Firmware datasheet

1 Firmware

This document is the technical datasheet of a communication firmware. It describes the structure, features and interfaces of a loadable communication firmware, running on a netX SoC based device. Each communication firmware consists of several software components. The document lists the technical data and limitations of each component separately. Furthermore 3rd party software components and their licenses are listed in this document.

A Loadable Firmware (LFW) is provided as single binary file (*.nxf, *.nxi) or set of binaray files (*.nxi + *.nxe). Each binary, respectively set of binary files, comes with a dedicated firmware datasheet document.

More information with a higher level of details, can be found in various additional documents. Primarily the Dual Port Memory Manuals and the Protocol API Manuals.

1.1 File information

Firmware	EtherNet/IP Adapter firmware for use case C
File name	X090H001.nxi
Version	V5.4.0.4

Table 1. File information

1.2 Requirements

For operation, the firmware requires the following hardware environment and parameters.

Requirements	Description
ASIC	netX90 (datecode 1910 or later)
Hardware design	Hardware design according to "netX90 Design-In Guide" use case C.
Maintanace Firmware	V1.3.0.x or newer
Device data	Device data provided the FDL - Flash Device Label or via DDP - Device Data Provider mailbox service
MAC addresses	MAC 0: EtherNet/IP and IP communication MAC 1: LLDP communication MAC 2: LLDP communication MAC 3: Ethernet API, if activated

Table 2. Requirements

2 Firmware interfaces and features

The following figure illustrates the internal structure and all available interfaces of the firmware.

