Flash command */
lld_ProgramBufferToFlashCmd(base_addr, last_loaded_addr);
```

2.4.2.5 lld_WriteBufferAbortResetCmd

Description:

This command is used to abort the Write Buffer Programming operation when DQ1 = 1 (Write Buffer Program Error) is encountered during polling.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being reset.

Behavior:

n/a

Related Commands: lld_ProgramBufferToFlashCmd, lld_WriteBufferProgramOp, lld_memcpy, lld_Poll, lld_StatusGet

Example Code:

```
if(dev_status != DEV_NOT_BUSY)
{
  if(dev_status == DEV_WRITE_BUFFER_ABORT)
  {
    lld_WriteBufferAbortResetCmd(base_addr);
  }
}
```

2.4.2.6 lld_ChipEraseCmd

Description:

The command begins the Chip Erase process. This can take quite a while. During this process the device is not in read array mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being erased.

Behavior:

The device is not in read array mode while the erase is in progress.

Related Commands: lld_SectorEraseCmd, lld_Poll, lldStatusGet

Example Code:

```
lld_ChipEraseCmd(base_addr);
```

2.4.2.7 lld_SectorEraseCmd

Description:

This command begins a sector erase process. In terms of CPU cycles, this command will take a little time. During that time, the device will not be in read array mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being erased.
offset	ADDRESS	An index into the sector to be erased.

Behavior:

The flash will not be in read array mode during this process. This process can be suspended.

Related Commands: lld_EraseSuspendCmd, lld_EraseResumeCmd, lld_Poll, lld_StatusGet, lld_SectorEraseOp

Example Code:

```
    lld_SectorEraseCmd(base_addr, offset);
```

2.4.2.8 lld_EraseSuspendCmd (CMD1)

Description:

This command is used to suspend the erase process. It is useful when reading/programming other sectors is necessary.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being suspended.
offset	ADDRESS	An index into the sector being erased.

Behavior:

Be sure to read the data sheet about this command.

Related Commands: lld_EraseResumeCmd, lld_SectorEraseCmd

Example Code:

```
    lld_EraseSuspendCmd(addr, offset);
```

2.4.2.9 lld_EraseSuspendCmd (CMD2)

Description:

This command is used to suspend the erase process. It is useful when reading/programming other sectors is necessary.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being suspended.

Behavior:

Be sure to read the data sheet about this command.

Related Commands: lld_EraseResumeCmd, lld_SectorEraseCmd

Example Code:

```
lld_EraseSuspendCmd(base_addr);
```

2.4.2.10 lld_EraseResumeCmd (CMD1)

Description:

This command resumes the erase process on a suspended sector.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being manipulated.
offset	ADDRESS	An index into the sector where the erase needs to be restarted.

Behavior:

The device will not be in read array mode any longer.

Related Commands: lld_EraseSuspendCmd, lld_SectorEraseCmd

Example Code:

```
lld_EraseSuspendCmd(addr, offset);
```

2.4.2.11 lld_EraseResumeCmd (CMD2)

Description:

This command resumes the erase process on a suspended sector.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being manipulated.

Behavior:

The device will not be in read array mode any longer.

Related Commands: lld_EraseSuspendCmd, lld_SectorEraseCmd

Example Code:

```
lld_EraseSuspendCmd( base_addr );
```

2.4.2.12 lld_ProgramSuspendCmd (CMD1)

Description:

This command is used to suspend the programming process. It is useful when reading/programming other sectors is necessary.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being suspended.
offset	ADDRESS	An index to the location being programmed.

Behavior:

Be sure to read the data sheet about this command.

Related Commands: lld_ProgramResumeCmd, lld_ProgramBufferToFlashCmd

Example Code:

```
lld_ProgramSuspendCmd(addr, offset);
```

2.4.2.13 lld_ProgramSuspendCmd (CMD2)

Description:

This command is used to suspend the programming process. It is useful when reading/programming other sectors is necessary.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being suspended.

Behavior:

Be sure to read the data sheet about this command.

Related Commands: lld_ProgramResumeCmd, lld_ProgramBufferToFlashCmd

Example Code:

```
  lld_ProgramSuspendCmd( base_addr );
```

2.4.2.14 lld_ProgramResumeCmd (CMD1)

Description:

This command resumes the programming process on the suspended location.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being manipulated.
offset	ADDRESS	An index into the location where the programming was occurring.

Behavior:

The device will not be in read array mode any longer.

Related Commands: lld_ProgramSuspendCmd, lld_ProgramBufferToFlashCmd

Example Code:

