NEC

User's Manual

# Library for the Data Flash Access Layer

32-/16-bit Single-Chip Microcontroller

Document No. U18400EE1V0UM00
Date Published September 2006

## NOTES FOR CMOS DEVICES

### ① VOLTAGE APPLICATION WAVEFORM AT INPUT PIN

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between $V_{IL}$ (MAX) and $V_{IH}$ (MIN) due to noise, etc., the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between $V_{IL}$ (MAX) and $V_{IH}$ (MIN).

### ② HANDLING OF UNUSED INPUT PINS

Unconnected CMOS device inputs can be cause of malfunction. If an input pin is unconnected, it is possible that an internal input level may be generated due to noise, etc., causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using pull-up or pull-down circuitry. Each unused pin should be connected to $V_{DD}$ or GND via a resistor if there is a possibility that it will be an output pin. All handling related to unused pins must be judged separately for each device and according to related specifications governing the device.

### ③ PRECAUTION AGAINST ESD

A strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it when it has occurred. Environmental control must be adequate. When it is dry, a humidifier should be used. It is recommended to avoid using insulators that easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors should be grounded. The operator should be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with mounted semiconductor devices.

### ④ STATUS BEFORE INITIALIZATION

Power-on does not necessarily define the initial status of a MOS device. Immediately after the power source is turned ON, devices with reset functions have not yet been initialized. Hence, power-on does not guarantee output pin levels, I/O settings or contents of registers. A device is not initialized until the reset signal is received. A reset operation must be executed immediately after power-on for devices with reset functions.

### ⑤ POWER ON/OFF SEQUENCE

In the case of a device that uses different power supplies for the internal operation and external interface, as a rule, switch on the external power supply after switching on the internal power supply. When switching the power supply off, as a rule, switch off the external power supply and then the internal power supply. Use of the reverse power on/off sequences may result in the application of an overvoltage to the internal elements of the device, causing malfunction and degradation of internal elements due to the passage of an abnormal current.
The correct power on/off sequence must be judged separately for each device and according to related specifications governing the device.

### ⑥ INPUT OF SIGNAL DURING POWER OFF STATE

Do not input signals or an I/O pull-up power supply while the device is not powered. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements.
Input of signals during the power off state must be judged separately for each device and according to related specifications governing the device.

*For further information, please contact:*

**NEC Electronics Corporation**
1753, Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8668,
Japan
Tel: 044-435-5111
http://www.necel.com/

**[America]**

**NEC Electronics America, Inc.**
2880 Scott Blvd.
Santa Clara, CA 95050-2554, U.S.A.
Tel: 408-588-6000
800-366-9782
http://www.am.necel.com/

**[Europe]**

**NEC Electronics (Europe) GmbH**
Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211-65030
http://www.eu.necel.com/

**Hanover Office**
Podbielskistrasse 166 B
30177 Hannover
Tel: 0 511 33 40 2-0

**Munich Office**
Werner-Eckert-Strasse 9
81829 München
Tel: 0 89 92 10 03-0

**Stuttgart Office**
Industriestrasse 3
70565 Stuttgart
Tel: 0 711 99 01 0-0

**United Kingdom Branch**
Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908-691-133

**Succursale Française**
9, rue Paul Dautier, B.P. 52180
78142 Velizy-Villacoublay Cédex
France
Tel: 01-3067-5800

**Sucursal en España**
Juan Esplandiu, 15
28007 Madrid, Spain
Tel: 091-504-2787

**Tyskland Filial**
Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 638 72 00

**Filiale Italiana**
Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02-667541

**Branch The Netherlands**
Steijgerweg 6
5616 HS Eindhoven
The Netherlands
Tel: 040 265 40 10

**[Asia & Oceania]**

**NEC Electronics (China) Co., Ltd**
7th Floor, Quantum Plaza, No. 27 ZhiChunLu Haidian
District, Beijing 100083, P.R.China
Tel: 010-8235-1155
http://www.cn.necel.com/