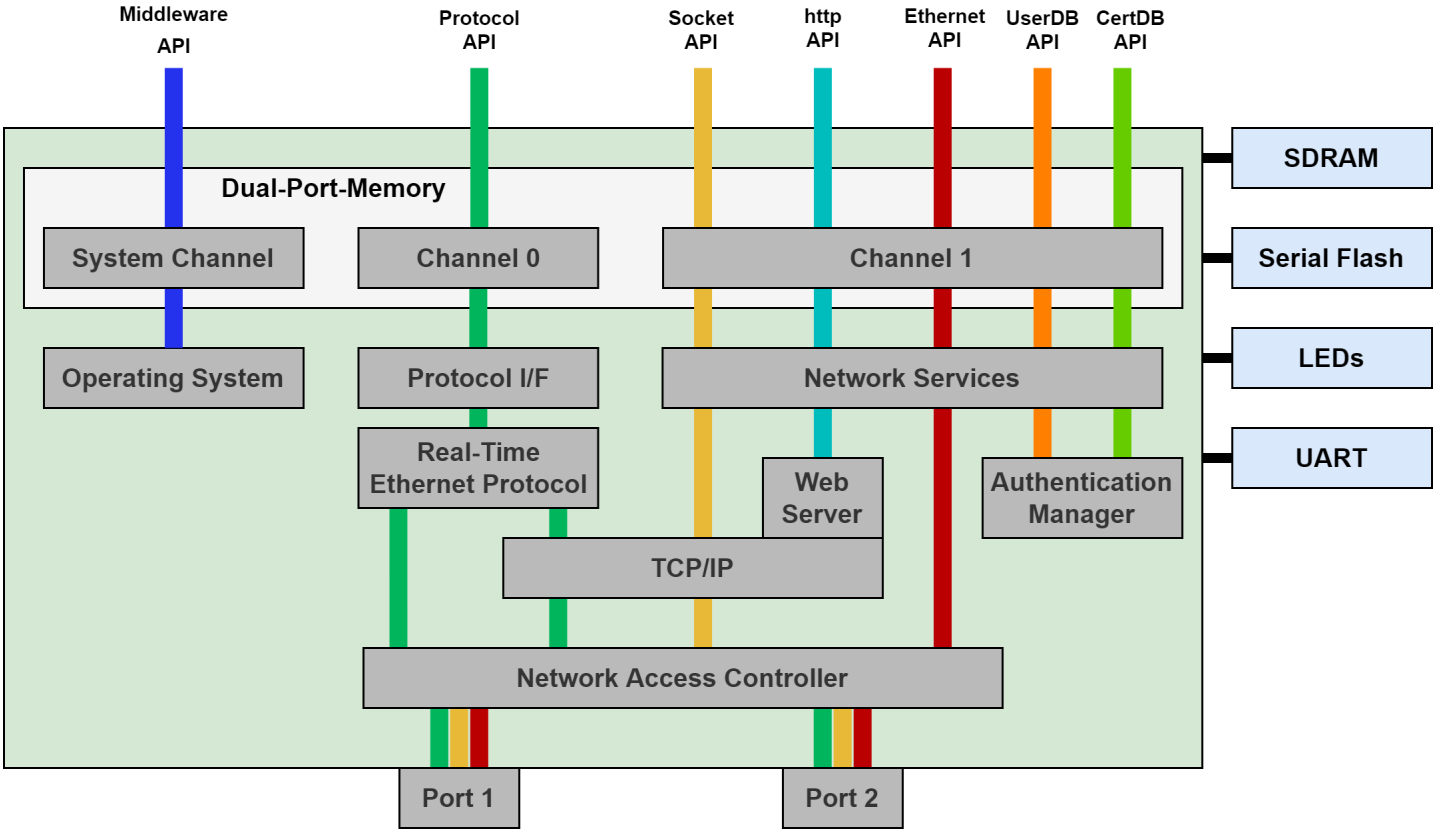

Figure 1. Firmware structure

2.1 Dual-Port-Memory

The Dual Port Memory (DPM), physically located inside the netX SoC, is the central interface between the communication firmware and the user application. It is physically accessed by an external parallel bus, serial SPI interface or an internal data bus. The DPM features a linear address space. The address space is partitioned into several consecutive sections, called "channels". The user application and the communication firmware exchange data, commands and notifications by reading from and writing to the DPM. Each DPM channel is mapped to dedicated firmware functionality and allows usage of respective APIs. You find detailed information about the DPM layout, address spaces and generic services in the Dual Port Memory Manuals.

Channel	API	Manual
System	Middleware API	Manual "netX Dual-Port Memory Interface DPM" and "netX Dual-Port Memory packet-based services"
Channel 0	EtherNet/IP Adapter API	Manual "EtherNet/IP Adapter"
Channel 1	Socket API	Manual "Socket Interface - Packet Interface"
Channel 1	Ethernet API	Manual "Ethernet Interface - Packet Interface"
Channel 1	http API	Manual "Web interface - Packet Interface"
Channel 1	User Database API	Manual "Authentication Manager User Database - Packet Interface"
Channel 1	Certificate Database API	Manual "Authentication Manager Certificate Database - Packet Interface"

Table 3. Dual-Port-Memory layout

2.2 EtherNet/IP

This firmware offers EtherNet/IP Adapter features and an API to access them.

2.2.1 Technical data

The following technical data applies to EtherNet/IP Adapter features.

Feature	Value
Max. number of input data	504 bytes per assembly instance
Max. number of output data	504 bytes per assembly instance
Max. number of assembly instances	32
IO connection types (implicit)	Exclusive owner Listen only Input only
IO connection trigger types	Cyclic Application triggered Change of state
IO connection RPI (O2T / T2O)	min. 1ms, max. 32,767ms* * Depending on the number of parallel connections and sizes of input and output data.
Explicit messages	Connected and unconnected
Unconnected Message Manager (UCMM)	Supported
Max. number of connections	Target Class 0/1: 8 Target Class 3: 8 UCMM: 8
Predefined standard objects	Identity Object (0x01) Message Router Object (0x02) Assembly Object (0x04) Connection Manager (0x06) DLR Object (0x47) Time Sync Object (0x43) QoS Object (0x48) TCP/IP Interface Object (0xF5) Ethernet Link Object (0xF6) LLDP Management Object (0x109)
Max. number of user-specific objects	20
Supported features	TCP/IP, UDP/IP DHCP, BOOTP Address Conflict Detection (ACD) Quality of Service CIP Reset services - Identity Object Reset service of type 0 and 1 QuickConnect LLDP, SNMP (LLDP MIB) Device Level Ring (DLR) - Media Redundancy CIP Sync
Ethernet Speed	10 and 100 MBit/s
Ethernet Duplex Modes	Half Duplex, Full Duplex, Auto-Negotiation
Ethernet MDI Modes	MDI, MDI-X, Auto-MDIX