```
lld_ProgramSuspendCmd(addr, offset);
```

2.4.2.15 lld_ProgramResumeCmd (CMD2)

Description:

This command resumes the program process on the suspended location.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being manipulated.

Behavior:

The device will not be in read array mode any longer.

Related Commands: lld_ProgramSuspendCmd, lld_ProgramBufferToFlashCmd

Example Code:

```
lld_ProgramSuspendCmd(base_addr);
```

2.4.2.16 lld_StatusRegReadCmd (CMD1)

Description:

This command reads the current value of the status register.

Returns: value of status register

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being manipulated.

Behavior:

n/a.

Related Commands: lld_StatusRegClearCmd

Example Code:

```
lld_StatusRegReadCmd( base_addr);
```

2.4.2.17 lld_StatusRegReadCmd (CMD2)

Description:

This command reads the current value of the status register.

Returns: value of status register

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being manipulated.
offset	ADDRESS	The status read corresponds to the location specified by the offset.

Behavior:

n/a.

Related Commands: lld_StatusRegClearCmd

Example Code:

```
lld_StatusRegReadCmd(base_addr, offset);
```

2.4.2.18 lld_StatusRegClearCmd (CMD1)

Description:

This command clears the current value of the status register.

Returns: value of status register

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.

Behavior:

n/a.

Related Commands: lld_StatusRegReadCmd

Example Code:

```
status = lld_ProgramSuspendOp( base_addr, offset);
lld_StatusRegClearCmd( base_addr);
```

2.4.2.19 lld_StatusRegClearCmd (CMD2)

Description:

This command clears the current value of the status register.

Returns: value of status register

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being manipulated.
offset	ADDRESS	The status register is cleared for the offset specified.

Behavior:

n/a.

Related Commands: lld_StatusRegReadCmd

Example Code:

```
lld_StatusRegClearCmd(base_addr, offset);
```

2.4.2.20 lld_BlankCheckCmd

Description:

This command checks if a sector is blank or not.

Returns: value of status register

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being manipulated.
offset	ADDRESS	An index into the location where to do blank check.

Behavior:

Blank check can only be issued while in array mode not in program or erase suspend mode. Reads to the array while in blank check mode is not allowed and will return unknown data.

Related Commands: lld_SectorEraseCmd

Example Code:

```
lld_BlankCheckCmd(base_addr, offset);
```

2.5 CFI Query APIs

2.5.1 CFI Query Operation

2.5.1.1 lld_ReadCfiWord

Description:

This function reads the CFI data register

Returns: FLASHDATA (Word CFI Register)

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.
offset	ADDRESS	An index into the flash sector to be read.

Behavior:

n/a

Related Commands: lld_CfiEntryCmd, lld_CfiExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
offset = (ADDRESS) strtoul(argv[2], 0, input_radix);

data = lld_ReadCfiWord(base_addr, offset);
printf("%8.8X\n", data);
```

2.5.2 CFI Query Commands

2.5.2.1 lld_CfiEntryCmd

Description:

This command causes the CFI data to be available in the first sector of the specified bank.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.

Behavior:

The first sector of the device specified in the base_addr parameter will be replaced with the CFI data. Use the lld_CfiExitCmd to return to read array mode.

Related Commands: lld_CfiExitCmd

Example Code:

```
lld_CfiEntryCmd(addr);
```

2.5.2.2 lld_CfiEntryCmd (Device with Address Space Overlay Mode)

Description:

This command causes the CFI data to be available in the specified sector of the specified bank.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.
offset	ADDRESS	Sector offset for ASO (Address Space Overlay).

Behavior:

The first sector of the device specified in the base_addr parameter will be replaced with the CFI data. Use the lld_CfiExitCmd to return to read array mode.

Related Commands: lld_CfiExitCmd

Example Code:

```
lld_CfiEntryCmd(base_addr, offset);
```

2.5.2.3 lld_CfiExitCmd

Description:

This command exits CFI mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.

Behavior:

Returns the device to read array mode.

Related Commands: lld_CfiEntryCmd

Example Code:

```
lld_CfiExitCmd(addr);
```

2.6 Autoselect APIs

2.6.1 Autoselect Commands

2.6.1.1 lld_AutoselectEntryCmd

Description:

This command replaces the first sector with the Autoselect information.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.

Behavior:

The device will no longer be in read array mode.

Related Commands: lld_AutoselectExitCmd

Example Code:

```
lld_AutoselectEntryCmd(addr);
```

2.6.1.2 lld_AutoselectEntryCmd (Device with Address Space Overlay Mode)

Description:

This command replaces the specified sector with the Autoselect information.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.
offset	ADDRESS	Sector offset for ASO (Address Space Overlay).

Behavior:

The device will no longer be in read array mode.