**NEC Electronics Shanghai Ltd.**
Room 2509-2510, Bank of China Tower,
200 Yincheng Road Central,
Pudong New Area, Shanghai P.R. China P.C:200120
Tel: 021-5888-5400
http://www.cn.necel.com/

**NEC Electronics Hong Kong Ltd.**
12/F., Cityplaza 4,
12 Taikoo Wan Road, Hong Kong
Tel: 2886-9318
http://www.hk.necel.com/

**Seoul Branch**
11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku,
Seoul, 135-080, Korea
Tel: 02-558-3737

**NEC Electronics Taiwan Ltd.**
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R. O. C.
Tel: 02-8175-9600
http://www.tw.necel.com/

**NEC Electronics Singapore Pte. Ltd.**
238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253-8311
http://www.sg.necel.com/

G06.8A

# Introduction

| | |
|---|---|
| **Readers** | This User's Manual is intended for users who want to understand the functionality of the Library for the Data Flash Access Layer for devices of the V850 core family. |
| **Purpose** | This User's Manual explains the background and handling of the Library for the Data Flash Access Layer. |
| **Organization** | This User's Manual contains the major sections: • Data Flash Access Library Usage • Data Flash Access Library API |
| **Legend** | Symbols and notation are used as follows: |

| | | |
|---|---|---|
| Weight in data notation | : | Left is high-order column, right is low order column |
| Active low notation | : | $\overline{\text{xxx}}$ (pin or signal name is over-scored) or /xxx (slash before signal name) |
| Memory map address: | : | High order at high stage and low order at low stage |
| **Note** | : | Explanation of (Note) in the text |
| **Caution** | : | Information requiring particular attention |
| **Remark** | : | Supplementary explanation to the text |
| Numeric notation | : | Binary... xxxx or xxxB<br>Decimal... xxxx<br>Hexadecimal... xxxxH or 0x xxxx |

Prefixes representing powers of 2 (address space, memory capacity)

K (kilo): $2^{10} = 1024$

M (mega): $2^{20} = 1024^2 = 1,048,576$

G (giga): $2^{30} = 1024^3 = 1,073,741,824$

# Table of Contents

# List of Figures

# Chapter 1 Overview

This document describes the usage of the Library for the Data Flash Access Layer (DFALib) for NEC's microcontroller with embedded single voltage data flash.

In order to perform low level operation (erase, write, ...) on this data flash, NEC offers the data flash access layer software 'DFALib'.
Using this DFALib also higher sophisticated EEPROM emulation can be realized.
NEC has prepared an application note 'U18005EE1V1AN00' to provide an example how such an EEPROM emulation could be done. Anyhow, also other customer approaches can be realized based on this DFALib, considering the specified data flash features and operating conditions.

## 1.1 Naming Convention

**Data Flash**
A dedicated flash macro including features supporting efficient EEPROM emulation.

**Data Word**
A data word always refers to a 32 bit value as this is the smallest write unit.

**ID_Tag**
The ID_Tag is an additional information stored in a separate area of the data flash. It consists of a single bit. This ID_Tag assists the hardware supported search for these data words provided by the data flash. For each data word in the Data Flash an ID_Tag exist. Those ID_Tags can be set independently of each other.

**DFALib**
Software layer performing the basic operation of the data flash.

## 1.2 Application Note

The application note of the EEPROM emulation is prepared to support the development environments of the companies Green Hills (GHS), IAR and NEC. It includes the source code of the DFALib together with the EEEPROM Emulation library (EEELib) and applications programs, using an installer tool allowing to select the appropriate environment.

## 1.3 Internal Verify

The internal verify is a function to check the condition of all flash cells within the checked range (margin check). If an internal verify passes successfully, the full specified data retention is ensured under the specified operating conditions.
It is important to notice that this function does not actively modify the conditions of the flash cells, but only confirms the correct status.