Feature	Value
Data transport layer	Ethernet II, IEEE 802.3

Table 4. Technical data

2.2.2 Configuration

- by packets (e.g via Dual-Port-Memory mailbox)
- by database (two files named `config.nxd` and `nwid.nxd`) created by Communication Studio or Sycon.NET

2.2.3 Restrictions and limitations

NOT supported are:

- Tags [common mechanism to address typed PLC data using string identifiers]
- CIP Motion
- CIP Safety
  This means the protocol stack itself does not implement the safety application layer. This has to be implemented on the host application side. However, the protocol stack supports all EtherNet/IP features required to build a device that is CIP Safety capable.

2.3 Socket API

The Socket API provides access to the integrated IP stack. Users can implement custom or standard TCP and UDP based protocols on the application side. Both server and client applications are possible. The Socket API has a POSIX socket like interface.

2.3.1 Technical data

The following technical data apply to LWIP component.

Feature	Description
Number of possible parallel active sockets	Default: 8 Can be configured via "Number of sockets for Socket API usage" in tag list from 1 up to 64.
Number of possible parallel Socket API services	Default: 4 Can be configured via "Number of Socket API Services at DPM level" in tag list from 1 up to 8.
Maximum transmission unit (MTU) size	up to 1500 bytes
Protocols	■ IPv4 protocol ■ TCP ■ UDP ■ netident - Hilscher specific protocol to configure IP stack. netIdent can be disabled via "LWIP netident behavior" in tag list.
Socket modes	■ Blocking ■ Non/Blocking
UDP ports	25383 (Hilscher netident)

Table 5. Technical data: Socket API

The maximum values for number of possible parallel active sockets and number of possible parallel Socket API services are configuration parameters in the tag list of the firmware.

Each of these features requires resources and the configuration parameters have to be set in order to not exceed the available resource (e.g. RAM) of a device. The same applies for protocol specific configuration parameters of the tag list as well.

All these configuration parameters compete with each other against the same limited available memory.

2.3.2 Restrictions and limitations

As Socket API is not the main functionality of this firmware, the possible transmission rates will be influenced by higher priority communication tasks and can’t be guaranteed.

- IPv6 protocol is not supported

2.4 Ethernet API

The Ethernet API allows RAW Ethernet frame handling by the user application. It is an independent network node with its own MAC address. While it is connected to the same physical interface, it operates in parallel to the Real Time Ethernet protocol. Typically, the Ethernet API is used on host systems which feature own TCP/IP Stacks, like Linux based application processors.

This functionality needs to be explicitly enabled in the firmware tag list tag "Ethernet NDIS support"

2.4.1 Technical data

The following technical data apply to the Ethernet driver component.

Feature	Description
Maximum frame length	1518 Bytes, starting with first byte of destination MAC address and ending with last byte of data
Size of receive/transmit queue	32 telegrams each
Data transport layer	Ethernet II, IEEE 802.3
Amount of supported multicast MAC addresses	either no multicast or all multicast mac addresses will be forwarded
Supported features	Sending and receiving of Ethernet frames
	Sending and receiving of Ethernet multicast frames (after activating the specific multicast MAC address)

Table 6. Technical data: Ethernet API

2.4.2 Restrictions and limitations

The following general limitations apply:

- The underlying switch in this EtherNet/IP firmware might apply filtering to the frames to protect the EtherNet/IP network from any disturbance due to frames sent or received by the host application. This means that specific multicast or broadcast frames received by the netX may not be forwarded to the Ethernet application. In addition, specific frames generated by the Ethernet application may not be sent to the network.
  For this firmware, the following frames/protocols are known to be blocked (the list may be incomplete):
  - all CIP Transport Class 0/1 messages (UDP Port 2222)
- The size of the receive and transmit queue is limited (see Technical data above). If more frames are received from the network by the switch integrated in the firmware, these frames are silently dropped.
- While handling of multicast MAC addresses it may be possible that frames with unexpected multicast MAC addresses are handed over to the Ethernet application.
- This API is not designed to be used by Ethernet application to implement any real-time capable protocol or application. This is due to its design and internal handling in the netX firmware whose main purpose is always executing the EtherNet/IP Industrial Ethernet protocol.
- Using the Ethernet API will increase the overall CPU usage by the firmware resulting in higher CPU load especially in case of high network load.

