Function Description

Second Stage Boot Loader

netX 10/50/51/52/100/500

V1.6

Hilscher Gesellschaft für Systemautomation mbH

www.hilscher.com

DOC070301FD17EN | Revision 17 | English | 2020-01 | Released | Public

Table of Contents

1 Overview

1.1 About the Second Stage Boot Loader

The Second Stage Boot Loader is necessary to setup a dual-port memory and to offer functionalities to get a netX-based hardware basically running and being able to download firmware files larger than 64 KB via the dual-port memory.

File names

- NETX100-BSL.bin for netX 100/500-based hardware
- NETX50-BSL.bin for netX 50-based hardware
- NETX10-BSL.bin for netX 10-based hardware
- NETX51-BSL.bin for netX 51-based hardware
- NETX52-BSL.bin for netX 52-based hardware

1.2 List of revisions

Rev	Date	Name	Chapter	Revision
14	2014-10-10	MT, SS	2.2.4 2.5.2 3.4.7 2.7 2.8.1.13 2.8.1.20 2.13 2.17.2 4.5 0 6.1.2	▪ INTRAM Memory Layout of netX 51/52 added. ▪ Default DPM timing value for DPM_TIMING_CFG register on netX 10 corrected (was 0x9, has to be 0x11). ▪ Description of new hardware parameters and flags (flash label, assembly features, custom MMIO mapping, MAC address) added. ▪ Description for AIFX module detection added. ▪ Expose MMIO Configuration of netX 52. ▪ Note added to Winbond Quad-SPI feature flag. ▪ Note added that SQIROM must be explicitly enable in tag list. Auto-detect does not support SQIROM. ▪ Description to enter Second Stage Boot Loader "Command Mode" from netX 10/51/52 ROM Loader added. ▪ Description of new boot error code BOOTERROR_E_LICENSE_CHECK added. ▪ Atmel AT45DB321E added to supported flash list. ▪ Winbond W25Q16/80 added to supported flash list.
15	2015-11-16	RM	2.20 2.15.1-3	▪ Removed (netX10 only) in figure text ▪ Added BSL DPM download information for netX50/51/52 ▪ Winbond W25Q64 added to supported flash list.
	2018-01-25	MT		Version 1.5.0.0 ▪ Added SFDP support (SFDP 1.5 only) ▪ Moved error codes from ROM loader and Boot Loader to Chapter 5 ▪ Updated figure 16: SQIROM / XiP firmware update sequence
16	2018-07-16	AM RM	2/3/4/5/6	Version V1.6.0.0 ▪ Reworked chapter structure ▪ Moved Boot loader identification and parameter passing to own chapter ▪ Created new chapter 6 for "Supported FLASH Devices", removed unsupported FLASH devices.
17	2020-01-10	HHe	1.3	Link to libfat updated.
			8.1	Section *Third party software licenses* added.

Table 1: List of revisions

1.3 References to documents

This document refers to the following documents:

[1] Hilscher Gesellschaft für Systemautomation mbH: Dual-Port Memory Interface Manual, netX based products, Revision 12, english, 2012

Table 2: References to documents

This software contains parts of the libfat library, Copyright (c) 2006 Michael "Chishm" Chisholm. (available at https://github.com/devkitPro/libfat, http://devkitpro.org). For conditions and disclaimer, see section *Third party software licenses* on page 67.

1.4 Features

- Boot loader program Size < 64 Kbyte binary code to allow loading via DPM on netX 100/500
- Serial COM support
- USB / CDC support
- DPM support (including host interrupt enabled in DPM mode)
- Security FLASH read / write access to user data area
- File System FAT based (format / read / write) on serial flash only
- Standard DPM layout for "System Channel" and "Handshake Channel" (like defined in netX DPM Manual)
- Extended read and write mailboxes for faster firmware download
- System channel hardware information handling
- Hardware independent (executed in netX internal memory)
- Boot loader update
- File System Restore from SD/MMC
- License Update
- ISA, DPM and PCI support
- Can be programmed as it is into serial flash. If another flash is used the binary needs to be updated using Bootwizard (see section Configuration on page 55)
- Automatic detection of serial flashes via SFDP (see limitations)

1.5 Limitations

- The boot loader does not include a FTL (Flash Translation Layer) and is limited to a FAT file system on the serial flashes only.
- Wear-leveling is not available
- For entering USB boot mode of the Second Stage Boot Loader, it is necessary to add an additional jumper/switch (see netX ROM loader documentation on how to enter extension bus boot mode, to design your hardware)
- Ethernet support not included
- DMA data transfer not included
- Firmware cannot be loaded into internal RAM on netX 10/50/100/500, as this is used by boot loader (netX 51/52 supports firmware in internal RAM since V1.4.13.0)
- Only serial flashes with the feature "erase single pages" are supported (see list in appendix)
- SQI Flashes must conform to netX SQIROM Unit limitations (i.e. only 3 Address bytes, not more than 7 dummy cycles)
- SFDP (Serial Flash Discoverable Parameters)
  - SFDP Flashes must conform to SFPD 1.5 specification
  - SFDP Flashes will be used with 50MHz in SQI and 25MHz in SPI mode
  - SFDP Flashes with special behavior not available via SFDP are not supported (e.g. Microchip SST with global unlock command or Cypress flashes that don’t conform to SFDP specification with 8 dummy cycles after reprogramming 4Bit dummy cycles to a value of 7)

- **License codes currently not readable on a netX 50 based hardware**
- **License Updates cannot be executed on a netX 10/50/51 based hardware**
- **SDRAM and DPM cannot be used simultaneously on netX 10**

2 Boot loader functions

2.1 Overview

- The loader is executed from the netX internal RAM, because this is always known and makes the loader independent from the hardware.
- To be able to start the Second Stage Boot Loader via the DPM, the DPM area must be 64 Kbyte. Otherwise the netX registers, necessary to restart the netX ROM loader to load the Second Stage Boot Loader, are not reachable.
- The Second Stage Boot Loader creates a "*System Channel*" and a "*Handshake Channel*" at DPM offset 0x0 to 0x2FF like defined in the "*netX DPM Interface Manual*".
- The rest of the DPM is used as a separate mailbox system. The mailboxes do not include the standard administration data (pending packets) and consist only of the raw packet data. The mailbox size differs from the standard channel mailboxes:
  - The size of the new send mailbox is calculated in the following way:
    SendMBX size = DPM size - Size of std. channels - Fixed size new RecvMBX
    Example:

DPM size	= 8192 Byte (0x2000)
Std channel size	= 768 Byte (0x300)
Std. Recv MBX size	= 124 Byte (NETX_SYSTEM_MAILBOX_MIN_SIZE)
==> Send MBX size	= 8192 - 768- 124 = 7300 Byte (0x1C84)

  - The new mailbox system has separate handshake pairs in the standard "*System Channel*" handshake cell (offset 0x200 in the DPM).
    HSF_EXT_SEND_MBX_CMD / NSF_EXT_SEND_MBX_ACK = Bit 6 and
    HSF_EXT_RECV_MBX_ACK / NSF_EXT_RECV_MBX_CMD = Bit 7.
  - The download via the extended mailbox system is done by using standard rcX packets. Except the data packet size is adjusted to the extended mailbox size

2.2 INTRAM memory layout

2.2.1 netX 100/500

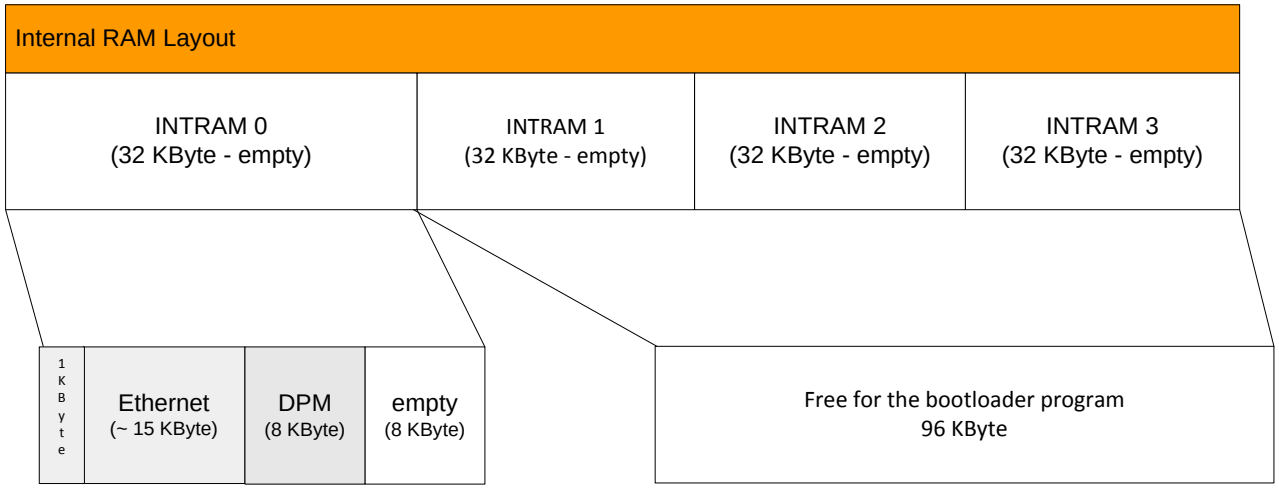


Figure 1: INTRAM memory layout netX 100/500

2.2.2 netX 50

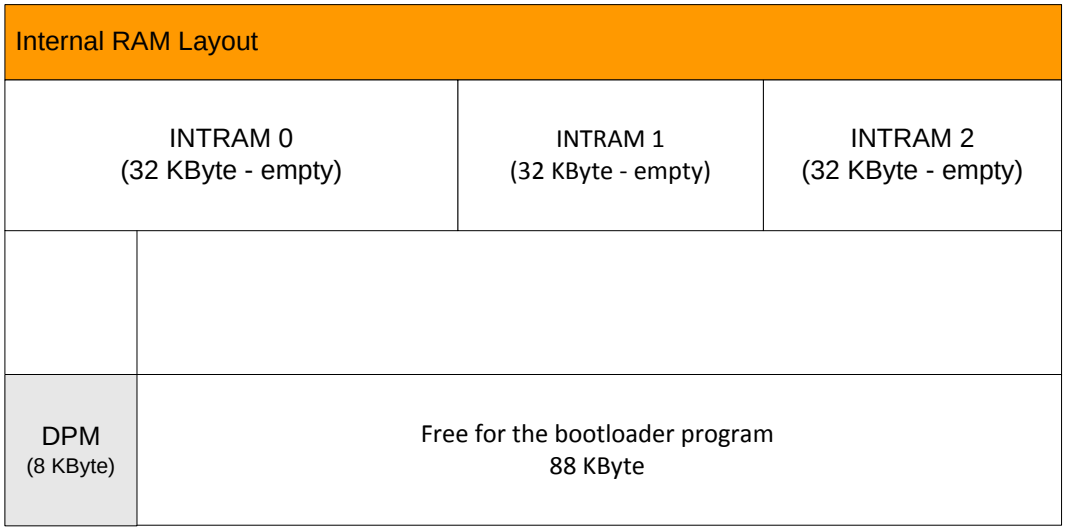


Figure 2: INTRAM memory layout netX 50

2.2.3 netX 10

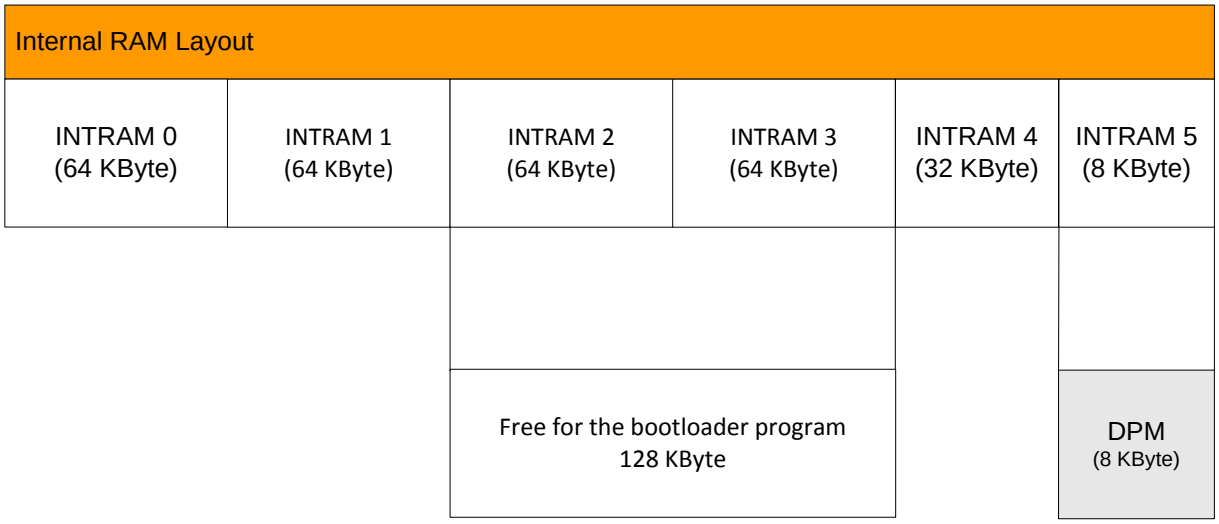


Figure 3: INTRAM memory layout netX 10

2.2.4 netX 51/52

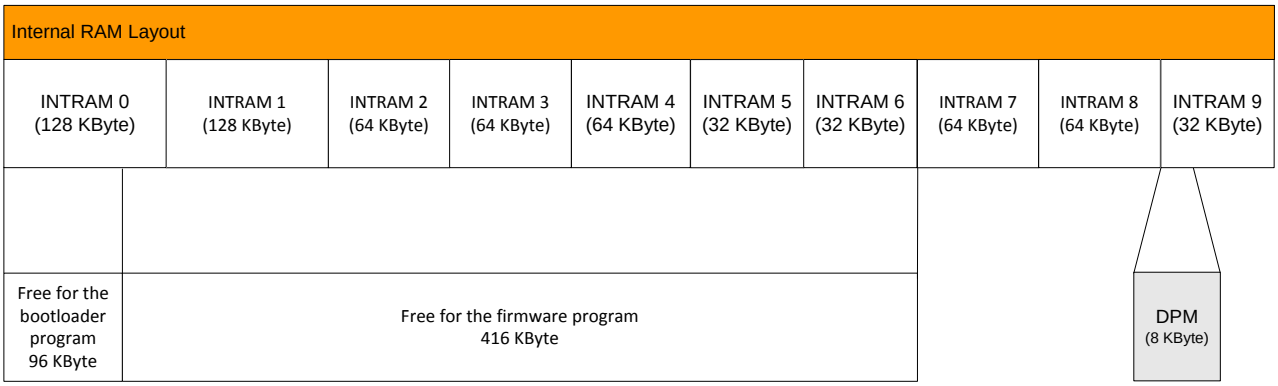


Figure 4: INTRAM memory layout netX 51/52

2.3 Second Stage Boot Loader DPM layout

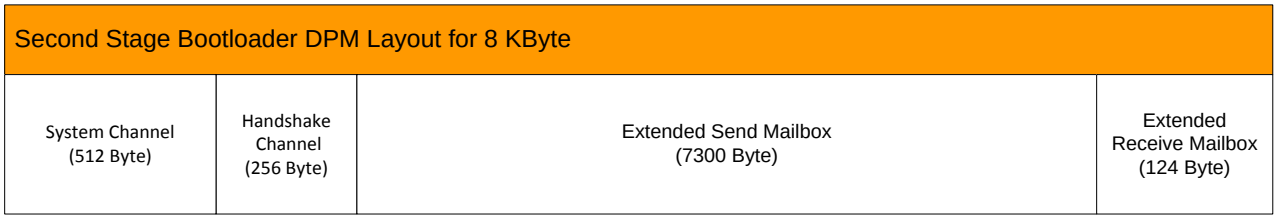


Figure 5: DPM layout

2.4 SDRAM parameter

SDRAM parameters are necessary to initialize the netX RAM controller. These parameters can be taken from different places, depending on the availability. The order where the parameters are taken is pre-defined.