**Note:** As the execution time of this function, during which the Data Flash is not accessible,
is rather long, it is desired to have as few as possible internal verify operation. Also, as the EEPROM emulation is causing a frequent rewrite of the data sets, the data will not reach an age as it would be the case for program code. Due to those conditions, a different implementation of the internal verify might be chosen.
For support on finding an optimized implementation of the internal verify please contact the NEC support at **flash_support@eu.necel.com**.

## 1.4 Limitations

- During data flash operation power safe mode must not be applied as it will leave the flash in an undefined state when the clock is stopped. For details on the power safe modes please refer to the appropriate chapter of the device UM.

**Caution: Please read all chapters of the users manual carefully and follow exactly the given sequences and recommendations. This is required in order to make full use of the high level of security and safety provided by the devices.**

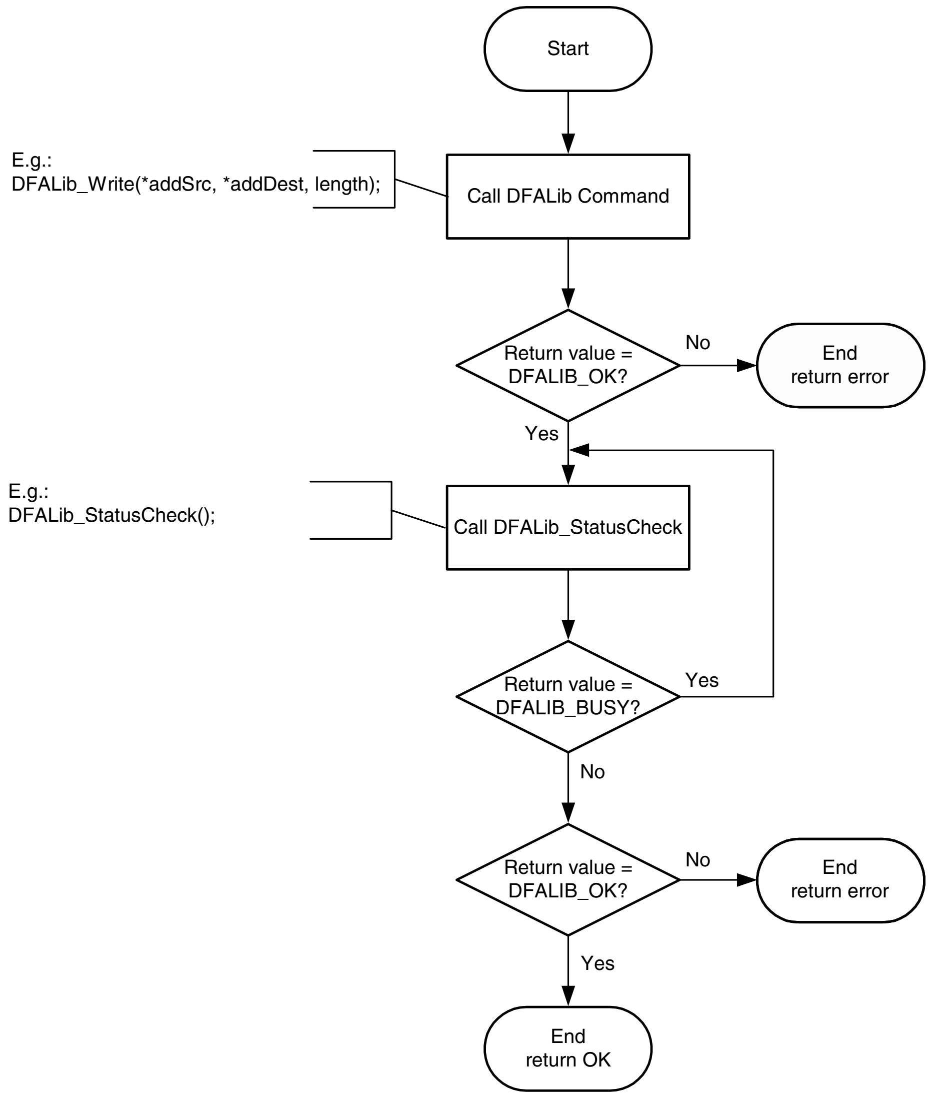
# Chapter 2 DFALib - Usage

## 2.1 Status check - Background operation

As the Flash operations take some time, a dedicated hardware will automatically perform the required tasks in the background without CPU interaction required. However, it has to be checked if those operations are finished. This check has to be performed by the application. The DFALib functions will return immediately after the Flash operation was initiated, allowing the user to perform other tasks while the flash operation is being executed in background by dedicated hardware. In order to check the status of the operation the DFALib_StatusCheck (3.2.7 ”Status Check” on page 20) command has to be used.

**Note:** The device internal hardware will ensure the Flash is in a safe state after the operation was finished. Thus there are no timing requirements on the usage of the status check.

***Figure 2-1: Principle Flow of a DFALib Function Call***



## 2.2 Data Flash Interrupt

In polling mode the *DFALib_StatusCheck* is frequently called by the application. With support of the device specific interrupt "upon data flash operation completion" that frequent call to the *DFALib_StatusCheck* is not necessary, as the *DFALib_StatusCheck* should be called in the interrupt context.

**Note:** For details on the interrupt please refer to the appropriate chapter of the device.
The availability of the interrupt depends on the delivery package's compile options.

## 2.3 Library Software

The DFALib is delivered as a pre-compiled version. Due to that some compile-time configurations in the sourcecode are fixed.

| Configuration | Description |
|---|---|