2.5 Web server

This firmware has an integrated web server. If desired, the web server can be disabled in firmware tag list via tag "Web server (HTTP) configuration". The web server functionality is described in "Component Description: Web Server + REST API".

2.5.1 Technical data

The data below applies to the web server, featured with the http API, component.

Feature	Description
Integrated Modules	■ GUI - Provides a built-in graphical user interface to other functions ■ Diagnostic Information - Provides information on the netX device as a json object ■ Firmware Update - Enables uploading a new firmware update file to netX flash ■ Update Reset - Initiates a netX reset cycle, i.e. to install a newly uploaded firmware ■ WebIf - Requests are forwarded to the user application via DPM (http API) ■ Serve File - Enables file serving of e.g. custom web content. Located in the Flash file system or tar archive ■ File Manager - Allows file and directory operations, e.g. upload, listing, deletion, etc. ■ User Manager - Allows user accounts creation and modification. ■ Authentication - Allows user account authentication. ■ Redirect - Default redirection of the root URL ("/").
Dispatch Configuration	The dispatch entries map modules to their URLs. They are placed in the order that they are matched for (*1): 1. Serve File ■ WEBSERVER_CFG_DISPATCH_ID_SERVEFILE: /files 2. WebIf ■ WEBSERVER_CFG_DISPATCH_ID_WEBIF: /webif 3. Firmware Update ■ WEBSERVER_CFG_DISPATCH_ID_FWUPLOAD: /netx/firmware 4. Update Reset ■ WEBSERVER_CFG_DISPATCH_ID_RESET: /netx/reset 5. Diagnostic Information ■ WEBSERVER_CFG_DISPATCH_ID_DIAG: /netx/diag 6. File Manager ■ WEBSERVER_CFG_DISPATCH_ID_FILEMANAGER: /netx/filemanager 7. User Manager ■ WEBSERVER_CFG_DISPATCH_ID_USERMANAGER: /netx/usermanager 8. Authentication ■ WEBSERVER_CFG_DISPATCH_ID_AUTHENTICATION: /netx/login 9. GUI ■ WEBSERVER_CFG_DISPATCH_ID_GUI: /netx 10. Redirect ■ / → WEBSERVER_CFG_DISPATCH_ID_SERVEFILE
TCP Port	Default: 80 (http), 443 (https) Can be changed in firmware tag list tag "Web server (HTTP) configuration" and "Web server (HTTPS) configuration" (*2)

Table 7. Technical data: Web interface

(*1) If a dispatch entry's full URL matches the first subpath(s) of other entries it should be placed after them, e.g. the "/netx" URL for the GUI entry is placed at the bottom.

(*2) Changing TCP ports to a port number that is also used by another component inside the firmware, will lead to

undefined firmware behavior and shall be avoided. Setting the TCP port to 0 disables the web server.

2.6 Certificate Database API

The Certificate Database consists of a database containing all the digital PEM/DER-encoded (X.509) certificates and PEM/DER-encoded keys used by the firmware.

The application can use the provided packet interface to generate the required resources for a security component. This includes generating/downloading a private key, generating a CSR, signing/downloading a certificate and more.

2.6.1 Technical data

The following technical data applies to the Certificate Database component.

Feature	Description
Maximum number of resources that can be installed (keys / certificates)	100
Supported encoding formats for certificates and keys	■ DER ■ PEM
Supported key algorithms	■ RSA-2048 ■ RSA-3072 ■ RSA-4096 ■ ECC secp256r1 ■ ECC secp384r1
Maximum certificate (-chain) file size	16Kb
Secure storage of the asymmetric keys	The asymmetric keys are stored in encrypted files using an 128-bits Key Encrpytion Key (KEK), provided as startup parameter on the firmware level. **NOTE** It is recommended to use a unique KEK for each netX instance. E.g. on netX90 use the unique Keys from the secure info page.

