AH1707

SJA1105Q Evaluation Board

Rev. 1.2 — 13. Feb 2018 **User Manual**

Document information

Info	Content
Keywords	SJA1105P, SJA1105Q, SJA1105Q Evaluation Board, SJA1105Q-EVB
Abstract	The SJA1105Q Evaluation Board is described in this document.

Revision history

Rev	Date	Description
1.0	20171020	Initial version
1.1	20180105	Update to recent HW release, Rotary dial configurations clarified,
1.2	20180213	fixed Fig. 8, note to use FW 1.0.1 for SABRE mode

Contact information

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: mailto:salesaddresses@nxp.com

1. Introduction

The SJA1105Q Evaluation Board demonstrates the capabilities of the SJA1105P/Q Automotive Ethernet switch family in combination with two TJA1102 Automotive Ethernet PHYs.

This user manual focusses on the µC software included on this board and details the features it provides, the control interface, and the reprogramming procedure.

2. Prerequisites

Next to this document you will need the following support material, consult https://www.docstore.nxp.com:

- SJA1105PQRS User Manual (UM11040)
- SJA1105P/Q/R/S Objective Data sheet
- SJA1105PQRS Application Hint (AH1704)
- TJA1102 Datasheet
- TJA1102 Application Hint (AH
- SJA1105PQRS Configuration Generation Tool
- SJA1105x Application Board Host Tools SW
- SJA1105Q Evaluation Board HW Package

3. System Overview

The primary functional components of the SJA1105Q Evaluation Board are shown in Fig 1.

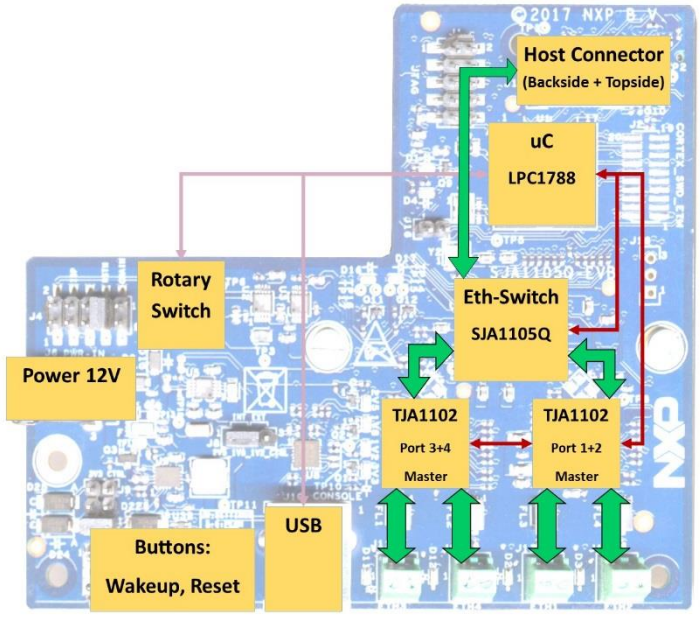

Fig 1. Block diagram indicating the primary functional blocks

3.1 Power

The SJA1105Q Evaluation Board is powered by a single 12 Volt input. It is advised to use an adapter that can deliver at least 1 Ampere of current. The power supply delivered with the board meets that specification.

If the SJA1105Q Evaluation Board is used as a Ethernet board for a host processor, it can also be powered by the host processor via the host connector.

3.2 SJA1105Q

The SJA1105Q is an IEEE 802.3-compliant 5-port automotive Ethernet switch supporting Audio Video Bridging (AVB) and Time-Sensitive Networking (TSN) standards. Each of the five ports can be individually configured to operate at 100Mbit/s or 1000Mbit/s. This arrangement provides the flexibility to connect a mix of PHY devices such as the TJA110x 100BASE-T1 PHYs from NXP Semiconductors and other commercially available Fast Ethernet and Gigabit Ethernet PHYs. The high-speed interface makes it easy to cascade multiple switches of the SJA1105 family for scalability. It can be used in various automotive scenarios such as gateway applications, body domain controllers or for interconnecting multiple ECUs in a daisy chain. Full Audio Video Bridging (AVB)

support means that the SJA1105 family can be used in infotainment and driver assistance systems. Time-triggered Ethernet and Time-Sensitive Networking support with IEEE 802.1QBv time-aware traffic shaping and IEEE 802.1Qci per-stream policing makes the SJA1105Q usable in applications with hard realtime communication requirements.