| *DFALib* Status check | The Flash background operations (e.g. Flash erase) need some time. Before continuing with any Flash related operation, the SW need to check the end of the Flash operation. |
| DFALIB_FLASH_BLOCK_SIZE | The Flash block size of the used device is set with this parameter. Current implementations is 2 kBytes (0x800) |

Please check the delivery package regarding the setting of these configurations. Please also refer to the application note of the DFALib.

### 2.3.1 Supported compilers

General code from the application note supports compilers from GHS, IAR and NEC.
Pre-compiled libraries according to this document are compiled and tested with one dedicated compiler version and dedicated compiler switches that are explicitly listed up in the release package.
The user application must be compiled according not to interfere with these settings.

### 2.3.2 Files required for the DFALib

**[root]**

| | |
|---|---|
| nec_types.h | NEC types convention |
| DFALib.h | Header file containing function prototypes and error definitions |

**[DFALib]**

| | |
|---|---|
| DFALib.o | Precompiled DFALib object file |

The source code of the DFALib will be provided as an application note. Please refer to '1.2 "Application Note" on page 9' for details.

## 2.4 Section Handling

The following sections are DFA library related:

- DFALib_Text, DFALib_TextRAM
  This is *DFALib* code.
  This section is executed from code flash.

**Note:** The above named section have to be added to the projects linker file.

# Chapter 3 DFALib API

The Data Flash Access library provides two types of user functions:

- Operational functions
  Operational functions control the basic flash operations like *Read*, *Write*, *Blankcheck*

- Service functions
  Providing service information to the user.

## 3.1 Used Types

The available error states are:

| Error Name | Description |
|---|---|
| DFALIB_OK | The returned value indicates that the background operation is started. It does not show the status of the operation itself, as this will be returned by the DFALib_StatusCheck() command only. |
| DFALIB_ERR_FLOW | A previous flash operation is not yet finished. Please call the DAFLIB_StatusCheck command until the flash operation is finished, then start the new flash operation. |
| DFALIB_ERR_PARAMETER | The call parameter does not contain valid data |
| DFALIB_BUSY | The operation of which the status has been checked is not yet finished and further status checks are required.<br>The address part does not contain valid data. |
| DFALIB_ERR_FLASHPROCn | (n=0x00~0x04): The operation of which the status has been checked finished with an error.<br>The address part contains the address at which the operation failed.<br>Please refer to the description of the function which initiated the operation to determine the status of the address part. |