Related Commands: lld_AutoselectExitCmd

Example Code:

```
lld_AutoselectEntryCmd(addr, offset);
```

2.6.1.3 lld_AutoselectExitCmd

Description:

This command returns the device/bank to read array mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.

Behavior:

Returns the device to read array mode.

Related Commands: lld_AutoselectEntryCmd

Example Code:

```
lld_AutoselectExitCmd(addr);
```

2.7 Unlock Bypass APIs

2.7.1 Unlock Bypass Commands

2.7.1.1 lld_UnlockBypassEntryCmd

Description:

This command puts the flash state machine in a mode where it will accept minimum cycle commands.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.

Behavior:

Use only Unlock Bypass Commands while in this mode.

Related Commands: lld_UnlockBypassExitCmd, lld_UnlockBypassProgramCmd

Example Code:

```
addr   = (ADDRESS) strtoul(argv[1], 0, input_radix);

lld_UnlockBypassEntryCmd(addr);
```

2.7.1.2 lld_UnlockBypassProgramCmd

Description:

This command is a faster version of the lld_ProgramCmd.

Note: Like lld_ProgramCmd, lld_UnlockBypassProgramCmd should not be used in systems that provide Write Buffer Programming.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be programmed.
offset	ADDRESS	An index to the location to be programmed.
pgm_data_ptr	FLASHDATA *	Pointer to the data to program.

Behavior:

This command can only be used in Unlock Bypass mode.

Related Commands: lld_UnlockBypassEntryCmd, lld_UnlockBypassExitCmd

Example Code:

```
addr   = (FLASHDATA *)   strtoul(argv[1], 0, input_radix);
offset = (ADDRESS)    strtoul(argv[2], 0, input_radix);
data   = (FLASHDATA) strtoul(argv[3], 0, input_radix);

lld_UnlockBypassProgramCmd(addr, offset, &data);
```

2.7.1.3 lld_UnlockBypassResetCmd

Description:

This command returns the flash state machine to the standard (non-Unlock Bypass) mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.

Behavior:

n/a

Related Commands: lld_UnlockBypassEntryCmd, lld_UnlockBypassProgramCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
```

2.8 Sector Protection APIs

2.8.1 SecSi Sector Commands

Upon SecSiSectorEntryCmd, the first sector is replaced by the SecSi sector. Once in this mode, use the standard programming and reading commands.

2.8.1.1 lld_SecSiSectorEntryCmd

Description:

This command grants access to the SecSi sector.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.

Behavior:

You should read and understand the data sheet section on the SecSi sector before writing to this OTP area.

Related Commands: lld_SecSiSectorExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_SecSiSectorEntryCmd(addr);
```

2.8.1.2 lld_SecSiSectorEntryCmd (Device with Address Space Overlay Mode)

Description:

This command grants access to the SecSi sector.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.
offset	ADDRESS	sector offset for ASO (Address Space Overlay).

Behavior:

You should read and understand the data sheet section on the SecSi sector before writing to this OTP area.

Related Commands: lld_SecSiSectorExitCmd

Example Code:

```
lld_SecSiSectorEntryCmd(base_addr, offset);
```

2.8.1.3 lld_SecSiSectorExitCmd

Description:

This command restores the first sector with read array data (from SecSi sector data)

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.

Behavior:

n/a

Related Commands: lld_SecSiSectorEntryCmd

Example Code:

```
lld_SecSiSectorExitCmd(base_addr);
```

2.8.2 Lock Register Operations

2.8.2.1 lld_LockRegBitsReadOp

Description:

This function returns the value of Lock Register.

Returns: FLASHDATA (Lock Register Word)

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.

Behavior:

n/a

Related Commands: lld_LockRegBitsProgramCmd, lld_LockRegEntryCmd, lld_LockRegExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

data = lld_LockRegBitsReadOp(addr);
printf("%8.8X\n", data);
```

2.8.2.2 lld_SSRLockRegBitsReadOp

Description:

This function reads the Secure Silicon Region (SSR) lock register.

Returns: FLASHDATA (Word Lock Register)

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being read.
offset	ADDRESS	An index into the flash sector to be read.

Behavior:

n/a

Related Commands: lld_SSRLockRegEntryCmd, lld_SSRLockRegExitCmd, lld_SSRLockRegBitsProgramCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
offset = (ADDRESS) strtoul(argv[2], 0, input_radix);

data = lld_SSRLockRegBitsReadOp(base_addr, offset);
printf("%8.8X\n", data);
```

2.8.2.3 lld_LockRegBitsProgramOp

Description:

This function programs the Lock Register with a value. Refer to the data sheet for bit definitions.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.
value	FLASHDATA	The Lock Register value to be programmed.