Parameter read order

1. Security Memory entry
2. Tag List
3. Firmware boot header (NXF Header)

Note: The first valid parameter found in the parameter read order will be used and will not be overwritten by later defined values (e. g. *Tag List* parameters cannot override parameters from security memory).

2.5 Default DPM parameters

2.5.1 netX 50/100/500

The following table provides the default netX register settings for the different modes:

Register	ISA (register value)		DPM (register value)	
	8 bit	16 bit	8 bit	16 bit
DPM_ARM_IO_MODE0	0xFFF7E108	0xFFF7FFEF	0x33FFE000	0x33FFEEE7
DPM_ARM_IO_MODE1	0x001FFFFF	0x001FFFFF	0x000E7E67	0x000E7E67
DPM_ARM_IF_CFG0	0x2024CDC2	0x30A4CDC2	0x20244912	0x30044912
DPM_ARM_IF_CFG1	0x0900FF00	0x0900FF00	0x01000000	0x01000000

Table 3: Default DPM timing parameters (netX 50/100/500)

2.5.2 netX 10

The following table provides the default netX register settings for the different modes:

Register	DPM (register value)	
	8 bit	16 bit
HIF_IO_CFG	0x00000031	0x00000031
DPM_CFG0	0x00000000	0x00000004
DPM_ADDR_CFG	0x00000035	0x00000035
DPM_TIMING_CFG	0x00000011	0x00000011
DPM_RDY_CFG	0x00000003	0x00000003
DPM_MISC_CFG	0x00000000	0x00000000
DPM_IO_CFG_MISC	0x00000000	0x00000004

Table 4: Default DPM timing parameters (netX 10)

Note: The DPM can only be used if no SDRAM is attached to the netX 10.

Starting with V1.4.8.0 the Second Stage Boot Loader has an additional option to automatically detect serial DPM via the DIRQ Pin which is read once during startup. See the following truth table shows for DPM Mode detection:

DIRQ	WIF/SIRQ	Mode
0	X	Serial DPM (SPI Mode 3)
1	0	Parallel DPM, 16 Bit
1	1	Parallel DPM, 8 Bit

Table 5: netX 10 serial / parallel DPM auto-detection

2.5.3 netX 51/52

The following table provides the default netX register settings for the different modes:

Register	DPM (register value)	
	8 bit	16 bit
HIF_IO_CFG	0x00000031	0x00000031
DPM_CFG0	0x00000000	0x00000004
DPM_IF_CFG	0x00000000	0x00000000
DPM_PIO_CFG0	0x00000000	0x00000000
DPM_PIO_CFG1	0x00000000	0x00000000
DPM_ADDR_CFG	0x00000017	0x00000017
DPM_TIMING_CFG	0x00000011	0x00000011
DPM_RDY_CFG	0x00000003	0x00000003
DPM_MISC_CFG	0x00000000	0x00000000
DPM_IO_CFG_MISC	0x00000080	0x00000080

Table 6: Default DPM timing parameters (netX 51)

The Second Stage Boot Loader has an additional option to automatically detect serial DPM via the DIRQ Pin which is read once during startup. See the following truth table shows for DPM Mode detection:

DIRQ	WIF (HIF_PIO35 / SIRQ)	Mode
netX 50 compatibility mode		
N/A	0 (HIF_PIO35)	Parallel DPM, 16 Bit
N/A	1 (HIF_PIO35)	Parallel DPM, 8 Bit
netX 51/52 native mode		
0	X (SIRQ)	Serial DPM (SPI Mode 3)
1	0 (SIRQ)	Parallel DPM, 16 Bit
1	1 (SIRQ)	Parallel DPM, 8 Bit

Table 7: netX 51 serial / parallel DPM auto-detection

Note: The auto-detection pins are different depending on the netX strapping pins SA18/19. See netX Design-In Guide.

2.6 Rotary switch support (slot number)

For devices with a security memory and a device class of RCX_HW_DEV_CLASS_CIFX, rotary switches connected to GPIO8-11 are read for the slot number.

Note: This is done independently from the manufacturer code in the security memory.

2.7 AIFX support

Hilscher provides pluggable AIFX modules to attach different fieldbus connectors to a single board. These AIFX modules provide the bus-specific electrical connection, converters and can be detected by the firmware via an I2C interface.

The AIFX identifier is read by the Second Stage Boot Loader, if the security memory contains the value RCX_HW_ASSEMBLY_I2C_PIO (0xFFFB) in a hardware options field. The DPM will show the correct detected hardware options, read from the AIFX modules. If the read fails the value will still be shown as RCX_HW_ASSEMBLY_I2C_PIO (0xFFFB).

2.8 Configuration through tag list

Since V0.915 the boot loader offers a tag list for configuration. This is the recommended method and is done using by using the Hilscher tag list editor.

The following chapter describes the layout of the tag list and the tags of the Second Stage Boot Loader. For instructions on patching the tags refer to chapter *Tag List Editor* on page 63.

2.8.1 Tag list element definitions

A tag list element is described by a generic header and a data part containing the configuration. Every tag element must have the generic header with a special tag id, to describe the following data.

2.8.1.1 Generic header

The generic header describes the following tag data and structure:

Parameter	Type	Description
ulTagType	DWORD	Type of the tag (describes the structure of the tag data)
ulTagDataLength	DWORD	Length of the following tag data

Table 8: Tag list - Generic header

2.8.1.2 SDRAM parameter

SDRAM parameters can be set using this tag. These settings will only be used if no security memory is available.

If no security memory is available and this tag is not setup (all data 0, or disabled), SDRAM parameters from Firmware boot header will be used.

Tag type	Value
TAG_BSL_SDRAM_PARAMS	0x40000000

Parameter	Type	Description
ulGeneralCtrl	DWORD	SDRAM_GENERAL_CTRL register value. Set to 0 if no SDRAM Parameters are available or the parameters from security memory or .NXF file should be used.
ulTimingCtrl	DWORD	SDRAM_TIMING_CTRL register value.

Table 9: Tag list - SDRAM parameters

2.8.1.3 HIF/DPM parameters (netX 50/100/500)

DPM parameters can be set using this tag. If no parameters are given the Second Stage Boot Loader will use the default settings.

Tag type	Value
TAG_BSL_HIF_PARAMS	0x40000001

Parameter	Type	Description
ulBustype	DWORD	Set the external Bus Type 0: Auto-Detect Bus (Depending on load medium PCI/DPM, PCI if *DeviceClass* is cifX and PCI Vendor ID != 0, ISA if *DeviceClass* is cifX and PCI Vendor ID == 0, DPM otherwise) 1: DPM Mode 2: ISA Mode 3: PCI Mode 0xFFFFFFFF: Disable external bus completely
DPM data		
ulIfConf0	DWORD	HIF timing register values. Set to 0 to use defaults Not needed if PCI Mode is selected
ulIfConf1	DWORD	
ulIoRegMode0	DWORD	
ulIoRegMode1	DWORD	
PCI data		
bEnablePin	BYTE	!=0 to use pin for PCI enable
bPinType	BYTE	Type of pin 0: Ignore CD pin and poll once 1: GPIO 2: PIO 3: HIFPIO 4: MMIO **Note:** Upper bit (#7) needs to be set if pin is inverted
usPinNumber	WORD	Number of pin to use for PCI enable

Table 10: Tag list - DPM parameters (netX 50/100/500)

2.8.1.4 SD/MMC parameter

To enable the restore from SD/MMC card function, the following tag must be setup.

Tag type	Value
TAG_BSL_SDMMC_PARAMS	0x40000002

Parameter	Type	Description
bEnable	BYTE	!=0 to enable SD/MMC Support
bDetectPinType	BYTE	Pin Type for card detection 0: Ignore CD pin and poll once 1: GPIO 2: PIO 3: HIFPIO **Note:** Upper bit (#7) needs to be set if pin is inverted
usDetectPin	WORD	Number of the pin to use (bDetectPinType must be setup correctly)

Table 11: Tag list - SD/MMC parameters

2.8.1.5 UART

The following tag can be used to disable the UART handling of the boot loader.

Tag type	Value
TAG_BSL_UART_PARAMS	0x40000003

Parameter	Type	Description
bEnable	BYTE	Set to 0 to disable UART
abReserved[3]	BYTE[3]	Unused / Reserved

Table 12: Tag list - UART parameters

2.8.1.6 USB

The following tag can be used to enable / disable the USB handling of the boot loader.

Tag type	Value
TAG_BSL_USB_PARAMS	0x40000004

Parameter	Type	Description
bEnable	BYTE	Set to 0 to disable USB
bPullupPinType	BYTE	Pin to use for enabling USB device mode (usually GPIO12 on newer hardware revisions) 0: Don't use enable pin 1: GPIO 2: PIO 3: HIFPIO **Note:** Upper bit (#7) needs to be set if pin is inverted
usPullupPinIdx	WORD	Number of pin to toggle for USB enable

Table 13: Tag list - USB parameters

2.8.1.7 Destination medium (volume)

It is possible to override the default volume being used and to enable automatic flashing of the boot loader, if it was downloaded through DPM.

Tag Type	Value
TAG_BSL_MEDIUM_PARAMS	0x40000005

Parameter	Type	Description
bFlash	BYTE	!=0 will flash the Second Stage Boot Loader if it was loaded via DPM/PCI or serial console, so it will always be available during startup
bMediumType	BYTE	Default medium to use for volumes/disk 0: Auto-Detect medium BSL running from serial Flash → Use serial Flash as volume BSL loaded via PCI/DPM (and not flashed) → Use RAM disk BSL loaded from ParFlash → Don't use any filesysten, but use a NXF File at offset 0xC0040000 1: Always use RAM Disk 2: Always use Serial Flash 3: Always use Parallel Flash (no Filesystem) 4: Use SQIROM (XiP), netX 10/51/52 only **NOTE:** SQIROM mode must be explicitly set and is available in auto-detect mode
abReserved[2]	BYTE[2]	Unused / Reserved

Table 14: Tag list - Volume / medium parameters

2.8.1.8 External chip select configuration (EXT_SRAM)

It is possible to let the boot loader initialize the external netX chip selects (CS0-CS3), using the following tag.

Tag type	Value
TAG_BSL_EXTSRAM_PARAMS	0x40000006

Parameter	Type	Description
aulRegVal[4]	DWORD	Register values (MEM_SRAM0_CTRL0-4) to set. If 0 the register won't be touched

Table 15: Tag list - External chip select parameters

2.8.1.9 Hardware data parameters

To allow firmware validation on devices without a security memory, it is possible to patch the hardware data into a tag. If a security memory is found, this tag will be ignored.

Tag type	Value
TAG_BSL_HWDATA_PARAMS	0x40000007

Parameter	Type	Description
bEnable	BYTE	!=0 to enable HW Data tag
abReserved[3]	BYTE[3]	unused / reserved
usManufacturer	WORD	Manufacturer code
usProductionDate	WORD	Production date of hardware
usDeviceClass	WORD	Device class
bHwCompatibility	BYTE	Hardware compatibility index
bHwRevision	BYTE	Hardware revision index
ulDeviceNumber	DWORD	Device Number to show in DPM
ulSerialNumber	DWORD	Serial Number to show in DPM
ausHwOptions	WORD[4]	Hardware options

Table 16: Tag list - Hardware data parameters

2.8.1.10 Fast Startup parameters

The following tag can be used to enable / disable the QSI serial flash handling on netX 50 for fast startup. This set's up 6 pins temporarily for accessing the QSI serial flash.

When using netX 51/netX 52 this tag enables the SQIROM unit of the chip and ignores the pinning options, as the chips integrated QSI support uses dedicated pins

Tag type	Value
TAG_BSL_FSU_PARAMS	0x40000008

Parameter	Type	Description
ulFSUMode	DWORD	Bit mask of fast startup mode **BSL_FSU_MODE_ENABLE (0x00000001)** - Enable fast startup **BSL_FSU_MODE_PINNING (0x000000FE)** - Select the used pins for QSI flash (see pinning table) **BSL_FSU_MODE_DISABLESECMEM (0x80000000)** - Disable security memory readout completely

Table 17: Tag list - Fast Startup parameters

Pinning value (mask)	SPI CSn	SPI CLK	SPI MOSI / IO0	SPI MISO / IO1	SPI WP / IO2	SPI HOLD / IO3
0 (0x01)	37	39	28	29	30	31
1 (0x03)	27	26	28	29	30	31
2 (0x05)	23	22	24	25	26	27

Table 18: Supported Fast Startup pinnings (netX 50)

Note: UART pins will be moved from MMIO34/35 to MMIO 0/1, when Pinning 1 or 2 is selected, and no MMIO configuration is patched into the tag list.