## 3.2 Operational Functions

### 3.2.1 Blank Check Backwards

**Library call:**

```
u32 DFALib_BlankCheckBW (       u32 startAdd)
```

**Function description:**

Checks if a single block is blank. The checked area is calculated based on the passed address. The blank check starts at the given address and continues on decreasing address until the beginning of the block is reached or a non blank word is detected. (e.g. with address 8 the following addresses would be checked:
address 8 -> address 4 -> address 0).

**Required parameters:**

| | |
|---|---|
| startAdd | Address within the Data Flash at which the blank check should start. |

**Returned value:**

| | |
|---|---|
| | Bits 0~7 return the following errorcodes while the bits 8~31 are undefined. |
| DFALIB_OK: | The returned value indicates that the background operation is started. It does not show the status of the operation itself, as this will be returned by the DFALib_StatusCheck() command only. |
| DFALIB_ERR_FLOW: | A previous flash operation is not yet finished. Please call the DAFLIB_StatusCheck command until the flash operation is finished, then start the new flash operation.<br>The call parameter does not contain valid data. |
| DFALIB_ERR_PARAMETER: | The call parameter does not contain valid data. |

The header file DFALib.h provides macro definitions to extract the address and the error information from the return value. Use the macro DFALIB_GETADD to extract the address information (bits 8~31) and the macro DFALIB_GETERR to extract the error information (bits 0~7).

### 3.2.2 Blank Check

**Library call:**

```
u32 DFALib_BlankCheck (      u32 blockNoStart,
                             u32 blockNoEnd,
                             u32 offset)
```

**Function description:**

Checks if the blocks defined by the parameter of the function call are blank. The blank check operation starts at the lower most address and continues with increasing addresses (e.g. address 0 -> address 4 -> address 8).

**Note:** The Blank Check Backwards (3.2.1 ”Blank Check Backwards” on page 14) shall be used to find out if a given range of the flash is blank. The DFALib_BlankCheck() is implemented for using in a reasonable time frame after erase operation. After that the Blank Check may fail.

**Required parameters:**

| | |
|---|---|
| blockNoStart | This is the block number (number of the blocks within the Data Flash - e.g. 0) at which the blank check starts. |
| blockNoEnd | Block number up to which (including this block) the blank check operation shall be performed. |
| offset | Offset is an word boundary address within the give block range at which the blank check should start. This parameter is useful to determine if the Flash is still empty above (from address point of view) a given address where data have already been written. |

**Returned value:**

| | |
|---|---|
| | Bits 0~7 return the following errorcodes while the bits 8~31 are undefined. |
| DFALIB_OK: | The returned value indicates that the background operation is started. It does not show the status of the operation itself, as this will be returned by the DFALib_StatusCheck() command only. |
| DFALIB_ERR_FLOW: | A previous flash operation is not yet finished. Please call the DAFLIB_StatusCheck command until the flash operation is finished, then start the new flash operation.<br>The call parameter does not contain valid data. |
| DFALIB_ERR_PARAMETER: | A wrong combination of start and end block is being used.<br>The call parameter does not contain valid data |

The header file DFALib.h provides macro definitions to extract the address and the error information from the return value. Use the macro DFALIB_GETADD to extract the address information (bits 8~31) and the macro DFALIB_GETERR to extract the error information (bits 0~7).

### 3.2.3 ID Search

This function is only available on Data Flash.