Behavior:

n/a

Related Commands: lld_LockRegEntryCmd, lld_LockRegBitsReadCmd, lld_LockRegExitCmd

Example Code:

```
addr   = (FLASHDATA *)   strtoul(argv[1], 0, input_radix);
value  = (FLASHDATA) strtoul(argv[2], 0, input_radix);

lld_LockRegBitsProgramOp(addr, value);
```

2.8.2.4 lld_SSRLockRegBitsProgramOp

Description:

This function programs the Secure Silicon Region (SSR) lock register.

Returns: DEVSTATUS (Program Complete, Program Error)

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the bank/device being programmed.
offset	ADDRESS	An index into the flash sector to be programmed.
write_data	FLASHDATA	The value to program into flash.

Behavior:

n/a

Related Commands: lld_SSRLockRegEntryCmd, lld_SSRLockRegExitCmd, lld_LockRegBitsReadCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
offset = (ADDRESS) strtoul(argv[2], 0, input_radix);
write_data = (FLASHDATA) strtoul(argv[3], 0, input_radix);

status = lld_SSRLockRegBitsProgramOp(addr, offset, write_data);
printf("status = %s\n", get_status_str(status));
```

2.8.2.5 lld_PpbLockBitReadOp

Description:

This function reads the value of the PPB Lock Bit.

Returns: FLASHDATA (0=PPB Protection selected, 1=not selected)

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.

Behavior:

n/a

Behavior:

n/a

Related Commands: lld_PpbLockBitEntryCmd, lld_PpbLockBitExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

data = lld_PpbLockBitReadOp(addr);
printf("%8.8X\n", data);
```

2.8.2.6 lld_PpbLockBitSetOp

Description:

This function sets the Flash Protection Mode to PPB Mode (as opposed to Password Protection Mode).

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.

Behavior:

n/a

Related Commands: lld_PpbLockBitEntryCmd, lld_PpbLockBitExitCmd, lld_Poll

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_PpbLockBitSetOp(addr);
```

2.8.2.7 lld_PpbAllEraseOp

Description:

This function un-protects all the PPB bits for sectors/sector groups.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.

Behavior:

All the PPB bits are erased at once.

Related Commands: lld_PpbEntryCmd, lld_PpbExitCmd, lld_Poll

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_PpbAllEraseOp(addr);
```

2.8.2.8 lld_PpbProgramOp

Description:

This function sets the PPB protection for a sector/sector group. When set, the PPB Status Read will return 0 (protected), otherwise it will return 1 (unprotected).

Returns: DEVSTATUS (Program Complete, Program Error)

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.
offset	ADDRESS	An index to the flash sector to be protected.

Behavior:

n/a

Related Commands: lld_PpbEntryCmd, lld_PpbExitCmd, lld_PpbLockBitReadOp, lld_Poll

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
offset = (ADDRESS) strtoul(argv[2], 0, input_radix);

lld_PpbProgramOp(addr, offset);
```

2.8.2.9 lld_PpbStatusReadOp

Description:

This function reads the status of the PPB Protection Bit for the addressed sector/sector group.

Returns: FLASHDATA (0 = protected, 1 = unprotected)

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.
offset	ADDRESS	An index to the sector/sector group.

Behavior:

n/a

Related Commands: lld_PpbEntryCmd, lld_PpbExitCmd, lld_PpbAllEraseCmd, lld_PpbProgramCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
offset = (ADDRESS) strtoul(argv[2], 0, input_radix);

data = lld_PpbStatusReadOp(addr, offset);
printf("%8.8X\n", data);
```

2.8.3 Lock Register Commands

2.8.3.1 lld_LockRegEntryCmd

Description:

This mode of operation is used to read and program the Lock Register Bits. Non-Lock Register commands should not be used while in this mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.

Behavior:

n/a

Related Commands: lld_LockRegBitsProgramCmd, lld_LockRegBitsReadCmd, lld_LockRegExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_LockRegEntryCmd(addr);
```

2.8.3.2 lld_LockReg2EntryCmd

Description:

This mode of operation is used to read and program the Lock Register Bits. Non-Lock Register commands should not be used while in this mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.

Behavior:

n/a

Related Commands: lld_LockRegBitsProgramCmd, lld_LockRegBitsReadCmd, lld_LockRegExitCmd

Example Code:

```
lld_LockReg2EntryCmd (base_addr);
```

2.8.3.3 lld_SSRLockRegEntryCmd (Device with Address Space Overlay Mode)

Description:

This mode of operation is used to read and program the Lock Register Bits.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.
Offset	ADDRESS	sector offset for ASO (Address Space Overlay).

Behavior:

n/a

Related Commands: lld_SSRLockRegBitsProgramCmd, lld_SSRLockRegBitsReadCmd, lld_SSRLockRegExitCmd

Example Code:

```
lld_SSRLockRegEntryCmd(base_addr, offset);
```

2.8.3.4 lld_LockRegBitsProgramCmd

Description:

Programs the Lock Register with a value. Refer to the data sheet for bit definitions.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.
Value	FLASHDATA	The Lock Register value to be programmed.

Behavior:

n/a

Related Commands: lld_LockRegEntryCmd, lld_LockRegBitsReadCmd, lld_LockRegExitCmd

Example Code:

```
addr   = (FLASHDATA *)   strtoul(argv[1], 0, input_radix);
value  = (FLASHDATA) strtoul(argv[2], 0, input_radix);

lld_LockRegBitsProgramCmd(addr, value);
```

2.8.3.5 lld_SSRLockRegBitsProgramCmd (Device with Address Space Overlay Mode)

Description:

Programs the Lock Register with a value. Refer to the data sheet for bit definitions.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.
offset	ADDRESS	sector offset for ASO (Address Space Overlay).
value	FLASHDATA	The Lock Register value to be programmed.

Behavior:

n/a

Related Commands: lld_SSRLockRegEntryCmd, lld_SSRLockRegBitsReadCmd, lld_SSRLockRegExitCmd

Example Code:

```
lld_SSRLockRegBitsProgramCmd(base_addr, offset,  value);
```

2.8.3.6 lld_LockRegBitsReadCmd

Description:

This command returns the value of the Lock Register.

Returns: FLASHDATA (Lock Register Word)

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.

Behavior:

n/a

Related Commands: lld_LockRegBitsProgramCmd, lld_LockRegEntryCmd, lld_LockRegExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

data = lld_LockRegBitsReadCmd(addr);
printf("%8.8X\n", data);
```

2.8.3.7 lld_SSRLockRegBitsReadCmd (Device with Address Space Overlay Mode)

Description:

This command returns the value of the Lock Register.

Returns: FLASHDATA (Lock Register Word)

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.
offset	ADDRESS	sector offset for ASO (Address Space Overlay).

Behavior:

n/a

Related Commands: lld_SSRLockRegBitsProgramCmd, lld_SSRLockRegEntryCmd, lld_SSRLockRegExitCmd

Example Code:

```
data = lld_SSRLockRegBitsReadCmd(base_addr, offset);
  printf("%8.8X\n", data);
```

2.8.3.8 lld_LockRegExitCmd

Description:

This command exits the Lock Register mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.

Behavior:

n/a

Related Commands: lld_LockRegBitsProgramCmd, lld_LockRegBitsReadCmd, lld_LockRegEntryCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_LockRegExitCmd(addr);
```

2.8.3.9 lld_SSRLockRegExitCmd (Device with Address Space Overlay Mode)

Description:

This command exits the SSR Lock Register mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.

Behavior: n/a

Related Commands: lld_SSRLockRegBitsProgramCmd, lld_SSRLockRegBitsReadCmd, lld_SSRLockRegEntryCmd,

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_SSRLockRegExitCmd(addr);
```

2.8.4 Password Protection Mode Commands

2.8.4.1 lld_PasswordProtectionEntryCmd

Description:

This command puts the state machine in Password Protection Modification mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.

Behavior:

Only Password Protection Mode commands should be issued while in this mode.

Related Commands: lld_PasswordProtectionProgramCmd, lld_PasswordProtectionReadCmd, lld_PasswordProtectionUnlockCmd, lld_PasswordProtectionExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_PasswordProtectionEntryCmd(addr);
```

2.8.4.2 lld_PasswordProtectionProgramCmd

Description:

This command is used to program the password once the device is in the Password Protection Modification mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.
offset	ADDRESS	The offset of the password to write. There are four separate passwords (offsets 0 - 3).
pwd	FLASHDATA	Password word.

Behavior:

You should read the data sheet about this feature, because this is OTP memory - you only get one chance.

Related Commands: lld_PasswordProtectionEntryCmd, lld_PasswordProtectionReadCmd, lld_PasswordProtectionUnlockCmd, lld_PasswordProtectionExitCmd

Example Code:

```
addr    = (FLASHDATA *)   strtoul(argv[1], 0, input_radix);
pwd1  = (FLASHDATA) strtoul(argv[2], 0, input_radix);
pwd2  = (FLASHDATA) strtoul(argv[3], 0, input_radix);
pwd3  = (FLASHDATA) strtoul(argv[4], 0, input_radix);
pwd4  = (FLASHDATA) strtoul(argv[5], 0, input_radix);

lld_PasswordProtectionProgramCmd(addr, 0, pwd1);
lld_PasswordProtectionProgramCmd(addr, 1, pwd2);
lld_PasswordProtectionProgramCmd(addr, 2, pwd3);
lld_PasswordProtectionProgramCmd(addr, 3, pwd4);
```

2.8.4.3 lld_PasswordProtectionReadCmd

Description:

This function issues the read password command.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.
pwd0	FLASHDATA *	A pointer to a location to store the first password word.
pwd1	FLASHDATA *	A pointer to a location to store the second password word.
pwd2	FLASHDATA *	A pointer to a location to store the third password word.
pwd3	FLASHDATA *	A pointer to a location to store the forth password word.

Behavior:

This command will not return the password after Password Protection mode is committed.

Related Commands: lld_PasswordProtectionProgramCmd, lld_PasswordProtectionEntryCmd, lld_PasswordProtectionUnlockCmd, lld_PasswordProtectionExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_PasswordProtectionReadCmd(addr, &pwd0, &pwd1, &pwd2, &pwd3);
printf("%8.8X %8.8X %8.8X %8.8X\n", pwd0, pwd1, pwd2, pwd3);
```

2.8.4.4 lld_PasswordProtectionUnlockCmd

Description:

This command presents the password to the flash. There is no indication of success.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.
pwd0	FLASHDATA	The first word of the password.
pwd1	FLASHDATA	The second word of the password.
pwd2	FLASHDATA	The third word of the password.
pwd3	FLASHDATA	The forth word of the password.

Behavior:

n/a

Related Commands: lld_PasswordProtectionProgramCmd, lld_PasswordProtectionReadCmd, lld_PasswordProtectionEntryCmd, lld_PasswordProtectionExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
pwd0   = (FLASHDATA) strtoul(argv[2], 0, input_radix);
pwd1   = (FLASHDATA) strtoul(argv[3], 0, input_radix);
pwd2   = (FLASHDATA) strtoul(argv[4], 0, input_radix);
pwd3   = (FLASHDATA) strtoul(argv[5], 0, input_radix);

lld_PasswordProtectionUnlockCmd(addr, pwd0, pwd1, pwd2, pwd3);
```

2.8.4.5 lld_PasswordProtectionExitCmd

Description:

This command exits the Password Protection Manipulation Mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.

Behavior:

Resets the device to read array mode.

Related Commands: lld_PasswordProtectionProgramCmd, lld_PasswordProtectionReadCmd, lld_PasswordProtectionUnlockCmd, lld_PasswordProtectionEntryCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_PasswordProtectionExitCmd(addr);
```

2.8.5 PPB Commands

2.8.5.1 lld_PpbEntryCmd

Description:

This command put the flash into PPB Command Set Mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.

Behavior:

n/a

Related Commands: lld_PpbStatusReadCmd, lld_PpbExitCmd, lld_PpbAllEraseCmd, lld_PpbProgramCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_PpbEntryCmd(addr);
```

2.8.5.2 lld_PpbProgramCmd

Description:

This command sets the PPB protection for a sector/sector group. When set, the PPB Status Read will return 0 (protected), otherwise it will return 1 (unprotected).

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.
offset	ADDRESS	An index to the flash sector to be protected.

Behavior:

n/a

Related Commands: lld_PpbEntryCmd, lld_PpbExitCmd, lld_PpbAllEraseCmd, lld_PpbStatusReadCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
offset = (ADDRESS) strtoul(argv[2], 0, input_radix);

lld_PpbProgramCmd(addr, offset);
```

2.8.5.3 lld_PpbAllEraseCmd

Description:

This command un-protects all the PPB bits for sectors/sector groups.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.

Behavior:

All the PPB bits are erased at once.

Related Commands: lld_PpbEntryCmd, lld_PpbExitCmd, lld_PpbStatusReadCmd, lld_PpbProgramCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_PpbAllEraseCmd(addr);?
```

2.8.5.4 lld_PpbStatusReadCmd

Description:

Reads the status of the PPB Protection Bit for the addressed sector/sector group.

Returns: FLASHDATA (0 = protected, 1 = unprotected)

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.
offset	ADDRESS	An index to the sector/sector group.

Behavior:

n/a

Related Commands: lld_PpbEntryCmd, lld_PpbExitCmd, lld_PpbAllEraseCmd, lld_PpbProgramCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
offset = (ADDRESS) strtoul(argv[2], 0, input_radix);

data = lld_PpbStatusReadCmd(addr, offset);
printf("%8.8X\n", data);
```

2.8.5.5 lld_PpbExitCmd

Description:

This command exits the PPB Command Set Mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.