The SJA1105 family comes in four variants:

Table 1. SJA1105P/Q/R/S variants

Device	SGMII interface	TT-Ethernet and TSN-Ethernet
SJA1105P	No	No
SJA1105Q	No	Yes
SJA1105R	Yes	No
SJA1105S	Yes	Yes

The SJA1105Q Evaluation Board comes with the SJA1105Q variant.

3.3 TJA1102

The TJA1102 is a 100BASE-T1[1] compliant Dual Ethernet PHY optimized for automotive use cases. The device provides 100Mbit/s transmit and receive capability over a single Unshielded Twisted Pair (UTP) cable, supporting a cable length of up to at least 15 m. Optimized for automotive use cases such as IP camera links, driver assistance systems and back-bone networks, the TJA1102 has been designed to minimize power consumption and system costs, while still providing the robustness required for automotive use cases.

The default configuration of the TJA1102 PHYs on the SJA1105Q Evaluation Board are detailed in Table 2, together with the interconnection to the switch ports. Refer to Fig 2 for the placement of the components. The component reference can be found on the board's silk screen.

Table 2. SJA1105Q port usage and default TJA1102 configurations

SJA1105Q port:	Connector	TJA1102:	SMI address:	Master / Slave[2]:	MII mode:
0	-	external host processor	-	-	RGMII
1	J13	U7A	08h	Master	MII
2	J14	U7B	09h	Master	MII
3	J11	U6A	0eh	Master	RMII
4	J12	U6B	0fh	Master	RMII

Please note, that the two PHYs are connected to the switch using different MII modes: MII and RMII. This must be addressed in any SJA1105Q switch configuration to be downloaded to the board.

Please also note that the board firmware does some PHY management, e.g it handles PHY interrupts.

[1] IEEE 802.3bw

[2] In configuration 0 (see section 3.5) the µC firmware periodically switches ports between Master and Slave mode when the port is waiting for a connection.

3.4 Microcontroller and host connector

The local LPC1788 microcontroller runs the board firmware, which enables a computer to control the SJA1105Q and TJA1102 devices via the USB interface. It implements a protocol converter between the serial commands on USB (see ch. 6) to SPI and SMI transactions going to the switch and the PHYs. For the SPI protocol see UM11040, and for the SMI protocol refer to the TJA1102 datasheet.

Besides that, the firmware loads standard configurations selected by the rotary switch. The firmware also does some simple HW checks, e.g. if it can talk to the switch via SPI, etc., and gives some error indication with its LEDs.

When the SJA1105Q-EVB is connected to a host board – either via the SABRE host connector (J15) or via the alternate host connector (J17) – it is considered as a “PHY” for the CPU board’s Ethernet interface. The microcontroller behaves passive and disconnects from the SPI and SMI busses to not mess up the communication between host CPU and switch/PHYs. More precisely, the HW starts up from power on with the host CPU connected to the busses. If the FW detects in an early phase of the booting process, that there is no host CPU, it connects the local CPU to the busses, and continues to boot by initializing switch and PHY.

The SABRE host connector is compatible with the following CPU boards currently available:

- i.MX 6Q SABRE AI CPU Card part number = MCIMX6QAICPU2
- i.MX 6DL SABRE AI CPU Card part number = MCIMX6DLAICPU2
- i.MX 6QP SABRE AI CPU Card part number = MCIMX6QPAICPU3

3.5 Rotary switch

The microcontroller firmware contains built-in configurations that can be selected by the rotary switch provided on the PCB. The µC automatically resets itself when it detects that the rotary switch changes position.