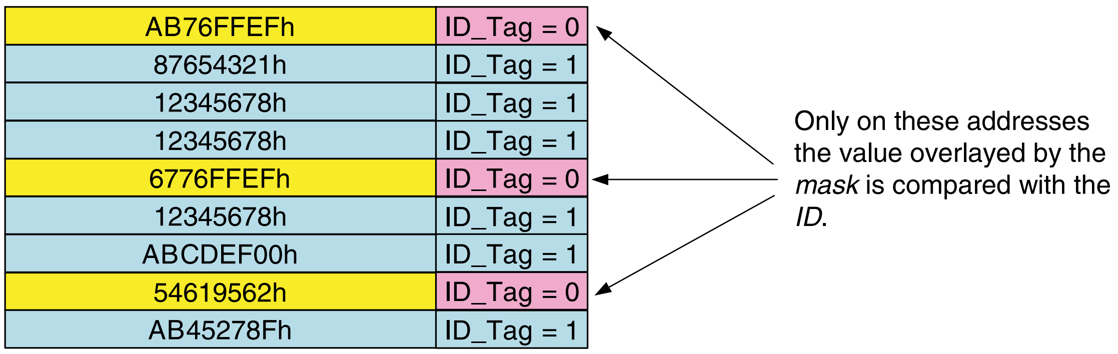
**Library call:**

```
u32 DFALib_SearchID (          u32 startBlk,
                               u32 endBlk,
                               u32 ID,
                               u32 mask)
```

**Function description:**

The given ID value is searched within the address range defined by start and end block. The search operation stops at the first match where the ID and the ID_Tag match. The mask allows to limit the compare operation to a chosen number of bits.
Example:
With an ID 0x$D_1D_2D_3D_4D_5D_6D_7D_8$ and a mask 0x0000FFFF data of the structure 0x$D_1D_2D_3D_4$xxxx and a written ID Tag would be found, with '$D_n$' representing a defined value of 4-bit and 'x' any value.

***Figure 3-1: Principle of the ID Search Function***



**Required parameters:**

| | |
|---|---|
| startBlk | This is the block number (number of the block within the Data Flash - e.g. 0) at which the ID search starts. |
| endBlk | Block number up to which (including this block) the ID search operation should be performed. |
| ID | Defines the ID which is being searched. Only bits which are not masked by the mask parameter are relevant for the search. |
| mask | Mask for the ID. Bits which equal '0' unmask the corresponding bit of the ID, those ID bits are compared during the search, whereas bits which equal '1' mask the corresponding ID bits and prohibits a comparison of those ID bits. |

**Returned value:**

|  |  |
|---|---|
|  | Bits 0~7 return the following errorcodes while the bits 8~31 are undefined. |
| DFALIB_OK: | The returned value indicates that the background operation is started. It does not show the status of the operation itself, as this will be returned by the DFALib_StatusCheck() command only. |
| DFALIB_ERR_FLOW: | A previous flash operation is not yet finished. Please call the DAFLIB_StatusCheck command until the flash operation is finished, then start the new flash operation.<br>The call parameter does not contain valid data |
| DFALIB_ERR_PARAMETER: | A wrong combination of start and end block is being used.<br>The call parameter does not contain valid data |

The header file DFALib.h provides macro definitions to extract the address and the error information from the return value. Use the macro DFALIB_GETADD to extract the address information (bits 8~31) and the macro DFALIB_GETERR to extract the error information (bits 0~7).

### 3.2.4 Write

**Library call:**

```
u32 DFALib_Write (          void *addSrc,
                            void *addDest,
                            u32 length)
```

**Function description:**

Data can be written in series of 1 or 4 words. In case the number of words to be written is 1, a single write operation is performed, 4 words are written sequentially. In case of '0'-length, the 32 data bits are written in one step automatically followed by the write of the ID Tag.