2.8.1.11 MMIO configuration (netX 50)

The MMIO matrix of the 40 MMIO pins on netX 50 can be programmed individually.

Tag type	Value
TAG_BSL_MMIO_NETX50_PARAMS	0x40000009

Parameter	Type	Description
bConfig[0]	BYTE	Function Code for MMIO pin 0
bFlags[0]	BYTE	Inversion flags for MMIO0 0x01 : Invert output 0x02 : Invert input
…		
…		
bConfig[39]	BYTE	Function Code for MMIO pin 39
bFlags[39]	BYTE	Inversion flags for MMIO39 0x01 : Invert output 0x02 : Invert input

Table 19: Tag List - MMIO parameters (netX 50)

Note: MMIO configuration is currently only available in Second Stage Boot Loader 1.4 and later.

2.8.1.12 MMIO configuration (netX 10)

The MMIO matrix of the 24 MMIO pins on netX 10 can be programmed individually.

Tag type	Value
TAG_BSL_MMIO_NETX10_PARAMS	0x4000000A

Parameter	Type	Description
bConfig[0]	BYTE	Function Code for MMIO pin 0
bFlags[0]	BYTE	Inversion flags for MMIO0 0x01 : Invert output 0x02 : Invert input
…		
…		
bConfig[23]	BYTE	Function Code for MMIO pin 23
bFlags[23]	BYTE	Inversion flags for MMIO23 0x01 : Invert output 0x02 : Invert input

Table 20: Tag list - MMIO parameters (netX 10)

2.8.1.13 MMIO configuration (netX 51/netX 52)

The MMIO matrix of the 49 MMIO pins on netX 51 respectively 24 MMIO pins on netX 52 can be programmed individually.

Tag type	Value
TAG_BSL_MMIO_NETX51_PARAMS	0x4000000F

Parameter	Type	Description
bConfig[0]	BYTE	Function Code for MMIO pin 0
bFlags[0]	BYTE	Inversion flags for MMIO0 0x01 : Invert output 0x02 : Invert input
…		
…		
bConfig[48]	BYTE	Function Code for MMIO pin 48
bFlags[48]	BYTE	Inversion flags for MMIO48 0x01 : Invert output 0x02 : Invert input

Table 21: Tag list - MMIO parameters (netX 51/netX 52)

Note: MMIO configuration is currently only available in Second Stage Boot Loader 1.4 and later.

Note: Only a subset of the available MMIO signals could be bonded out on the smaller 244 pin package of the netX 52.

2.8.1.14 HIF/DPM parameters (netX 10)

DPM parameters can be set using this tag. If no parameters are given the Second Stage Boot Loader will use the default settings.

Tag type	Value
TAG_BSL_HIF_NETX10_PARAMS	0x4000000B

Parameter	Type	Description
ulBustype	DWORD	Set the external Bus Type 0: Auto-Detect Bus 1: DPM Mode 0xFFFFFFFF: Disable external Bus completely
ulHifIoCfg	DWORD	HIF timing register values. See netX 10 Program Reference Guide for the register contents / definitions.
ulDpmCfg0	DWORD	
ulDpm_addr_cfg	DWORD	
ulDpm_timing_cfg	DWORD	
ulDpm_rdy_cfg	DWORD	
ulDpm_misc_cfg	DWORD	
ulDpm_io_cfg_misc	DWORD	

Table 22: Tag list - DPM parameters (netX 10)

2.8.1.15 HIF/DPM parameters (netX 51)

DPM parameters can be set using this tag. If no parameters are given the Second Stage Boot Loader will use the default settings.

Tag type	Value
TAG_BSL_HIF_NETX51_PARAMS	0x40000010

Parameter	Type	Description
ulBustype	DWORD	Set the external Bus Type 0: Auto-Detect Bus 1: DPM Mode 0xFFFFFFFF: Disable external Bus completely
ulHifIoCfg	DWORD	HIF timing register values. See netX 51 Program Reference Guide for the register contents / definitions.
ulDpmCfg0	DWORD	
ulDpm_if_cfg	DWORD	
ulDpm_pio_cfg0	DWORD	
ulDpm_pio_cfg1	DWORD	
ulDpm_addr_cfg	DWORD	
ulDpm_timing_cfg	DWORD	
ulDpm_rdy_cfg	DWORD	
ulDpm_misc_cfg	DWORD	
ulDpm_io_cfg_misc	DWORD	

Table 23: Tag list - DPM parameters (netX 51)

2.8.1.16 USB descriptor setup

The USB Identification can be changed.

Default Values:

	netX 100/500	netX 50	netX 10	netX 51	netX 52
VendorID	0x1939	0x1939	0x1939	0x1939	0x1939
ProductID	0x0011 / 0x0012	0x0010	0x0101	0x0130	0x0200

Table 24: USB default identification

Tag type	Value
TAG_BSL_USB_DESCR_PARAMS	0x4000000C

Parameter	Type	Description
bCustomSettings	BYTE	!=0 to use custom settings provided
bDeviceClass	BYTE	These fields are only configurable on netX 50/100/500 and should contain the default values. Otherwise the Host may not be able to connect to the target. bDeviceClass=0x02 bSubClass=0x00 bProtocol=0x00
bSubClass	BYTE	
bProtocol	BYTE	
usVendorId	WORD	USB Vendor ID (VID)
usProductId	WORD	USB Product ID (PID)
usReleaseNr	WORD	Release number of device
bCharacteristics	BYTE	Device characteristics (see USB specification)
bMaxPower	BYTE	Maximum bus power in mA (see USB specification)
szVendor	char[16]	Vendor string
szProduct	char[16]	Product string
szSerial	char[16]	Serial number

Table 25: Tag list - USB descriptor parameters

Note: USB descriptor configuration is currently only available in Second Stage Boot Loader 1.4 and later.

2.8.1.17 File system position and size

The position and size of the default disk can be customized using this tag. Customization is only supported for disk with file systems (RAM and serial devices). Parallel flash will always use the defaults.

Default Values:

	RAM disk	Serial Flash	Parallel Flash
Offset	7MB (0x00700000)	64kB (rounded up to next page)	No file system support.
Size	1MB (0x00100000)	From Offset to end of flash	Firmware is always located at address 0xC0040000

Table 26: Default disk position and size

Tag type	Value
TAG_BSL_DISK_POS_PARAMS	0x4000000D

Parameter	Type	Description
ulOffset	DWORD	The disk offset from start of medium. 0xFFFFFFFF: use defaults
ulSize	DWORD	The size of the disk. 0xFFFFFFFF: use defaults

Table 27: Tag list - Disk position parameters

Note: Position and size configuration is available in Second Stage Boot Loader V1.3.7.0 and later.

Note: Position and size configuration was removed in Second Stage Boot Loader V1.5.0.0. It was replaced by the Flash Layout configuration (see 2.8.1.19).

Note: The Second Stage Boot Loader does not check for overlapping areas.
Make sure the Flash file system (disk position), Backup partition and the storage library position do not overlap. Also, the last Flash sector might be occupied by a Flash Device Label (FDL).

2.8.1.18 Backup partition

The *TAG_BSL_BACKUP_POS_PARAMS* tag can be used to reserve a flash area for recovery firmware support. The *Backup Partition* will be located at the position given in the *TAG_BSL_BACKUP_POS_PARAMS*. A recovery firmware placed into the recovery partition will be started if no firmware has been found in default file system.

This recovery firmware **cannot** be updated using Second Stage Boot Loader mechanisms.

Tag type	Value
TAG_BSL_BACKUP_POS_PARAMS	0x4000000E

Parameter	Type	Description
bMedium	UINT8	Set medium containing recovery firmware 0: Feature disabled (default) 1: Serial Flash 2: Parallel Flash
abReserved[3]	UINT8	Unused (set to zero)
ulOffset	UINT32	Offset inside destination medium
ulSize	UINT32	Size of the reserved area in destination medium

Table 28: Tag list – Backup partition parameters

Note: Backup partition configuration is available in Second Stage Boot Loader V1.3.7.0 and later.

Note: Backup partition configuration was removed in Second Stage Boot Loader V1.5.0.0. It was replaced by the Flash Layout configuration (see 2.8.1.19).

Note: The Second Stage Boot Loader is only able to start a recovery firmware, but cannot update it. For updating such a firmware a customer specific mechanism must be included in the recovery firmware itself, or by using the Hilscher Bootwizard tool to update the flash completely.

Configuration of the backup partition

The configuration is done via the Hilscher Tag List Editor and will be patched into the Second Stage Boot Loader binary. To allow usage of a recovery firmware, the default file system size must be reduced to get a free flash area for Backup partition placement.

Note: The Second Stage Boot Loader does not check for overlapping areas.
Make sure the Flash file system (disk position), Backup partition and the storage library position do not overlap. Also, the last Flash sector might be occupied by a Flash Device Label (FDL).

Recovery firmware start-up handling

The default startup sequence will be executed and if no bootable firmware was found in the default FLASH File System (e.g. Checksum Error, no File Found, firmware does not match hardware options), the Second Stage Boot Loader will check the *Backup Partition* for a bootable firmware.

Such a recovery firmware will also be validated like any other bootable firmware (see chapter “Firmware Validation”) if a security memory exists or corresponding parameters are patched into the *Tag List*.

And it must contain at least a *netX Bootheader* to be bootable. But we suggest generating a recovery firmware like a standard netX firmware (.NXF) including a complete NXF header definition allowing a complete validation of the firmware.

The file size contained in the *netX Bootheader* will be checked against the configured backup partition size and if it is lager it won't be started.

Example

Default Flash layout:

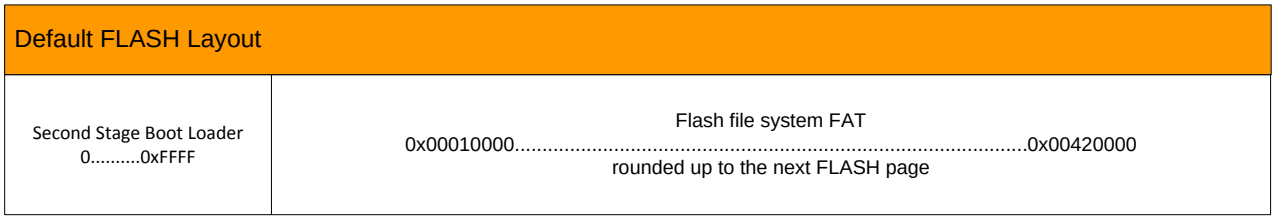


Figure 6: Default Flash layout

Modified Flash layout (sample for AT45DB321, with 528 byte pages):

TAG definition			
Default MEDIA settings			
TAG_BSL_MEDIUM_PARAMS		**See chapter “Destination medium (volume)**	
Disk position tag		**Backup partition tag settings**	
Offset	0x000101D0	Offset	0x00300000
Size	0x002EFC80	Size	0x00120000
n.c.		Medium	Serial Flash

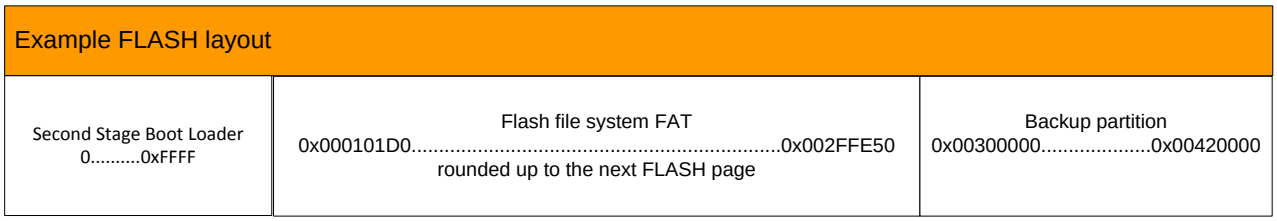


Figure 7: Example Flash layout

2.8.1.19 Flash Layout

The flash layout tag consists of the file system position and size, the backup partition and the storage library position and size.

File system position and size

For a detailed description see chapter 2.8.1.17

Backup partition

For a detailed description see chapter 2.8.1.18

Storage library position and size

The position and size of a potential storage library partition can be customized using this tag. The storage library position is not used within the 2nd Stage Boot Loader. The parameters are handed over to the firmware by using Boot Tokens.

Tag type	Value
TAG_BSL_FLASH_LAYOUT_PARAMS	0x40000013

Parameter	Type	Description
ulFileSystemOffset	UINT32	The disk offset from start of medium. 0xFFFFFFFF: use defaults
ulFileSystemSize	UINT32	The size of the disk. 0xFFFFFFFF: use defaults
bMedium	UINT8	Set medium containing recovery firmware 0: Feature disabled (default) 1: Serial Flash 2: Parallel Flash
abReserved[3]	UINT8	Unused (set to zero)
ulBackupPartOffset	UINT32	Offset inside destination medium
ulBackupPartSize	UINT32	Size of the reserved area in destination medium
ulLibStorageOffset	UINT32	Offset/Position of the storage library area. Default value: 0x00000000.
ulLibStorageSize	UINT32	Size of the storage library area. Default value: 0x00000000. If set to 0, no storage library area is available.

Table 29: Tag list – Storage library position parameters

Note: The Second Stage Boot Loader does not check for overlapping areas.
Make sure the Flash file system (disk position), Backup partition and the storage library position do not overlap. Also, the last Flash sector might be occupied by a Flash Device Label (FDL).

Note: A flash layout configuration is available in Second Stage Boot Loader V1.5.0.0 and later.

2.8.1.20 Custom serial Flash

The *TAG_BSL_SERFLASH_PARAMS* tag can be used to specify a custom serial flash that is not supported out of the box.

Attention: The used flash must support the page-erase opcode or must be able to perform a combined erase and program single page instruction.