Table 8. Technical data: Certificate Database API

2.7 User Database API

The User Database consists of a database of users, including their authentication information (passwords). It is used as a central point for authentication requests within the firmware.

The component provides a packet interface that allows performing different actions on the users. These include authenticating existing users, adding new users, deleting users, changing passwords and more.

2.7.1 Technical data

The following technical data applies to the User Database component.

Feature	Description
Maximum number of users	1 root user + 32 additional users
Default root user	■ Name: "root" ■ Password: "password" **NOTE** It is highly recommended to change the default root user name and password in the final product. Please refer to the component API Manual for more details.
Maximum user name length	28 bytes (including NULL-termination)
Maximum user password length	28 bytes (including NULL-termination)
Minimum user password length	1 byte (without NULL-termination)
Password storage algorithm	PBKDF2

Table 9. Technical data: User Database API

2.8 General firmware features

In addition to protocol and component specific features, which are described in other parts of this document, the following general features are provided by this firmware:

Feature	Description
File system	Fail-save 8.3 FAT like file system
Diagnosis interface	UART Can be disabled via "UART Diagnostics Interface" in tag list.

Table 10. General firmware features

2.8.1 IP ports used by the firmware and its components

The firmware uses the following ports for IP communication by default:

Protocol	Ports
UDP	2222 (EtherNet/IP Class 0/1), 44818 (EtherNet/IP Class 3 and UCMM), 25383 (Hilscher netident), 68 (DHCP client), 161 (SNMP) , 319 and 320 (CIP Sync)
TCP	44818 (EtherNet/IP encapsulation protocol), 80 (http), 443 (https)

Table 11. Summary of IP ports used by firmware

3 Taglist

The Taglist allows users to configure and customize the protocol firmware. The taglist is part of the firmware binary and can be modified by the taglist editor utility.

Name	Description
Remanent Data Responsibility	With this tag, you can adjust whether loading and storing of remanent data is entirely performed by the communication firmware or by the host application.
DDP Mode after firmware startup	With this tag, you can control the DDP mode on firmware startup.
Phy enable timeout after firmware startup	With this tag you can specify a maximum delay until Phys will be activated by firmware.
Ethernet NDIS support	With this tag you can enable NDIS support for the Ethernet API.
UART Diagnostics Interface	UART interface of netX Diagnostics and Remote Access component.
Web server (HTTP) configuration	Sets the number of the TCP port that the web server listens on.
Web server (HTTPS) configuration	Configures web server's HTTPS related parameters.
LWIP Ports for IP 0.0.0.0	With this tag you can enable IP ports for usage with Broadcast communication when IP is 0.0.0.0.
LWIP netident behaviour	With this tag, you can adjust whether the firmware shall activate the Hilscher netident protocol, which is build-in in the IP stack, or not.
Socket API Quantity Structure	adjust the resources allocated and provided by the build-in IP stack.
DLR Protocol	With this tag you can disable Ethernet/IP DLR protocol.

Table 12. Taglists of firmware X090H001.nxi

4 Third Party Components

Component Name	Component URL	License Type	License URL
eCOS Pro	https://ecos.sourceware.org/	Commercial use	Not applicable
lwIP	https://savannah.nongnu.org/projects/lwip/	BSD	https://lwip.fandom.com/wiki/License
mbedTLS	https://tls.mbed.org/	Apache 2.0	https://github.com/ARMmbed/mbedtls/blob/development/LICENSE
MD5	https://github.com/ARM-software/patrace/blob/master/thirdparty/md5/md5.c	proprietary "as-is"	https://github.com/ARM-software/patrace/blob/master/thirdparty/md5/md5.c
newlib	https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads	License types depending on target (not LGPL, not shipping linux targets)	https://sourceware.org/newlib/COPYING.NEWLIB

Table 13. Third Party Components