**Required parameters:**

|  |  |
|---|---|
| addSrc | Source address of data to be written to the Flash. |
| addDest | Destination address in Flash at which the data are to be written. The address is the offset in the Flash itself, thus for Data Flash the lower most address is 0x00000000. |
| length | Number of words to be written. The write size for Data without ID_Tag has to be either 1 or 4 words. If a data word with ID_Tag is to be written, the size passed to the function has to be 0. This will then write 1 data word with the ID_Tag. |

**Returned value:**

| | |
|---|---|
| | Bits 0~7 return the following errorcodes while the bits 8~31 are undefined. |
| DFALIB_OK: | The returned value indicates that the background operation is started. It does not show the status of the operation itself, as this will be returned by the DFALib_StatusCheck() command only. |
| DFALIB_ERR_FLOW: | A previous flash operation is not yet finished. Please call the DAFLIB_StatusCheck command until the flash operation is finished, then start the new flash operation. The call parameter does not contain valid data |
| DFALIB_ERR_PARAMETER: | The call parameter does not contain valid data |

The header file DFALib.h provides macro definitions to extract the address and the error information from the return value. Use the macro DFALIB_GETADD to extract the address information (bits 8~31) and the macro DFALIB_GETERR to extract the error information (bits 0~7).

### 3.2.5 Internal Verify

**Library call:**

```
u32 DFALib_IVerify  (        u32 blockNoStart,
                             u32 blockNoEnd)
```

**Function description:**

Performs an internal verify on the specified blocks. The internal verify checks if all flash cells within the blocks still have enough read margin to have the full specified data retention time.

**Required parameters:**

| | |
|---|---|
| blockNoStart | This is the block number (number of the block within the Data Flash - e.g. 0) at which the internal verify starts. |
| blockNoEnd | Block number up to which (including this block) the internal verify operation should be performed. |

**Returned value:**

| | |
|---|---|
| | Bits 0~7 return the following errorcodes while the bits 8~31 are undefined. |
| DFALIB_OK: | The returned value indicates that the background operation is started. It does not show the status of the operation itself, as this will be returned by the DFALib_StatusCheck() command only. |
| DFALIB_ERR_FLOW: | A previous flash operation is not yet finished. Please call the DAFLIB_StatusCheck command until the flash operation is finished, then start the new flash operation. The call parameter does not contain valid data |
| DFALIB_ERR_PARAMETER: | A wrong combination of start and end block is being used. The call parameter does not contain valid data. |

The header file DFALib.h provides macro definitions to extract the address and the error information from the return value. Use the macro DFALIB_GETADD to extract the address information (bits 8~31) and the macro DFALIB_GETERR to extract the error information (bits 0~7).

### 3.2.6 Erase

**Library call:**

```
u32 DFALib_Erase (             u32 blockNoStart,
                               u32 blockNoEnd)
```

**Function description:**

Erases the blocks defined by the parameter of the function.

**Required parameters:**

| | |
|---|---|
| blockNoStart | This is the block number (number of the block within the Data Flash - e.g. 0) at which the erase starts. |
| blockNoEnd | Block number up to which (including this block) the erase operation should be performed. |

**Returned value:**

| | |
|---|---|
| | Bits 0~7 return the following errorcodes while the bits 8~31 are undefined. |
| DFALIB_OK: | The returned value indicates that the background operation is started. It does not show the status of the operation itself, as this will be returned by the DFALib_StatusCheck() command only. |
| DFALIB_ERR_FLOW: | A previous flash operation is not yet finished. Please call the DAFLIB_StatusCheck command until the flash operation is finished, then start the new flash operation.<br>The call parameter does not contain valid data. |
| DFALIB_ERR_PARAMETER: | A wrong combination of start and end block is being used.<br>The call parameter does not contain valid data. |

The header file DFALib.h provides macro definitions to extract the address and the error information from the return value. Use the macro DFALIB_GETADD to extract the address information (bits 8~31) and the macro DFALIB_GETERR to extract the error information (bits 0~7).

### 3.2.7 Status Check

**Library call:**

u32 DFALib_StatusCheck ( void)

**Function description:**

Performs a check on the status of the ongoing Flash operation.