Tag type	Value
TAG_BSL_SERFLASH_PARAMS	0x40000011

Parameter	Type	Description
bUseCustomFlash	UINT8	!= 0 to enable the custom flash
szName[16]	UINT8	Name of the flash (NUL terminated)
ulSize	UINT32	Size of the flash memory in bytes
ulClock	UINT32	Maximum speed in kHz
ulPageSize	UINT32	Size of one page in bytes
ulSectorPages	UINT32	Size of one sector in pages
bAdrMode	UINT8	Addressing mode 0: Linear 1: Bitshift derived from the pagesize
bReadOpcode	UINT8	Opcode for 'continuous array read' command
bReadOpcodeDCBytes	UINT8	Don't care bytes after readOpcode and address
bWriteEnableOpcode	UINT8	Opcode for 'write enable' command, 0x00 means no write protect mechanism
bErasePageOpcode	UINT8	Opcode for 'erase page'
bEraseSectorOpcode	UINT8	Opcode for 'erase sector'
bEraseChipCmdLen	UINT8	Length of the 'erase chip' command, 0 means not available
abEraseChipCmd[4]	UINT8	Command to erase the complete chip
bPageProgOpcode	UINT8	Opcode for 'page program (without built-in erase)'
bBufferFill	UINT8	Opcode for 'fill buffer with data'
bBufferWriteOpcode	UINT8	Opcode for 'write buffer to flash'
bEraseAndPageProgOpcode	UINT8	Opcode for 'page program with built-in erase'
bReadStatusOpcode	UINT8	Opcode for 'read status register'
bStatusReadyMask	UINT8	Bitmask indicating device busy
bStatusReadyValue	UINT8	Xor bitmask for device busy
bInitCmd0_length	UINT8	Length of the first init command in bytes
abInitCmd0[3]	UINT8	First command string to init the device
bInitCmd1_length	UINT8	Length of the second init command in bytes
abInitCmd1[3]	UINT8	Second command string to init the device
bIdLength	UINT8	Length in bytes of the id_send, id_mask and id_magic fields
abIdSend[9]	UINT8	Command string to request the id
abIdMask[9]	UINT8	Mask for the device id
abIdMagic[9]	UINT8	Magic sequence of this device

Parameter	Type	Description
ulFeatures	UINT32	Bitmask of features 1: Windbond Quad-SPI (netX 50/51/52 only) **Note:** To enable Quad-SPI mode the BSL_FSU_MODE_ENABLE bit must be set in Fast Startup parameters (see section *Fast Startup parameters* on page 19). **Note:** Execute in Place support is not affected by this feature flag.

Table 30: Tag list – Serial Flash parameters

Note: Custom serial flash configuration is available in Second Stage Boot Loader V1.4.9.0 and later.

Note: A special firmware that uses the flash parameters from the Second Stage Boot Loader must be used.

2.8.1.21 Custom SQI Flash (netX10/51/52)

The *TAG_BSL_SQIFLASH_PARAMS* tag can be used to specify a custom SQI flash that is not supported out of the box. It must be used in conjunction with the custom serial flash tag.

Attention: The used flash must conform to the SQIROM unit inside the netX chip (e.g. max wait cycles of 7 clocks, not more than 3 Address bytes, …).

Tag type	Value
TAG_BSL_SQIFLASH_PARAMS	0x40000012

Parameter	Type	Description
bValid	UINT8	!= 0 to enable custom SQI parameters
usFreqMHz	UINT16	Maximum Clock frequency in MHz (must be inside netX Limits)
bAddrBytes	UINT8	Number of address bytes (netX only support 3 address bytes)
bFastReadCmd	UINT8	Command to enter 0-4-4 mode
bModeClocks	UINT8	Number of mode clocks
bDummyCycles	UINT8	Number of dummy cycles
bQERType	UINT8	Quad Enable Type (see Quad Enable Requirements in SFDP Specification for more details) 1: QE is Bit1 in status register 2 (word accessible) 2: QE is Bit6 in status register 1 3: QE is Bit7 in status register 2 4: QE is Bit1 in status register 2 (byte accessible) 5: QE is Bit1 in status register 2 (split transaction)
bEntryType	UINT8	Sequence to enter 0-4-4 mode 1:Send 0xA5 as last address bytes 2: Set 0xA5 in configuration register
bExitType	UINT8	Sequence to exit 0-4-4 mode 1: Send 0x00 at end of current read 2: Send 0xFF as address

Table 31: Tag list – SQI Flash parameters

Note: Custom SQI flash configuration is available in Second Stage Boot Loader V1.5.0.0 and later.

Note: A special firmware that uses the flash parameters from the Second Stage Boot Loader must be used.

2.9 Default detection of supported media

The following chapters describe the default detection order of flashes and RAM. These priorities are only valid if no target medium is forced through the tag list.

2.9.1 PCI mode

The PCI Mode defaults to RAM disk operation which sets up a RAM disk with an offset of 7MB (0x00700000) in the SDRAM with a size of 1MB.

2.9.2 DPM mode

The following table shows the order of medium detection if the boot loader is executed from flash media. Once a medium in the list is found, this will be used for file downloads.

Medium	Description
Serial Flash	The file system will be placed in serial flash using the FAT file system. This allows non-volatile storage of firmware and configuration
Parallel Flash	Only a single .NXF can be downloaded. There is no flash file system available. This allows non-volatile storage of firmware.
SDRAM	A FAT file system will be created in the last 1MB of the SDRAM. This will only allow volatile storage of firmware and configuration

Table 32: Medium detection in DPM mode (if flashed)

Note: If the boot loader was downloaded and executed through DPM (without being flashed) it will default to RAM disk operation.

2.10 RAM disk layout

The RAM disk will always be created at offset 7MB (0x00700000) in the SDRAM and will have a size of 1 M. For this to work SDRAM parameters are needed. You can either provide them by a security memory or by patching them into the tag list.

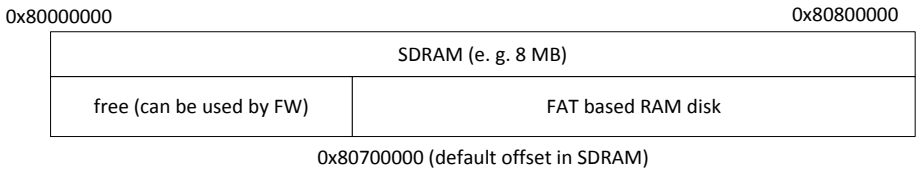


Figure 8: RAM disk layout

2.11 Serial Flash layout / Flash disc / File system

The Second Stage Boot Loader creates flash disc with a FAT file system in the serial flash. The file system starts at the next free page/sector after the Second Stage Boot Loader and occupies the rest of the flash.

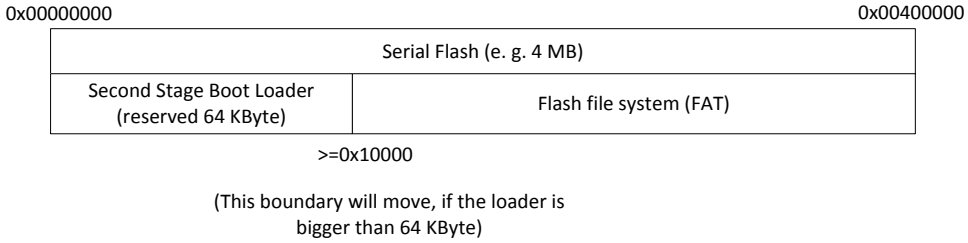


Figure 9: Serial Flash layout

Directory layout

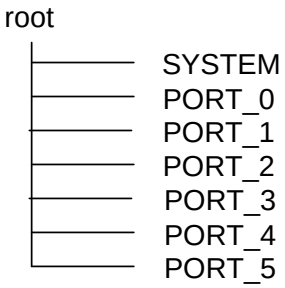


Figure 10: Directory layout

2.12 Parallel Flash support

Since V0.915 the boot loader is able to work with parallel flashes with the following limitations:

- Can only handle a single firmware file (.NXF)
- Needs CFI compliant parallel flash
- No Flash file system support
- No configuration database download (.NXD), as no flash file system is available. Downloading configuration / Warmstart parameters must be handled by firmware

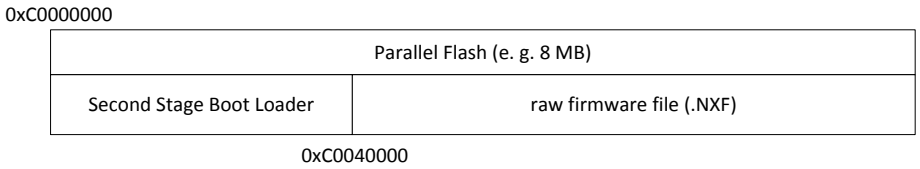


Figure 11: Parallel Flash layout

Note: When using parallel flash all downloads of RCX_XFER_MODE_FILESYSTEM will be checked for the extension ".NXF". All other files will be rejected.

2.13 SQIROM/Execute in place support (netX 10/51/52 only)

Since V1.400 the boot loader is able to work with serial 4 Bit flashes.

The Second Stage Boot Loader creates flash disc with a FAT file system in the serial flash. The file system starts at a fixed offset of 2MB and occupies the rest of the flash.

The Execute-in-Place (XiP) firmware will always be placed at offset 128KB from start of flash.

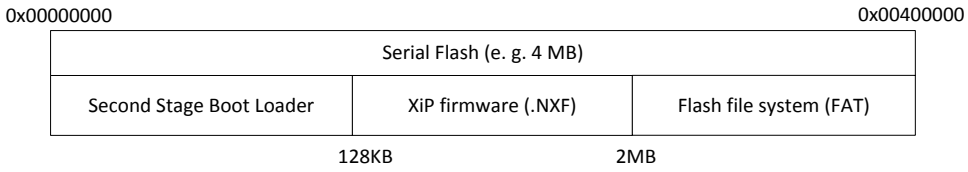


Figure 12: Serial Flash layout (in SQIROM mode)

Directory layout

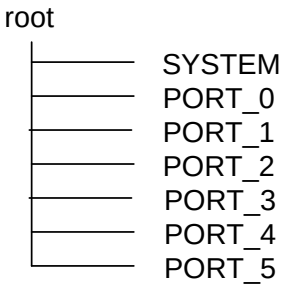


Figure 13: Directory layout

Note: This mode is only available on netX 10, netX 51 and netX 52 and it must be explicitly enabled in the tag list by setting boot loader medium to SQIROM.

2.14 Restore the Flash file system from SD/MMC

Tools like Sycon.net can create a backup of the Flash File System. This backup is stored on SD/MMC. The Second Stage Boot Loader will restore such a backup if all of the following conditions are met:

- “Restore from SD/MMC” is enabled in the configuration (see chapter 2.6)
- a SD/MMC is inserted during the system boot sequence (see chapter 2.8.1.4 to configure the insertion signal)
- the inserted SD/MMC holds a valid backup configuration

If all 3 conditions are met, the Second Stage Boot Loader attempts to restore the backup to the Flash File System. If a notification was set in the backup’s configuration, the SYS led toggles between yellow and green with 16Hz for the selected time. The restore operation can be canceled in this phase by simply removing the SD/MMC. If no notification was enabled, the restore operation will start immediately.

Attention:	This operation cannot be reversed! The restore operation formats the Flash File System of the device and copies the contents of the backup to the File System. Any content of the devices file system will be lost.

Make sure to backup any necessary data before starting the restore process.

Once the restore operation is finished – successful or not – the boot sequence will continue.

Note:	To enable this feature you will need to patch the SD/MMC tag with the *TAG List* editor.

2.15 Flashing boot loader via DPM

The Second Stage Boot Loader can be flashed using the internal netX ROM loader. The handling differs between the different netX chips.

2.15.1 netX 100 / 500

Limitations

- DPM size of 64 kByte (for accessing the netX status register)
- 8 Bit DPM access, as the netX ROM loader uses an 8 Bit interface
- Endianness (default = Little Endian) must be handled by the software

"*Bitflip*" Mode

The following steps are required:

1. Activate the netX hardware DPM boot mode
2. Set *bFlash* in the Volume tag of the Bootloader Tag-List**(skipped for V0.913 and earlier)**
3. Copy the *Second Stage Boot Loader* binary into the DPM (start offset *0x0000*)
4. Read the DPM_SYS_STA register at offset 0xFFD8
5. Set bit 7 to the value of bit 3 and write the new value back
6. If an error occurs, bit 3 in the DPM_SYS_STA register is inverted, bit 8-15 show an error code (Error codes are shown in chapter 7 "Error Codes").

The Second Stage Boot Loader will be started and flashed automatically into the detected serial or parallel flash.

2.15.2 netX50

The netX5 offers the following mode to download and start a binary.

"*Mailbox"* mode

Mailbox mode download can be used for binary files larger than 62 KByte. The data will be transferred in smaller blocks via a mailbox mechanism. This mechanism is described in the "*netX50 DPM Boot manual*".

2.15.3 netX 51 / 52

The netX51/52 offers two ways to download and start a binary

"*Bitflip*" mode

This mode compares to the netX 100/ 500 mode but other offsets and sizes are used.

Limitations

- DPM size of 64 kByte
- Size of the boot image <= 62 kBytes
- The DPM connection of configured correctly
  (read / write operations on any 32bit registers works without errors)
- Endianness (default = Little Endian) must be handled by the software.

The host must execute these steps to start the boot image on the netX:

1. Activate the netX hardware DPM boot mode
2. Copy the *Second Stage Boot Loader* binary into the DPM (start offset *0x0800*)
3. Read the *DPM_SYS_STA* register at offset 0x00d8
4. Set bit 7 to the value of bit 3 and write the new value back
5. A changed 'boot ID' in the "*HBoot configuration area*" indicates a correct software start.

If an error occurs, bit 3 in the DPM_SYS_STA register is inverted, bit 8-15 show an error code.
Error codes are listed in chapter: **7.1 ROM Loader Error Codes**.

"*Mailbox*" mode

Mailbox mode download can be used for binary files larger than 62 KByte. The data will be transferred in smaller blocks via a mailbox mechanism. This mechanism is described in the "*netX51/52 DPM Boot manual*".

2.16 Updating the boot loader via DPM

The update procedure is implemented in the Second Stage Boot Loader, using the standard rcX file download mechanism.