Currently, there is only one configuration.

Table 3. Built-in configurations

Configuration:	Description:
0	Initializes the Switch and the PHYs with a simple configuration: • Ports 1, 2 are set to 100Mbps MII MAC mode; Master-Slave toggle • Ports 3, 4 are set to 100Mbps RMII MAC mode; Master-Slave toggle Further switch configuration details can be found in the SW package in the file example_configs/cfg_applicationBoard.py This configuration setting is intended to make getting started as easy as possible. Therefore, in this configuration the µC firmware toggles the PHY mode between Master and Slave, when there is no established link on that port.
1-F	Reserved configurations. Currently these settings leave the SJA1105Q unconfigured. It can be configured later via the host command interface. However, the PHYs are initialized in all these configurations, and PHY interrupts are always handled by the firmware. For example: link interrupts are always used to set/clear the link LEDs.

Configuration:	Description:
	This can lead to the rare situation that the firmware works against user's PHY settings. Keep in mind: this is primarily a switch evaluation board. There are other solutions for evaluating the TJA1102 PHYs. Configuration F will always leave the SJA1105Q unconfigured. In later firmware releases positions 1-E might contain special switch configurations.

3.6 LEDs

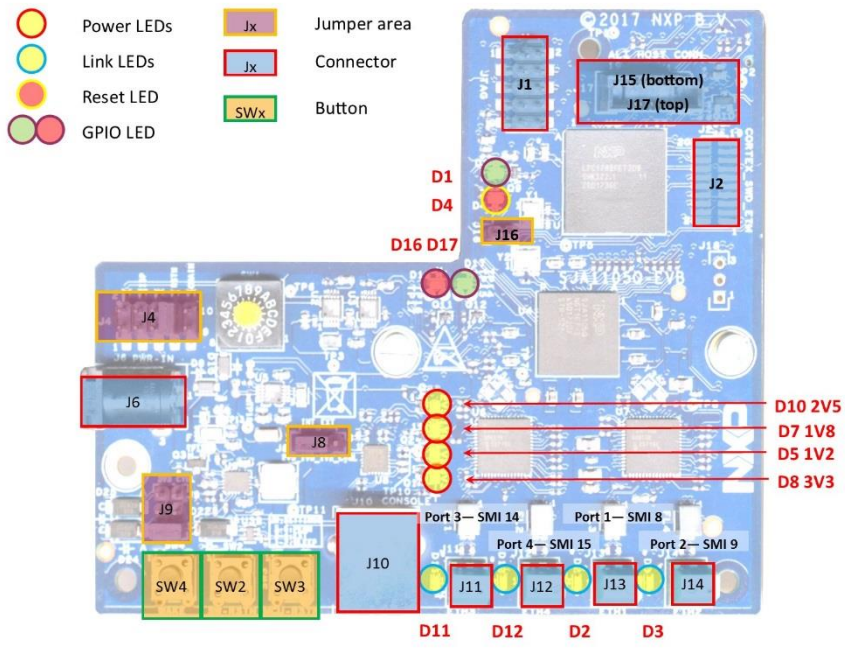

Fig 2. LEDs and Connectors

Table 4. SJA1105Q Evaluation Board LEDs

LED label	Color	Description:
D4	red	Indicates the reset state of the on-board µC.
D1	green	general purpose LED connected to the on-board uC. Used by the µC software as a "heartbeat" or "alive LED". This LED should continuously blink.
D16	red	general purpose LED connected to the on-board uC. Can be used by python scripts to indicate status. "Red LED"
D17	green	general purpose LED connected to the on-board uC. Can be used by python scripts to indicate status. "Green LED"
D2, D3, D11, D12	green	link-up indication LEDs controlled by the on-board uC. Each LED is set/cleared by the LinkUp/LinkFail interrupt of the associated port's PHY.