**Required parameters:**

none

**Returned value:**

| | |
|---|---|
| | The returned value consists of two parts, bit 0~7 contain the status itself, whereas bits 8~31 contains an address. |
| DFALIB_OK: | No error occurred and the operation, of which the status has been checked completed successfully.<br>Please refer to the description of the function which initiated the operation to determine the status of the address part. |
| DFALIB_BUSY: | The operation of which the status has been checked is not yet finished and further status checks are required.<br>The address part does not contain valid data. |
| DFALIB_ERR_FLASHPROCn | (n=0x00~0x04): The operation of which the status has been checked finished with an error.<br>The address part contains the address at which the operation failed.<br>Please refer to the description of the function which initiated the operation to determine the status of the address part. |

The header file DFALib.h provides macro definitions to extract the address and the error information from the return value. Use the macro DFALIB_GETADD to extract the address information (bits 8~31) and the macro DFALIB_GETERR to extract the error information (bits 0~7).

**Notes:**

1. Each function can only be completed during the execution of the DFALib_StatusCheck(), thus it is mandatory to execute this command after each function, except DFALib_LibVersion().
2. The DFALib_StatusCheck() has to be called as long as the flash programming hardware is operating and the return value is DFALIB_BUSY.
3. As soon as the operation is finished, the next DFALib_StatusCheck() will return the result of the operation and also automatically reset the flash programming hardware. The user application has to use this return value as any further DFALib_StatusCheck() will not return the status of the operation anymore since the hardware has been reset by the previous function call.

## 3.3 Service Functions

### 3.3.1 Library Version

**Library call:**

u32 DFALib_LibVersion ( void)

**Function description:**

This function returns the version of the library.

**Required parameters:**

none

**Returned value:**

| | |
|---|---|
| u32: | Version of the library. |

**[MEMO]**

# Chapter 4 Electrical Specification

**Timing Characteristics**

($T_A$ = -40 to +125°C,
$V_{DD} = EV_{DD} = BV_{DD}$, $4.0 \leq AV_{REF0} \leq 5.5$ V, $V_{SS} = EV_{SS} = BV_{SS} = AV_{SS} = 0$ V,
$C_L$ = 50 pF,
$f_{XX}$ = 48 MHz)

| Parameter | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|
| Erase | 1block | 19,2 | 24 | 285 | ms |
| | 2 blocks | 19,3 | 24,2 | 285 | ms |
| | 4 blocks | 19,7 | 24,7 | 286 | ms |
| | 8 blocks | 20,6 | 25,8 | 287 | ms |
| Write | 1 word (pass) | 106 | 133 | 1014 | us |
| | 4 words (pass) | 196 | 245 | 1146 | us |
| | 1 word (fail) Note | 676 | 845 | 1014 | us |
| | 4 words (last word fail) Note | 800 | 1000 | 1200 | us |
| IVerify | 1 block | 1,7 | 2,2 | 2,7 | ms |
| | 2 blocks | 3,5 | 4,4 | 5,3 | ms |
| | 4 blocks | 7 | 8,8 | 10,6 | ms |
| | 8 blocks | 14 | 17,5 | 21 | ms |
| Blank Check | 1 block | 257 | 322 | 387 | us |
| | 2 blocks | 462 | 578 | 694 | us |
| | 4 blocks | 800 | 1000 | 1200 | us |
| | 8 blocks | 960 | 1200 | 1440 | us |
| | 1 block (1st word fail) | 53 | 67 | 81 | us |
| Blank Check Backwards | 1 block | 256 | 320 | 384 | us |
| Data Search | 1st word found | 55 | 69 | 83 | us |
| | found on 1000h | 464 | 580 | 696 | us |
| | found on 2000h | 880 | 1100 | 1320 | us |
| | not found (0x4000) | 1680 | 2100 | 2520 | us |

**Note:** Write fail measurement with max specified retries.

[MEMO]