This standard mechanism defines some general rules:

- The download transfer command must be executed using the following transfer types:
  RCX_FILE_XFER_SERIAL_FLASH if boot loader runs from serial flash,
  RCX_FILE_XFER_PARALLEL_FLASH if boot loader runs from parallel flash
- The file name must be given in an 8.3 representation <*name*>.<*extension*>, because of the underlying FAT file system.

2.17 Updating the Second Stage Boot Loader or firmware

If the Second Stage Boot Loader itself should be updated it must be switched to its command mode which allows the update.

This is necessary, because a firmware is not able to update the Second Stage Boot Loader and as long as a firmware exists in the FAT file system, the Second Stage Boot Loader will always try to start it.

When updating a firmware that is running from parallel flash or a firmware that has no access to the flash volume, the firmware is not able to update itself and the users must activate the Second Stage Boot Loader before updating the firmware.

2.17.1 Enter Second Stage Boot Loader "Command Mode" from a running firmware

The firmware startup can be deactivated by setting the HSF_BOOTLOADER flag (Bit 0x02) in the system channel handshake flags (host side). If the Second Stage Boot Loader detects this bit, it will not start a firmware and stays in the *"Command Mode"* which is also signaled by specified system LED blink code.

Note: The system LED (SYS LED) blink code is described in the corresponding device manual (default is a yellow/green flashing)

Note: This feature must be supported by the firmware.

2.17.2 Enter Second Stage Boot Loader "Command Mode" from netX ROM loader

The Second Stage Boot Loader Second Stage Boot Loader *"Command Mode"* can also be activated by setting this bit through the netX ROM loader serial command console.

netX 100/500

1. Activate the netX ROM loader mode.
2. Enter "fill 00103200 02000000" to set the bit.
3. Enter "boot" to start the Second Stage Boot Loader.

netX 51/52

1. Activate the netX ROM loader mode by entering "*#" (RUN pin at high level)
2. Enter "m 1018c300 c0000200".
3. Enter "m 1018c380 00000001".
4. Enter "m 1A008200 02000000".
5. Jumper the RUN pin to low level
6. Enter "g 0 0" to start the Second Stage Boot Loader.

netX 50

1. Activate the netX ROM loader mode.
2. Enter "fill 1C003200 02000000" to set the bit.
3. Enter "boot" to start the Second Stage Boot Loader.

netX 10

1. Activate the netX ROM loader mode by entering "*#" (RUN pin at high level)
2. Enter "m 101c1100 00000200".
3. Enter "m 101c1180 00000001".
4. Enter "m 10048200 02000000".
5. Jumper the RUN pin to low level.
6. Enter "g 0 0" to start the Second Stage Boot Loader.

2.17.3 Updating a firmware

The following figure shows how to update a firmware on a device with the following pre-requisites:

- Flashed Second Stage Boot Loader and using the flash as the file storage
- Firmware supporting the Second Stage Boot Loader

ATTTENTION: This update procedure **does NOT work on** so called "**RAM based devices**" (e.g. cifX PCI cards) where Second Stage Boot Loader and firmware are loaded into RAM by the host (driver).

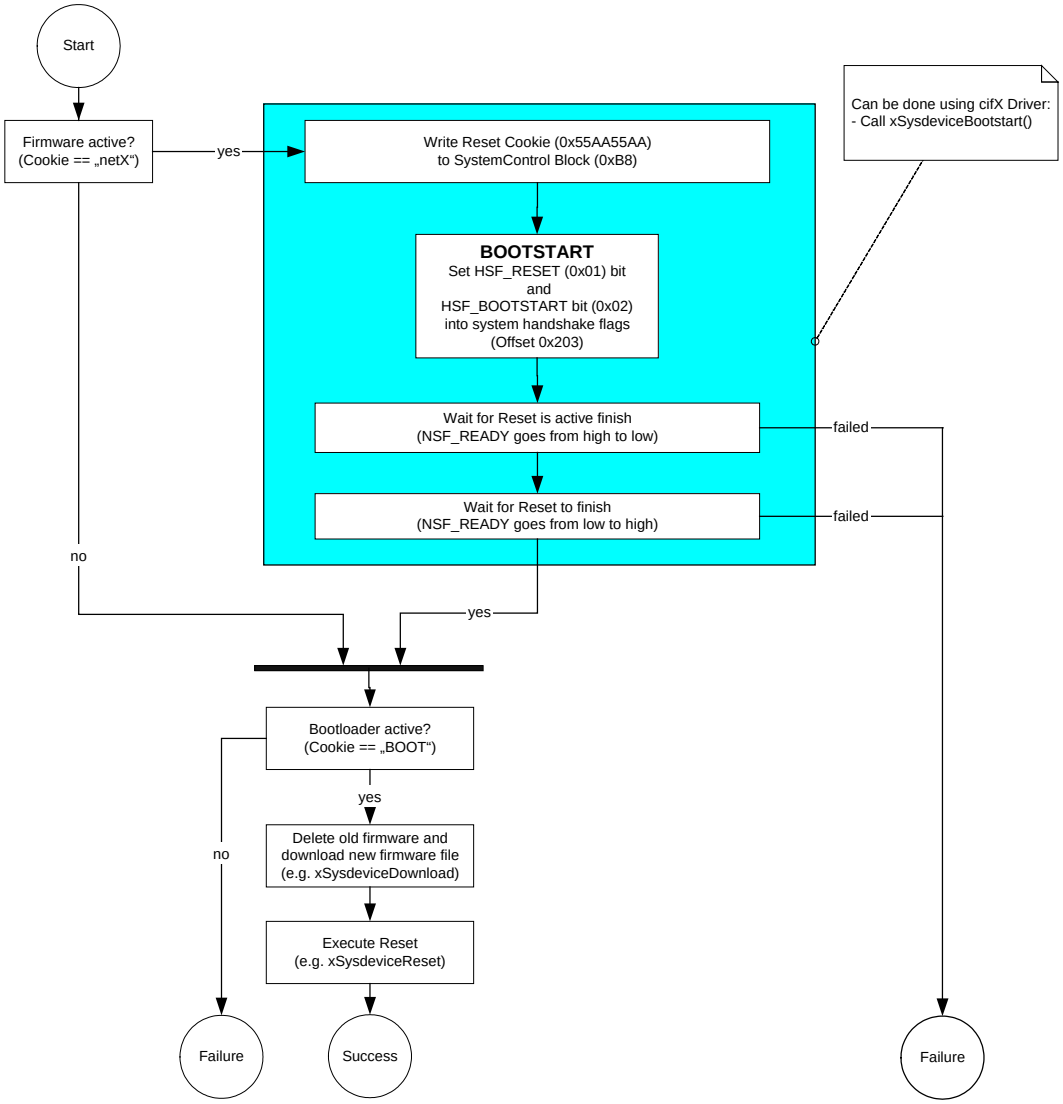


Figure 14: Firmware update sequence

2.18 netX boot mode definitions

The boot mode passed by the netX ROM loader is evaluated by the Second Stage Boot Loader. The following modes are available:

Mode	Description
PCI/Memory (RDY = 1, RUN = 1) DPM (RDY = 1, RUN = 0)	**PCI:** PCI will only be activated if FORCE_PCI is set, usDeviceClass is set to cifX and PCI Vendor ID is != 0, or if the bus type in the tag list is set to PCI. The boot loader will setup the PCI registers according to the security memory, if it is running from flash. Downloads will be made to a RAM disk located at the last 1 MB of the SDRAM, if the loader was downloaded to RAM (via DPM) **ISA:** ISA will be activated it FORCE_ISA is set, usDeviceClass is set to cifX and PCI Vendor ID is 0, or if the bus type in the tag list is set to ISA. Boot loader will use the whole serial flash as a flash disk for firmware and data storage (except its own location), if it is available. The location and size of the flash disk will be determined automatically and passed to the firmware. Boot loader will use parallel flash if available and NO serial flash is found ISA_CSn (PIO 51) defines 8 / 16 Bit mode **DPM:** Default mode Boot loader will use the whole serial flash as a flash disk for firmware and data storage (except its own location), if it is available. The location and size of the flash disk will be determined automatically and passed to the firmware. Boot loader will use parallel flash if available and NO serial flash is found The setup of the DPM is determined by the own boot block of the boot loader. If no settings are patched the default Hilscher values are used. WIF pin defines 8 Bit or 16 Bit mode.
EXTENSIONBUS (RDY = 0, RUN = 0)	This mode is not available as standard mode. It will be used to enter the USB/DPM boot mode ignoring any firmware inside the flash (e.g. restore a broken device / firmware)

Note: RDY and RUN are netX hardware pins.

Note: When entering DPM boot mode and the tag list contains DPM settings the WIF / ISA_CS pin will be ignored and the settings inside the header are used.

2.19 Boot loader start sequence

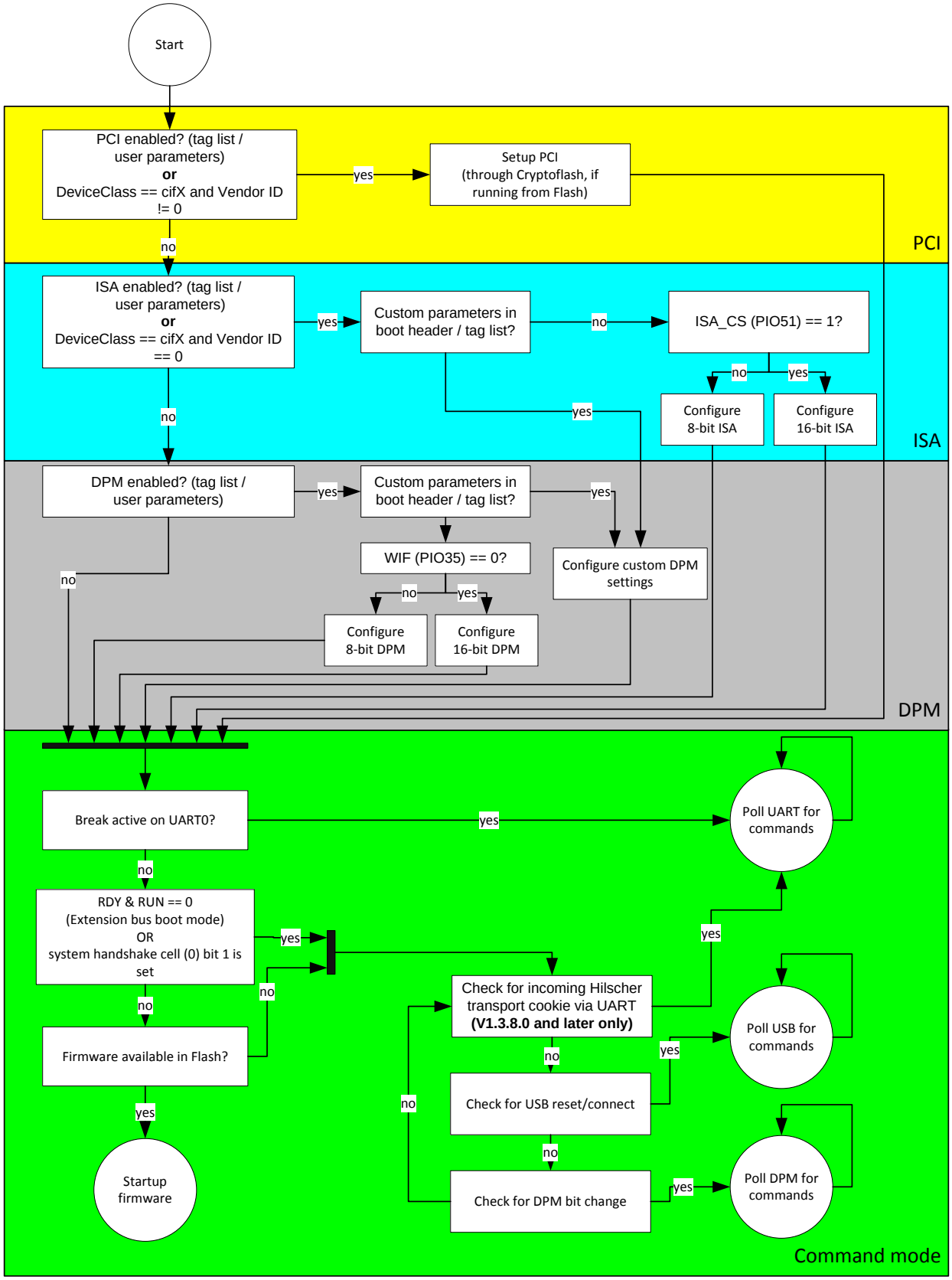


Figure 15: Boot loader startup sequence

2.20 SQIROM / XiP firmware update (netX 10/51/52 only)

As the firmware is unable to update itself in flash, the firmware file is placed in the flash file system and will be copied to the XiP Area on next reboot.