LED label	Color	Description:
D5	green	power-good indicator for 1.2V rail (Switch)
D7	green	power-good indicator for 1.8V rail (Switch+PHYs)
D10	green	power-good indicator for 2.5V rail (Switch)
D8	green	power-good indicator for 3.3V rail (Processor, Logic)

Positions of the LEDs are shown in Fig 2 and on the board's silk screen.

3.7 USB connector

A USB-to-UART converter (FTDI FT231X[3]) is included on the board to provide an easy connection between a computer and the µC for interacting with the SJA1105Q and TJA1102 devices.

The settings for this serial connection are detailed in Table 5. The provided Python scripting environment is described in Section 5. Low level details of the implemented communication protocol can be found in Section 6.

It is also possible to replace the uC firmware via this USB connector. This procedure is explained in Section 7.

Table 5. Serial port settings for the virtual COM port

Configuration setting:	Value:
Baud rate	115200
Data bits	8
Parity	None
Stop bits	1
Flow control	None

3.8 Buttons

Separate buttons are provided to reset the SJA1105Q ("SW2") or the µC ("SW3").

There is a 3rd button ("SW4") to leave sleep state. See chapter 7.2.

3.9 Jumper blocks and connectors

Positions of the jumper blocks and the connectors are shown in Fig 2.

Table 6. Jumper blocks and connectors

Name	Meaning/Usage
J1	JTAG connector - debugging
J2	SWD, Trace – program development for LPC1788 – connector not mounted
J4	Jumper block and access pins for specific signals 2: PTP clock output 3-4: place jumper for In-System-Programming of new firmware 5: wakeup signal for PHYs 6: reserved 7-8: Keep local CPU in reset (e.g. host CPU operation) 9-10: JTAG-Chain-Select; place jumper for SWD debugging.

[3] http://www.ftdichip.com/Products/ICs/FT231X.html

Name	Meaning/Usage
J6	Power Jack 12V input
J8	enable 1V2, 1V8, 2V5 power rails 1-2: internal control (standalone operation) 2-3: external CPU board (host CPU operation)
J9	Enable for 3V3 power rail comes from: 1-2: PHY-INH signal (enable PHY controlled sleep/wakeup) 3-4: external CPU board (host CPU operation) 5-6: 12V from power jack (standalone operation) A jumper must be placed in one of the 3 positions
J10	USB Type B receptacle for the serial link to PC
J11..J14	100BASE-T1 cable screw blocks
J15	host CPU connector (back side of PCB)
J16	Jumper to separate the PHYs' wakeup signal from the host CPU connector
J17	alternate host CPU connector (top side of PCB) with limited capability compared to J15

There is a description of typical use cases for the board in chapter 7. The correct jumper settings for each use case are listed there.

4. Getting Started

This section will provide a quick introduction on how to setup and use the SJA1105Q Evaluation Board in *standalone mode*. Other operation modes can be found in chapter 7.

4.1 Unpack the host software

The host SW package is supplied as a .zip file. The package is available at NXP's DocStore web portal *https://www.docstore.nxp.com*.

Unpack it at a place you like.

The SW package relies on a working installation of Python 2.7. For a Windows machine, we recommend Python-XY, available from *https://python-xy.github.io/* Make sure the path of the Python package is contained in the %PATH% variable. In case you already have the old SJA1105 software installed, %PATH% contains a link to the Python environment contained with the old software. The new SW package is known to not work with it, so uninstall the old SJA1105 software.

The host SW package provides example scripts, which cover the fundamental basis of serial communications as well as configurations of the SJA1105Q Evaluation Board.

A view of the installation directory structure is shown in Table 7.

Table 7. Installation directory structure

Directory / File:	Description:
.	Python scripts to control and program the SJA1105Q Evaluation Board over the serial USB. The scripts can be called from the Windows command prompt (DOS-Box).
firmware	Default LPC microprocessor firmware
switch_configurations	Default python script for creating SJA1105Q configurations The python script "download_config" from the example_com directory can be used for downloading a HEX file. As download_config.py uses a fixed filename rename your created HEX file. Note: In order to generate the hex file from the python file, the "Configuration Generation Tool" must be installed separately (DocStore).
modules	Additional python libraries used under the hood.