Behavior:

n/a

Related Commands: lld_PpbEntryCmd, lld_PpbStatusReadCmd, lld_PpbAllEraseCmd, lld_PpbProgramCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_PpbExitCmd(addr);
```

2.8.6 DYB Commands

2.8.6.1 lld_DybEntryCmd

Description:

This command enters the DYB Protection Command Set Mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.

Behavior:

n/a

Related Commands: lld_DybSetCmd, lld_DybClrCmd, lld_DybReadCmd, lld_DybExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_DybEntryCmd(addr);
```

2.8.6.2 lld_DybSetCmd

Description:

This command sets the DYB to 0 (protected).

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.
offset	ADDRESS	An index to the sector to be protected.

Behavior:

n/a

Related Commands: lld_DybEntryCmd, lld_DybClrCmd, lld_DybReadCmd, lld_DybExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
offset = (ADDRESS) strtoul(argv[2], 0, input_radix);

lld_DybSetCmd(addr, offset);
```

2.8.6.3 lld_DybClrCmd

Description:

This command un-protects the appropriate DYB.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.
offset	ADDRESS	An index to the sector to be un-protected.

Behavior:

n/a

Related Commands: lld_DybEntryCmd, lld_DybSetCmd, lld_DybReadCmd, lld_DybExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
offset = (ADDRESS) strtoul(argv[2], 0, input_radix);

lld_DybClrCmd(addr, offset);
```

2.8.6.4 lld_DybReadCmd

Description:

This command reads the value of the sector's DYB bit.

Returns: FLASHDATA (0=protected, 1=un-protected)

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.
offset	ADDRESS	An index to the appropriate sector.

Behavior:

n/a

Related Commands: lld_DybEntryCmd, lld_DybSetCmd, lld_DybClrCmd, lld_DybExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);
offset = (ADDRESS) strtoul(argv[2], 0, input_radix);

data   = lld_DybReadCmd(addr, offset);
printf("%8.8X\n", data);
```

2.8.6.5 lld_DybExitCmd

Description:

This commands exits the DYB Protection Command Set Mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank.

Behavior:

n/a

Related Commands: lld_DybEntryCmd, lld_DybSetCmd, lld_DybClrCmd, lld_DybReadCmd

Example Code:

```
addr    = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_DybExitCmd(addr);
```

2.8.7 PPB Lock Bit Commands

2.8.7.1 lld_PpbLockBitEntryCmd

Description:

This command enters the PPB Lock Bit Manipulation Mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.

Behavior:

n/a

Related Commands: lld_PpbLockBitSetCmd, lld_PpbLockBitReadCmd, lld_PpbLockBitExitCmd

Example Code:

```
addr    = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_PpbLockBitEntryCmd(addr);
```

2.8.7.2 lld_PpbLockBitSetCmd

Description:

This command sets the Flash Protection Mode to PPB Mode (as opposed to Password Protection Mode).

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.

Behavior:

n/a

Related Commands: lld_PpbLockBitEntryCmd, lld_PpbLockBitReadCmd, lld_PpbLockBitExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_PpbLockBitSetCmd(addr);
```

2.8.7.3 lld_PpbLockBitReadCmd

Description:

This command read the value of the PPB Lock Bit.

Returns: FLASHDATA (0=PPB Protection selected, 1=not selected)

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.

Behavior:

n/a

Related Commands: lld_PpbLockBitEntryCmd, lld_PpbLockBitSetCmd, lld_PpbLockBitExitCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

data = lld_PpbLockBitReadCmd(addr);
printf("%8.8X\n", data);
```

2.8.7.4 lld_PpbLockBitExitCmd

Description:

This command exits the PPB Lock Bit Manipulation Mode.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash.

Behavior:

n/a

Related Commands: lld_PpbLockBitEntryCmd, lld_PpbLockBitSetCmd, lld_PpbLockBitReadCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

lld_PpbLockBitExitCmd(addr);
```

2.8.8 Sector Protection Commands

2.8.8.1 lld_SectorLockCmd

Description:

This command locks and protects all sectors.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.
offset	FLASHDATA	An index into the location where the sector to lock.

Behavior:

All sectors will be locked for writing.

Related Commands: lld_SectorUnlockCmd, lld_SectorLockRangeCmd

Example Code:

```
lld_SectorLockCmd(base_addr, offset);
```

2.8.8.2 lld_SectorUnlockCmd

Description:

This command unlocks and un-protects a sector.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.
offset	FLASHDATA	An index into the location where the sector to unlock.

Behavior:

The specified sector will be unlocked for writing.

Related Commands: lld_SectorUnlockCmd, lld_SectorLockRangeCmd

Example Code:

```
lld_SectorUnlockCmd(base_addr, offset);
```

2.8.8.3 lld_SectorLockRangeCmd

Description:

This command locks and protects a range of sectors.