The following sequence diagram defines the procedure:

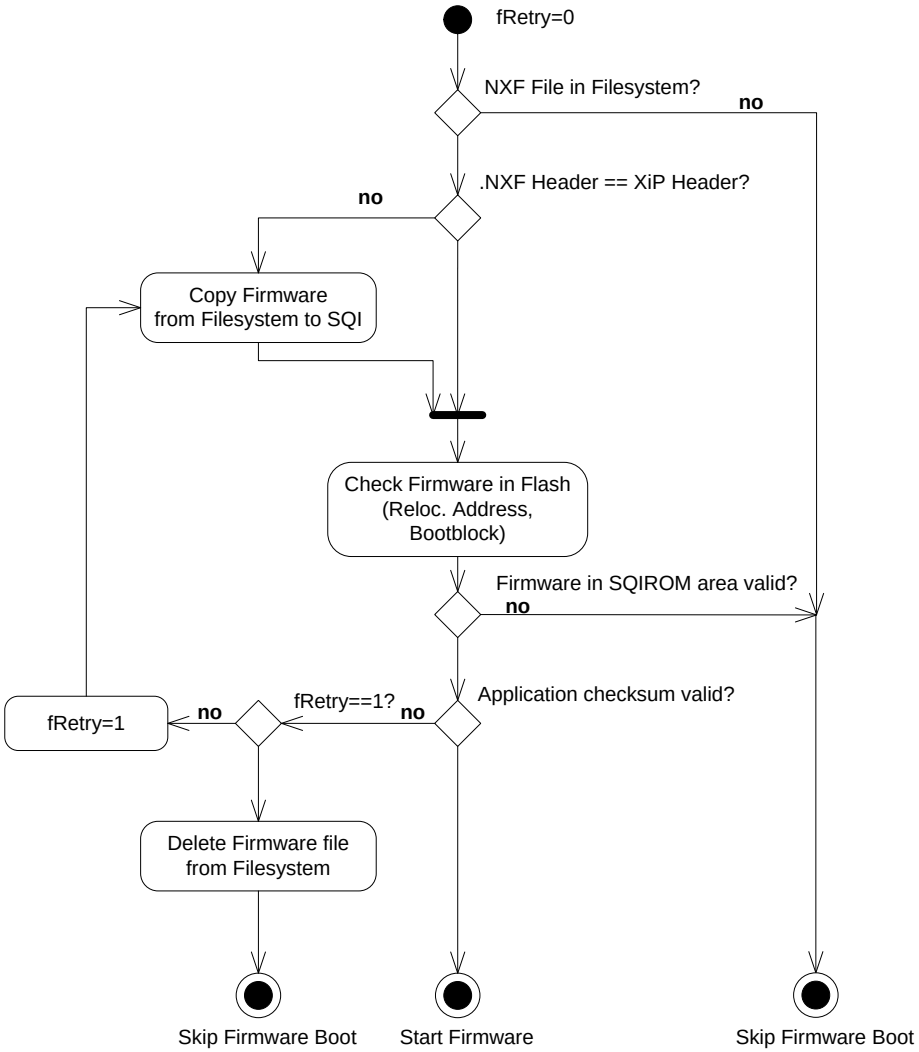


Figure 16: SQIROM / XiP firmware update sequence

The blink code on the system status LEDs will change during the firmware copying:

State	Blink frequency (green->yellow, yellow->green)
Erasing Flash	4 Hz (250 ms)
Copying firmware	16 Hz (62,5 ms)
Default	2 Hz (500 ms)

Table 33: Blink frequency during XIP firmware update

3 Boot loader identification and information passing

Beside the hardware abstraction, the boot loader provides a so called "System Channel" layout inside the Hilscher DPM to allow standard applications the usage of standard functionalities to access a hardware, like done with a Hilscher communication firmware.

The boot loader offers his own identification and functionalities to pass hardware depending data to a firmware.

3.1 Boot loader identification

The boot loader is detectable by a unique identifier. This identifier consists of a name, version and state information.

The boot loader information is placed into the "*System Channel*" system information area of the standard DPM.

Name:	"**BOOT**" (four ASCII characters) Placed into the first four bytes in the DPM instead of "**netX**", signaling the new boot loader is active instead of the existing rcX based boot loader
Version:	Readable via system mailbox (rcX packet)
State:	State information is placed in the existing system information area 0x00C4 = *ulSystemStatus* 0x00C8 = *ulSystemError* 0x00D0 = *bSystemLED*

3.2 Security Flash handling

The Second Stage Boot Loader provides security FLASH information, which is placed in the "*System Channel*" information block. Access to the security data is handled by an xPEC program.

Security information is also passed to the firmware.

3.3 Implemented rcX packets/commands

The *Second Stage Boot Loader* offers an rcX identical packet interface with a reduced functionality.

The following table holds an overview of the implemented commands.

Command	Description
Generic commands	
RCX_FIRMWARE_RESET_REQ	Reset the Second Stage Boot Loader restarting the chip at 0
RCX_FIRMWARE_IDENTIFY_REQ	Query the version information of the Second Stage Boot Loader
RCX_HW_IDENTIFY_REQ	Identifies the hardware
RCX_HW_HARDWARE_INFO_REQ	Query hardware information
RCX_HW_LICENSE_INFO_REQ	Get license code information
RCX_SYSTEM_INFORMATION_BLOCK_REQ	Get DPM system information block
RCX_CHANNEL_INFORMATION_BLOCK_REQ	Get DPM channel information block
RCX_SYSTEM_CONTROL_BLOCK_REQ	Get DPM system control block
RCX_SYSTEM_STATUS_BLOCK_REQ	Get DPM system status block
File access	
RCX_FILE_DOWNLOAD_REQ	Start a file download
RCX_FILE_DOWNLOAD_DATA_REQ	File chunk during download
RCX_FILE_DOWNLOAD_ABORT_REQ	Abort a currently pending download
RCX_FILE_UPLOAD_REQ	Start a file upload
RCX_FILE_UPLOAD_DATA_REQ	File chunk during upload
RCX_FILE_UPLOAD_ABORT_REQ	Abort a currently pending upload
RCX_FILE_DELETE_REQ	Deletes a file on the device
RCX_DIR_LIST_REQ	Queries a directory list on the device
RCX_FILE_GET_MD5_REQ	Queries the MD5 for the given file
RCX_PHYSMEM_READ_REQ	Read physical memory (V1.3.1.0 and later)
RCX_PHYSMEM_WRITE_REQ	Write physical memory (V1.3.1.0 and later)
RCX_FORMAT_REQ	Formats the default volume (V1.3.3.0 and later)
Security memory access	
RCX_SECURITY_EEPROM_READ_REQ	Read the security memory user/non-protected zones
RCX_SECURITY_EEPROM_WRITE_REQ	Write the security memory user/non-protected zones

Table 34: Boot loader command overview

3.4 Boot loader information and parameter passing

The firmware needs additional information like:

- Position of the FLASH file system
- DPM / PCI mode
- Own boot header location

The "DPM / PCI mode" and the "Own header location" are passed in the same way like done by the netX ROM loader (register r0/r1). The position of the FLASH file system will be passed as a structure pointer in register r2 (see chapter 3.4).

Furthermore, the *Second Stage Boot Loader* is also able to pass additional parameters to the started firmware. The parameter passing is an extension to the standard netX ROM loader parameters which are passed through registers r0 and r1.

An additional boot option (passed via r1) was added for the firmware to recognize the presence of a Second Stage Boot Loader.

Boot Option	Description
`BOOTOPTION_2NDSTAGELOADER_FLASH (0x07)`	Firmware was started via the Second Stage Boot Loader that is running from flash (e.g. comX / NXHX) A firmware should jump to the ROM loader it a system start (xSysdeviceReset) is executed.
`BOOTOPTION_2NDSTAGELOADER_RAM (0x08)`	Firmware was started through Second Stage Boot Loader that was downloaded to RAM through DPM/PCI (e. g. cifX) A firmware must not jump into ROM loader during system start (xSysdeviceReset)

Table 35: New boot options (Second Stage Boot Loader only)

3.4.1 Token List

The loader uses a token list for parameter passing which are described in this chapter. Each token starts with a header containing the token type and the total data length. A pointer to the token list is passed in register r2, during firmware startup.

First element in the list is a special cookie (0xAA5511EE) which allows validation checking.

Element	Data type	Description
bToken	BYTE	Token type (0 means end of list)
bLength	BYTE	Length of token data

Table 36: Token header structure

The following token types are defined and explained in the next section.

Token		Description
BOOTTOKEN_ENDOFLIST	(0x00)	Marks the end of the token list
BOOTTOKEN_FLASHDISK	(0x01)	Settings for the flash disk are passed (which resides in the serial flash on CS0)
BOOTTOKEN_RAMDISK	(0x02)	Settings for the RAM disk are passed
BOOTTOKEN_LICENSE	(0x03)	License codes read from a present security memory are passed (for displaying purposes only)
BOOTTOKEN_RAMSIZE	(0x04)	Pass the SDRAM Size in Bytes
BOOTTOKEN_HARDWARE	(0x05)	Pass the hardware data
BOOTTOKEN_SFLASHPARAMS	(0x06)	Serial Flash parameters
BOOTTOKEN_BSLINFO	(0x07)	Second Stage Boot Loader version information
BOOTTOKEN_SQIPARAMS	(0x08)	SQI Flash parameters
BOOTTOKEN_LIBSTORAGE_PARAMS	(0x09)	Storage library parameters

Table 37: Available tokens

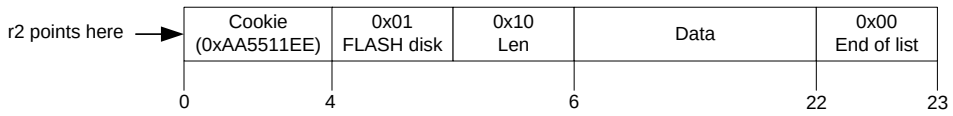


Table 38: Example token list

3.4.2 Flash disk parameters

To allow sharing of the flash disk, inside the serial flash, between the boot loader and the firmware, the start offset and must be known by the firmware.

Token data are preceded by the token type `BOOTTOKEN_FLASHDISK (0x01)`.

Element	Data type	Description
`ulSpiSlave`	DWORD	SPI Chip select (defaults to CS0)
`ulBlockSize`	DWORD	Sector size of the chip/disk
`ulOffset`	DWORD	Start offset in bytes inside the flash
`ulDiskSize`	DWORD	Total disk size

Table 39: Flash disk token data

3.4.3 Storage library parameters

To reserve space for a storage library area on the flash disk, the start offset and size must be known by the firmware.

Token data are preceded by the token type `BOOTTOKEN_LIBSTORAGEPARAMS (0x09)`.

Element	Data type	Description
`ulOffset`	DWORD	Start offset in bytes inside the flash
`ulSize`	DWORD	Total storage library size

Table 40: Storage library token data

3.4.4 RAM disk parameters

To allow sharing of a RAM disk with the boot loader the start offset and must be known to the firmware.

The token data are preceded by a token type `BOOTTOKEN_RAMDISK (0x02)`.

Element	Data type	Description
`ulBlockSize`	DWORD	Sector size of the chip/disk
`ulOffset`	DWORD	Start offset in bytes inside SDRAM
`ulDiskSize`	DWORD	Total disk size

Table 41: RAM disk token data

3.4.5 License code parameters

To allow the displaying of security memory license codes the Second Stage Boot Loader reads the crypted zone of the security flash and passes them to the firmware.

The token data are preceded by a token type `BOOTTOKEN_LICENSE 0x03)`.

Element	Data type	Description
`ulLicenseFlags1`	DWORD	License flags 1
`ulLicenseFlags2`	DWORD	License flags 2
`usNetxLicenseId`	WORD	License ID
`usNetxLicenseFlags`	WORD	License Flags

Table 42: License code token data

3.4.6 RAM size parameters

To allow a firmware to work with different RAM size (i.e. adjust heap size) the SDRAM memory size is passed.

The token data are preceded by a token type `BOOTTOKEN_RAMSIZE 0x04)`.

Element	Data type	Description
`ulRamsize`	DWORD	RAM Size in Bytes

Table 43: RAM size token data

3.4.7 Hardware parameters

To allow a firmware to detect the hardware it is running on during startup, the boot loader passes basic hardware identification from security memory or TAG list.

The token data are preceded by a token type `BOOTTOKEN_HARDWARE 0x05)`.

Element	Data type	Description
`ulFlags`	DWORD	Valid Flags **0x00000001 HARDWARE_FLAGS_DEVICEDATA** - Device data is valid **0x00000002 HARDWARE_FLAGS_HWOPTIONS** - Hardware options are valid **0x00000004 HARDWARE_FLAGS_SLOTNUMBER** - Slot number is valid **0x00000010 HARDWARE_FLAGS_CUSTOM_MMIO_MAPPING** - Custom MMIO mapping set up in 2nd Stage Loader **0x00000020 HARDWARE_FLAGS_HW_ASSEMBLY_FEATURES** - Assembly features are valid **0x00000040 HARDWARE_FLAGS_MACADDRESS** - MAC address is valid **0x40000000 HARDWARE_FLAGS_FLASHLABEL** - Flash label is available **0x80000000 HARDWARE_FLAGS_SECMEM** - Security Memory is available
Device data		

Element	Data type	Description
usManufacturer	WORD	Manufacturer Code
usProductionDate	WORD	Date of production
usDeviceClass	WORD	Device class
bHwRevision	BYTE	Hardware revision
bHwCompatibility	BYTE	Hardware compatibility index
ulDeviceNumber	DWORD	Device number
ulSerialNumber	DWORD	Serial number
Hardware data		
ausHwOptions	WORD[4]	Hardware Options
bSlotNumber	BYTE	Current value of the rotary switch
bBusType	BYTE	External DPM Bus 0: No external bus 1: PCI 2: DPM 3: ISA
bHwAssemblyFeatures	BYTE	Hardware assembly features
abMacAddress	BYTE[6]	MAC address

Table 44: Hardware parameter token data

Note: If the bus type is set to PCI the firmware must **NOT** enable Host Interrupts

3.4.8 Serial Flash parameters

To allow custom serial flashes the Second Stage Boot Loader passes the serial flash information to the firmware.

Token data are preceded by the token type `BOOTTOKEN_SFLASHPARAMS (0x06)`.

Element	Data type	Description
szName[16]	UINT8	Name of the flash (NUL terminated)
ulSize	UINT32	Size of the flash memory in bytes
ulClock	UINT32	Maximum speed in kHz
ulPageSize	UINT32	Size of one page in bytes
ulSectorPages	UINT32	Size of one sector in pages
bAdrMode	UINT8	Addressing mode 0: Linear 1: Bitshift derived from the pagesize
bReadOpcode	UINT8	Opcode for 'continuous array read' command
bReadOpcodeDCBytes	UINT8	Don't care bytes after readOpcode and address
bWriteEnableOpcode	UINT8	Opcode for 'write enable' command, 0x00 means no write protect mechanism
bErasePageOpcode	UINT8	Opcode for 'erase page'
bEraseSectorOpcode	UINT8	Opcode for 'erase sector'
bEraseChipCmdLen	UINT8	Length of the 'erase chip' command, 0 means not available
abEraseChipCmd[4]	UINT8	Command to erase the complete chip
bPageProgOpcode	UINT8	Opcode for 'page program (without built-in erase)'
bBufferFill	UINT8	Opcode for 'fill buffer with data'
bBufferWriteOpcode	UINT8	Opcode for 'write buffer to flash'
bEraseAndPageProgOpcode	UINT8	Opcode for 'page program with built-in erase'
bReadStatusOpcode	UINT8	Opcode for 'read status register'
bStatusReadyMask	UINT8	Bitmask indicating device busy
bStatusReadyValue	UINT8	Xor bitmask for device busy
bInitCmd0_length	UINT8	Length of the first init command in bytes
abInitCmd0[3]	UINT8	First command string to init the device
bInitCmd1_length	UINT8	Length of the second init command in bytes
abInitCmd1[3]	UINT8	Second command string to init the device
bIdLength	UINT8	Length in bytes of the id_send, id_mask and id_magic fields
abIdSend[9]	UINT8	Command string to request the id
abIdMask[9]	UINT8	Mask for the device id
abIdMagic[9]	UINT8	Magic sequence of this device
ulFeatures	UINT32	Bitmask of features 1: Winbond Quad-SPI

Table 45: Serial Flash token data

3.4.9 SQI Flash parameters

To allow custom SQI flashes the Second Stage Boot Loader passes the SQI flash information to the firmware.

Token data are preceded by the token type `BOOTTOKEN_SQIPARAMS (0x08)`.

Element	Data type	Description
fAvailable	UINT8	!=0 if SQI Parameters are available/valid
usFreqMHz	UINT16	Maximum Clock frequency in MHz (must be inside netX Limits)
bAddrBytes	UINT8	Number of address bytes (netX only support 3 address bytes)
bFastReadCmdUINT8	UINT8	Command to enter 0-4-4 mode
bModeClocks	UINT8	Number of mode clocks
bDummyCycles	UINT8	Number of dummy cycles
bQERType	UINT8	Quad Enable Type (see Quad Enable Requirements in SFDP Specification for more details) 1: QE is Bit1 in status register 2 (word accessible) 2: QE is Bit6 in status register 1 3: QE is Bit7 in status register 2 4: QE is Bit1 in status register 2 (byte accessible) 5: QE is Bit1 in status register 2 (split transaction)
bEntryType	UINT8	Sequence to enter 0-4-4 mode 1:Send 0xA5 as last address bytes 2: Set 0xA5 in configuration register
bExitType	UINT8	Sequence to exit 0-4-4 mode 1: Send 0x00 at end of current read 2: Send 0xFF as address

Table 46: SQI Flash token data

3.4.10 Second Stage Boot Loader version information

This token allows the firmware to determine the used Second Stage Boot Loader version on the current hardware.

Token data are preceded by the token type `BOOTTOKEN_BSLINFO (0x07)`.

Element	Data type	Description
usMajor	UINT16	Major version number
usMinor	UINT16	Mionor version number
usBuild	UINT16	Build number
usRevision	UINT16	Revision number
bNameLength	UINT8	Length of the following name string
abName	UINT8[63]	Second Stage Boot Loader name
usYear	UINT16	Build date of the Second Stage Boot Loader
bMonth	UINT8	
bDay	UINT8	

Table 47: Second Stage Boot Loader version information token data

4 Using the boot loader

4.1 LED indication

The boot loader uses the standard *System Status* LED of a Hilscher device to indicate its activity.

State	Indication
Boot loader running	Device System Status LED (SYS LED) flashing 1 Hz, yellow / green

Note: The system LED (SYS LED) blink code is described in the corresponding device manual (default is a yellow/green flashing)

4.2 Configuration

The boot loader can be configured using a tag list.

This configuration makes the boot loader very flexible and adjustable to different hardware configuration and operating conditions.

The following table shows the suggested patching:

DPM based devices	
Second Stage Boot Loader	If no security memory with SDRAM parameters is available, patch SDRAM parameters into the tag list Set custom DPM parameters (IF_CFG0/1, IO_REGMODE0/1) if needed, or set them to 0 for using the default Hilscher settings and the 8/16 Bit DPM mode detection via WIF pin.
Firmware	Only patch SDRAM settings if they do not reside inside security memory, and if they are not patched into the Second Stage Boot Loader. **Recommendation:** Always use a security memory, or the tag list to patch SDRAM parameters, to be able to use the standard un-patched firmware.
PCI devices	
Second Stage Boot Loader	SDRAM parameters must be patched into the tag list to be able to create a RAM disk. **Note:** If the security memory contains valid SDRAM parameters you can skip the patching.
Firmware	SDRAM parameters can be patched, if they are not available from security memory or Second Stage Boot Loader. **Note:** Changing SDRAM parameters between Second Stage Boot Loader and Firmware may destroy RAM contents like RAM disk. **Recommendation:** Patch Second Stage Boot Loader or use a valid security memory.

Note 1: The Second Stage Boot Loader must be flashed to the start of the serial or parallel flash.

Note 2: Make sure to correctly patch serial / parallel parameters to Second Stage Boot Loader when flashing.

The configuration of a firmware or boot loader file can be generated (patched) or changed by a tool named ***Bootwizard*** or ***Tag List Editor***.

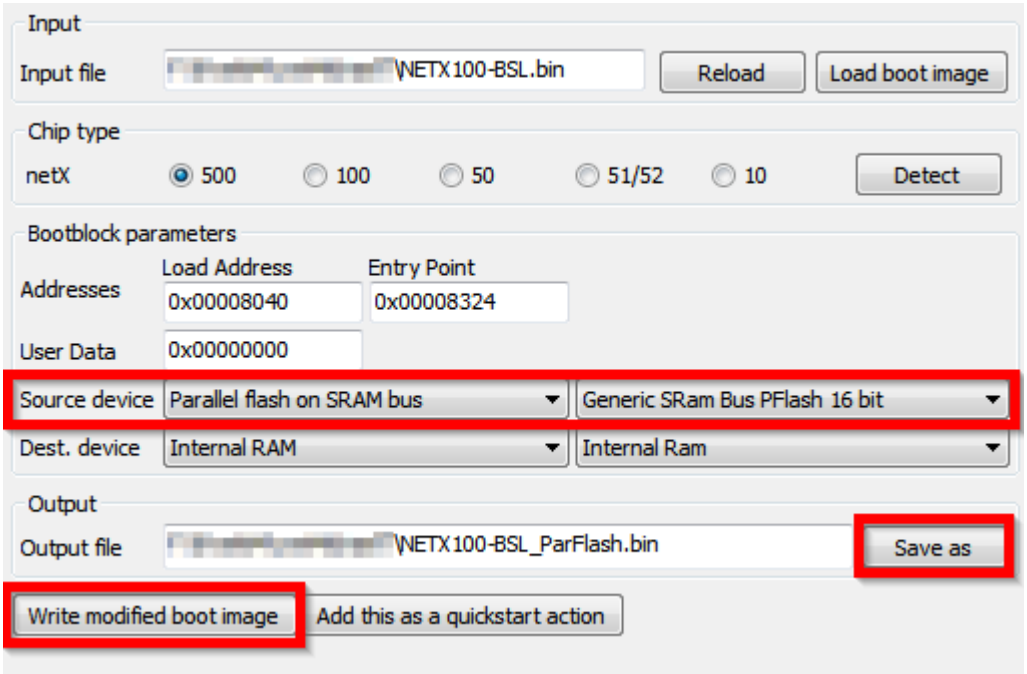
Note: When using *Bootwizard* on an .NXF or boot loader file you will need to run "nxUpdate.exe" for correcting the internal file checksums.

Example updating the binary for booting from other flash devices (e.g. parallel flash):

To be able to start the binary from other flash devices than serial flashes the boot header of the binary file must be updated, otherwise the ROM loader of the netX won't start the 2nd Stage Loader binary.

This is done via the following steps

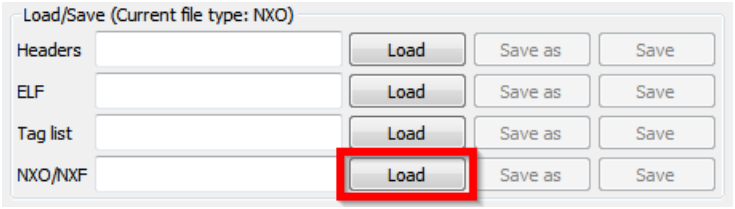
- Open the bootwizard tool
- Select **Modify Image** and load the binary file (e.g. NETX100-BSL.bin)
- Adjust **Source Device**, select the resulting binary location / name and click **Write modified boot image**



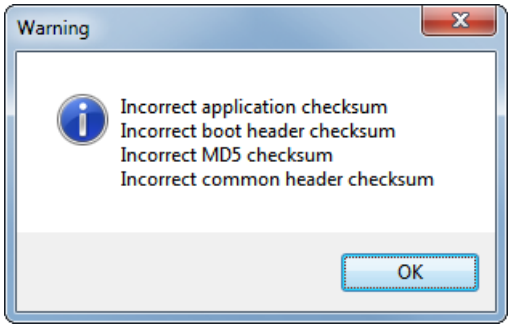
- Update the checksum in the resulting binary file. This can be done using

The Tag List Editor:

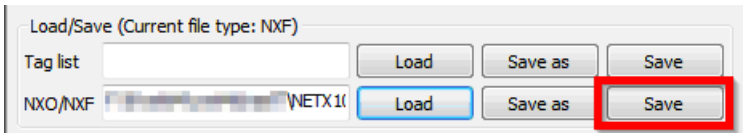
- Load the file as NXO/NXF



- Ignore the warnings about incorrect checksums



- Click **Save** and overwrite the existing file



or

the nxUpdate Tool:

- ren NETX100-BSL_ParFlash.bin NETX100-BSL_ParFlash.nxf
- nxUpdate NETX100-BSL_ParFlash.nxf
- ren NETX100-BSL_ParFlash.nxf NETX100-BSL_ParFlash.bin

4.3 Start-up procedure

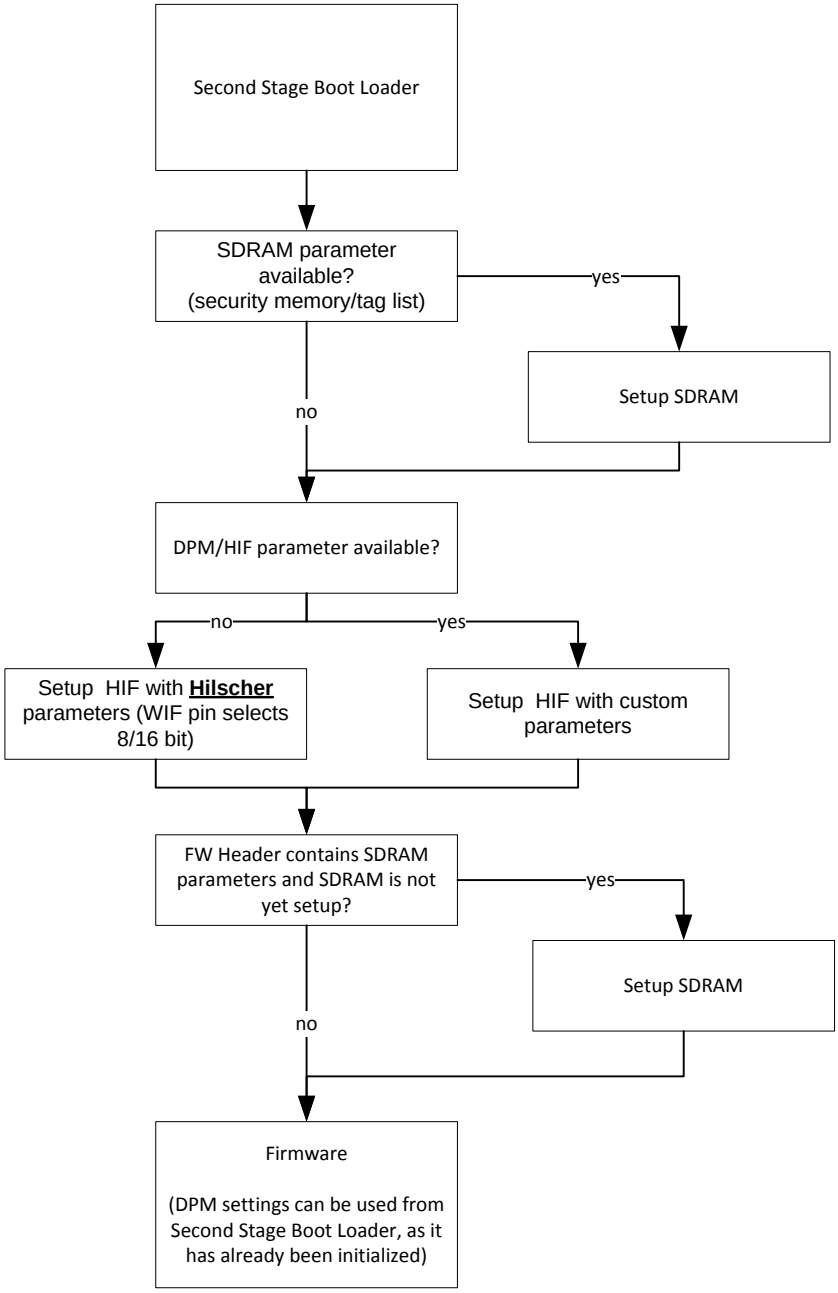


Figure 17: Second Stage Boot Loader startup procedure (DPM mode, with available firmware)

4.4 Firmware validation

The Second Stage Boot Loader (V1.3.0.0 and later) will validate the firmware against the hardware. Hardware data from security memory are verified with the .NXF header contents.