Returns: -1 for error, 0 for success

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.
StartSec	ADDRESS	An index into the location where the start sector to lock.
StopSec	ADDRESS	An index into the location where the stop sector to lock.

Behavior:

The specified range of sectors will be lock for writing.

Related Commands: lld_SectorUnlockCmd, lld_SectorLockRangeCmd

Example Code:

```
lld_SectorLockRangeCmd(base_addr, StartSec, StopSec);
```

2.9 Miscellaneous APIs

2.9.1 Miscellaneous Commands

2.9.1.1 lld_SetConfigRegCmd

Description:

This command is used to set the Configuration Register. Refer to the data sheet for specific bit information.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.
value	FLASHDATA	The configuration data to be written.

Behavior:

n/a

Related Commands: lld_ReadConfigRegCmd

Example Code:

```
addr   = (FLASHDATA *)   strtoul(argv[1], 0, input_radix);
value  = (FLASHDATA) strtoul(argv[2], 0, input_radix);

lld_SetConfigRegCmd(addr, value);
```

2.9.1.2 lld_SetConfigRegCmd (Device with Address Space Overlay Mode)

Description:

This command is used to set the Configuration Register. Refer to the data sheet for specific bit information.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.
offset	ADDRESS	sector offset for ASO (Address Space Overlay).
value	FLASHDATA	The configuration data to be written.

Behavior:

n/a

Related Commands: lld_ReadConfigRegCmd

Example Code:

```
lld_SetConfigRegCmd(base_addr, offset, value);
```

2.9.1.3 lld_SetConfigRegCmd (WS-P Device)

Description:

The command is used to set the Configuration Registers for WS-P devices. Refer to the data sheets for specific bit information.

Returns: n/a

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash bank to be manipulated.
value	FLASHDATA	The data to be written to Configuration Register 0.
Value1	FLASHDATA	The data to be written to Configuration Register 1.

Behavior:

n/a

Related Commands: lld_ReadConfigRegCmd

Example Code:

```
lld_SetConfigRegCmd( base_addr, value, value1);
```

2.9.1.4 lld_ReadConfigRegCmd

Description:

This command reads the Configuration Register word.

Returns: Configuration Register Word

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.

Behavior:

n/a

Related Commands: lld_SetConfigRegCmd

Example Code:

```
addr   = (FLASHDATA *) strtoul(argv[1], 0, input_radix);

data = lld_ReadConfigRegCmd(addr);
printf("%8.8X\n", data);
```

2.9.1.5 lld_ReadConfigRegCmd (Device with Address Space Overlay Mode)

Description:

This command reads the Configuration Register word.

Returns: Configuration Register Word

Parameters:

Name	Type	Description
base_addr	FLASHDATA *	The base address of the flash to be manipulated.
offset	ADDRESS	Sector offset to read.

Behavior:

n/a

Related Commands: lld_SetConfigRegCmd

Example Code:

```
data = lld_ReadConfigRegCmd(base_addr, offset);
  printf("%8.8X\n", data);
```

3. Revision History

Section	Description
Revision 01 (January 7, 2010)	
	Initial revision
Revision 02 (February 24, 2011)	
Files	Added sentence about trace.c/trace.h
Making the LLD Work in Your Environment	Changed S29GL512R and S29GLxxxR to S29GL512S and S29GLxxxS
Revision 03 (September 13, 2011)	
Global	Rearranged the order in which the sections to improve usability of document.
Common API	Added paragraph under section heading
Basic Operations	Modified note
LLD Clean-Up	Modifications: Moved lld_GetVersion to of “Basic Operations” section Changed the name of lld_EraseSuspendnOp to lld_EraseSuspendnOp(CMD1) Changed the name of lld_ProgramSuspendOp to lld_ProgramSuspendOp(CMD1) and edited the table description Edited the table description of lld_BlandkCheckOP Changed the name of lld_StatusRegReadCmd to lld_StatusRegReadCmd(CMD2) Edited the example code of lld_StatusRegReadCmd Changed the name of lld_StatusRegClearCmd to lld_StatusRegClearCmd(CMD2) Removed: lld_BitfieldProgrammingOp lld_BitFieldCmd APIs. lld_SecSiSectorExitCmd Added: lld_StatusGetReg lld_StatusClear(CMD1) lld_StatusClear(CMD2) lld_EraseSuspendOp(CMD2) lld_ProgramSuspendOp(CMD2) lld_GetVersion API lld_StatusRegReadCmd(CMD1) lld_StatusRegClearCmd(CMD1) lld_AutoselectEntryCmd(Device with Address Space Overlay Mode) lld_LockReg2EntryCmd lld_SetConfigRegCmd(WS-P Device)

Colophon

Trademarks and Notice