The following figures show the validation algorithm:

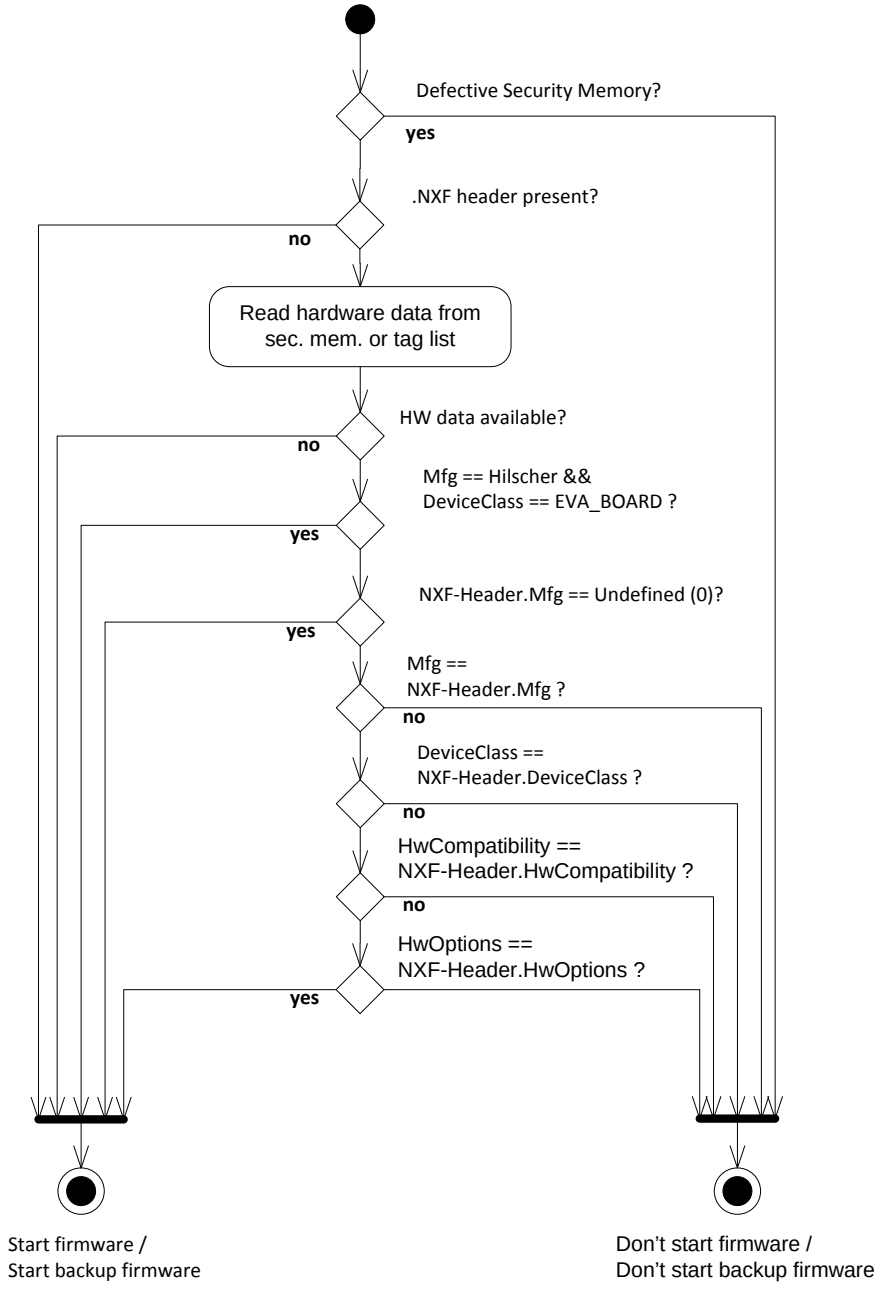


Figure 18: Firmware validation

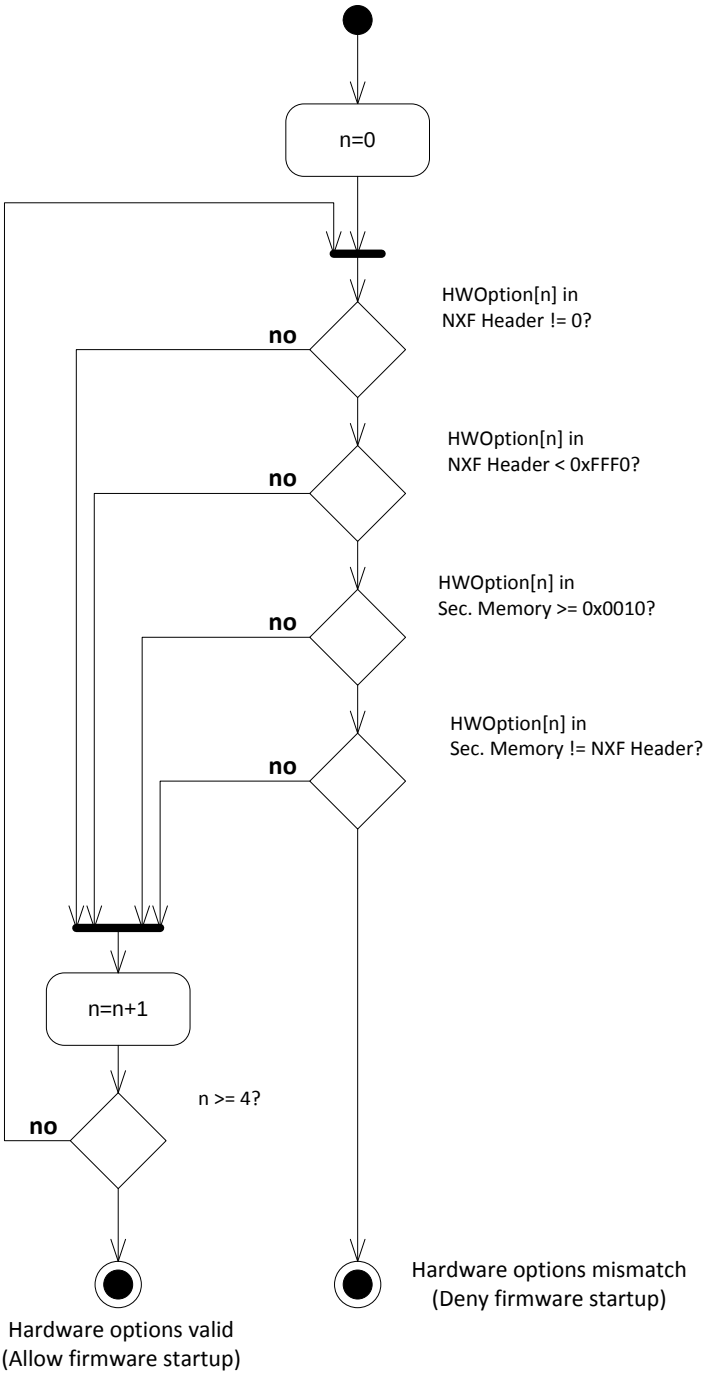


Figure 19: Hardware option validation

4.5 Start-up error detection

If the Second Stage Boot Loader is not able to boot a firmware file residing in parallel flash or in a file system on the serial flash.

Boot loader error codes are written into the "System Status Block" (offset 0xCC) of the DPM.

See Chapter: **7.2 Boot Loader Error Codes**

4.6 Entering boot loader mode

The boot loader provides DPM, serial and USB boot mode. The following chapters describe the necessary steps to enter these modes.

This is needed if a corrupted firmware has been flashed or if the boot loader itself must be updated.

4.6.1 Entering UART boot mode

To enter UART boot loader mode, a break signal must be initiated on the UART serial RxD line during the device / boot loader startup phase. This also suppresses firmware startup. Afterwards the boot loader only listens on the serial port 0.

Since V1.3.8.0 / V1.4.3.0 it is possible to enter the serial boot mode while the Second Stage Boot Loader is running, by sending a Hilscher Transport Packet (e.g. via SYCON.NET). This establishes a UART connection to your device. Once a connection is detected, all other connections won't be checked again.

4.6.2 Entering USB/DPM boot mode

The USB/DPM mode is a shared mode, as there is only one external signal available. The netX ROM loader "extension bus boot mode" is used to signal the Second Stage Boot Loader to enter USB/DPM mode.

Therefore it is necessary to pull the RUN and RDY pin of the netX to ground (via pull down resistors) during the startup of the hardware.

The boot loader will be started from flash and checks the pin states and if detected, it starts polling of the USB and the DPM, until one of the connections is detected.

Note: Once a connection is detected, all other connections are not checked again.

This mode can also be entered through special register settings during power up. Refer to chapter 2.17 for detailed explanation.

5 Tools

Currently two tools are available to support the handling with firmware files and the Second Stage Boot Loader.

5.1 Tag List Editor

To configure the boot loader a tag list is available inside the binary boot loader image. The tag list editor is used to edit these parameters.

5.2 Bootwizard

- The firmware and boot loader files must contain a file header with certain hardware dependent settings. Therefore, the ***Bootwizard*** tool was created, which allows to create (patch) a file header or to change the information in a file header. It can also be used to download binary file to netX based hardware, via a serial or USB connection.

Note: When using Bootwizard for modifying the boot header on an .NXF or boot loader file you will need to run "nxUpdate.exe" for correcting the internal file checksums.

6 Supported FLASH Devices

Since version V1.5 of the boot loader, FLASH detection is changed from an internal fixed list to SFDP (Serial Flash Detection Protocol) allowing to automatically detect connected flash devices, as long as the devices supporting SFDP.

In the past, FLASH devices have to be inserted into the internal boot loader list or as custom parameters, always resulting in a version or software change of the boot loader.

Now the internal FLASH list is only used anymore for backward compatibility with some devices, extensively used in the past and not 100% compatible to the SFDP standard.

The FLASH detections works in 3 stages.

1. Use the boot loader "custom parameter tag list" values
2. Try a JEDEC detection via the internal FLASH compatibility list
3. Use SFDP to detect the FLASH

6.1 Backward Compatibility List

6.1.1 Serial Flash

- Atmel AT45DB321C/D/E
- Atmel AT45DB642D

6.1.2 Serial Flash for netX SQIROM / XiP mode

- Winbond W25Q128
- Winbond W25Q64
- Winbond W25Q32

7 Error Codes

7.1 ROM Loader Error Codes

Error	Name	Description
1	Device Not Allowed	The destination device for the boot image cannot be used in this setup.
2	Cookie Invalid	The boot image has no valid boot cookie.
3	File Signature Invalid	The boot image has no valid file signature.
4	Header Checksum Invalid	The checksum of the boot image header is invalid.
5	Application Checksum Invalid	The checksum of the application data in the boot header is invalid.

7.2 Boot Loader Error Codes

The following table shows the possible error codes and possible reasons:

Error	Value	Description
BOOTERROR_S_OK	0	No Error
BOOTERROR_E_NO_BOOTFILE	1	There was no bootable file found inside the flash disk or parallel flash
BOOTERROR_E_NO_VOLUME	2	No flash disk could be determined by boot loader Possible cause: Defect of the serial flash
BOOTERROR_E_NO_TARGETRAMCONFIG	3	Timing parameters for target could not be determined Possible causes: ▪ No SDRAM Parameters found in security memory or .NXF file ▪ SRAM / Flash parameters in .NXF file are missing
BOOTERROR_E_BOOTHEADER_CHECKSUM	4	Boot header in .NXF file is corrupt.
BOOTERROR_E_APPL_CHECKSUM	5	Application checksum in .NXF file is invalid Possible causes: ▪ Invalid RAM Parameters ▪ Defective .NXF file ▪ Flash error
BOOTERROR_E_FILE_OPEN	6	Error opening file on flash disk Possible cause: Defective flash volume
BOOTERROR_E_FILE_READ	7	Error reading file from flash disk Possible cause: Defective flash volume
BOOTERROR_E_FORCED	8	Command mode of boot loader was forced by user Possible causes: Rdy/Run Pins of netX are configured to Extension bus boot mode *HSF Bootstart* bit was set in DPM
BOOTERROR_E_FIRMWAREMISMATCH	9	Firmware does not match device. (Firmware validation failed)
BOOTERROR_E_FIRMWARESIZEERROR	10	File does not fit in SQIROM (XiP) Area, while trying to update firmware
BOOTERROR_E_FILEWRITE	11	Error copying firmware from to SQIROM (XiP) area
BOOTERROR_E_LICENSE_CHECK	12	Error during license check (netX 100/500 only) e. g. i2c transaction error

Table 48: Start-up error codes in System Status Block

8 Appendix

8.1 Third party software licenses

libfat

fat.h

Simple functionality for startup, mounting and unmounting of FAT-based devices.

Copyright (c) 2006 - 2012

Michael "Chishm" Chisholm

Dave "WinterMute" Murphy

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

8.2 List of tables

8.3 List of figures

8.4 Contacts

Headquarters

Germany
Hilscher Gesellschaft für
Systemautomation mbH
Rheinstrasse 15
65795 Hattersheim
Phone: +49 (0) 6190 9907-0
Fax: +49 (0) 6190 9907-50
E-Mail: info@hilscher.com

Support
Phone: +49 (0) 6190 9907-99
E-Mail: de.support@hilscher.com

Subsidiaries

China
Hilscher Systemautomation (Shanghai) Co. Ltd.
200010 Shanghai
Phone: +86 (0) 21-6355-5161
E-Mail: info@hilscher.cn

Support
Phone: +86 (0) 21-6355-5161
E-Mail: cn.support@hilscher.com

France
Hilscher France S.a.r.l.
69500 Bron
Phone: +33 (0) 4 72 37 98 40
E-Mail: info@hilscher.fr

Support
Phone: +33 (0) 4 72 37 98 40
E-Mail: fr.support@hilscher.com

India
Hilscher India Pvt. Ltd.
Pune, Delhi, Mumbai
Phone: +91 8888 750 777
E-Mail: info@hilscher.in

Italy
Hilscher Italia S.r.l.
20090 Vimodrone (MI)
Phone: +39 02 25007068
E-Mail: info@hilscher.it

Support
Phone: +39 02 25007068
E-Mail: it.support@hilscher.com

Japan
Hilscher Japan KK
Tokyo, 160-0022
Phone: +81 (0) 3-5362-0521
E-Mail: info@hilscher.jp

Support
Phone: +81 (0) 3-5362-0521
E-Mail: jp.support@hilscher.com

Korea
Hilscher Korea Inc.
Seongnam, Gyeonggi, 463-400
Phone: +82 (0) 31-789-3715
E-Mail: info@hilscher.kr

Switzerland
Hilscher Swiss GmbH
4500 Solothurn
Phone: +41 (0) 32 623 6633
E-Mail: info@hilscher.ch

Support
Phone: +49 (0) 6190 9907-99
E-Mail: ch.support@hilscher.com

USA
Hilscher North America, Inc.
Lisle, IL 60532
Phone: +1 630-505-5301
E-Mail: info@hilscher.us

Support
Phone: +1 630-505-5301
E-Mail: us.support@hilscher.com