4.2 Install FTDI device driver

To access the board via USB under Windows, a device driver from FTDI must be installed. This device driver provides a virtual COM port to be used by the Python tools. The driver can be downloaded from *http://www.ftdichip.com/FTDrivers.htm*

This is not further covered in this application hint.

4.3 Board setup

The steps required to attach the SJA1105Q Evaluation Board to a computer are:

- Connect a USB cable between the USB port and the computer

- Connect the first 100BASE-T1 port to a computer via a media converter (100BASE-T1 to 100BASE-TX)[4]. The media converter should be set to slave mode as this port is set to operate as a master node by default.
- Set the rotary switch to position 0.
- Check that the jumpers are set according to the following table:

Table 8. Jumper settings for standalone operation

Jumper	Setting
J4	none installed
J8	1-2
J9	5-6
J16	not installed

- Connect a 12 Volt adapter to the power input jack.
- Reset the local CPU by pressing the corresponding reset button SW3.
- Observe a short flash from the reset LED (D4), and a blinking pattern on D2, D3, D11, D12. When D1 (alive LED) is blinking at a constant, slow pace the board is ready for host communication.

See Fig 3 for a connection diagram.

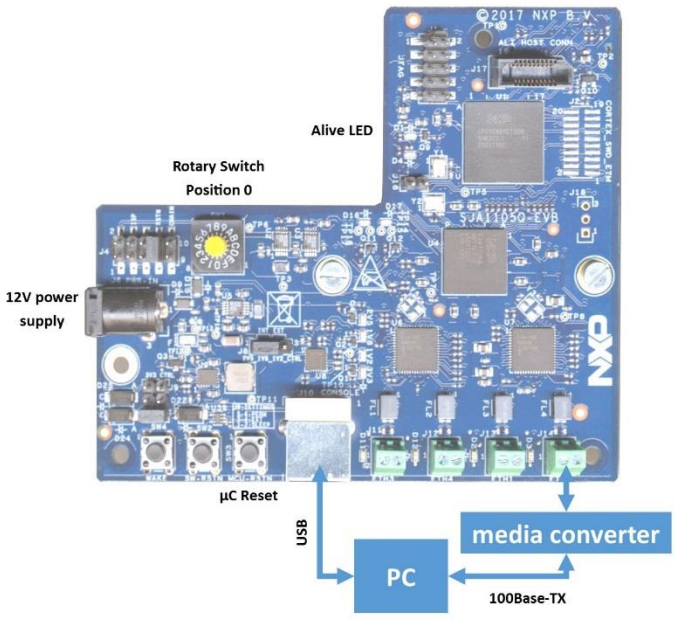

Fig 3. Connection diagram

[4] An alternative to a media converter is the USB-to-100BASE-T1 stick from Fibrecode Embedded Solutions. See http://fibrecode.com/usb-oabr-broadr-reach-100base-t1-stick.html

4.4 Determining the serial port

The USB connector of the SJA1105Q Evaluation Board is connected to a USB-UART converter and therefore the virtual COM port driver (see 0) will assign a serial (or COM) port number to the connection.

The example Python scripts detect the COM port automatically: they probe all COM ports and try to query the switch device ID from a presumably connected SJA1105Q-EVB board. If a valid device ID is found, this COM port will be used.

In some cases, COM port probing cannot be used, e.g. because it disturbs other connected devices, or multiple boards must be controlled from the same PC. To disable automatic COM port detection, the actual COM port must be explicitly stated in the .py script. Substitute “auto” by “COM11” (or whatever the system assigned to the port) in the following code line, which can be found in every example Python script:

```
try:
    platform = uart_platform.UartPlatform("auto")
except Exception as e:
    print e
    exit(0)
```

Fig 4. How to disable COM port auto detection

On a Windows based machine the COM port can be determined using the Device Manager application (see Fig 5). It is also possible to change the serial port number by right clicking on the current serial port number and subsequently navigating to Properties → Advanced → COM Port Number.

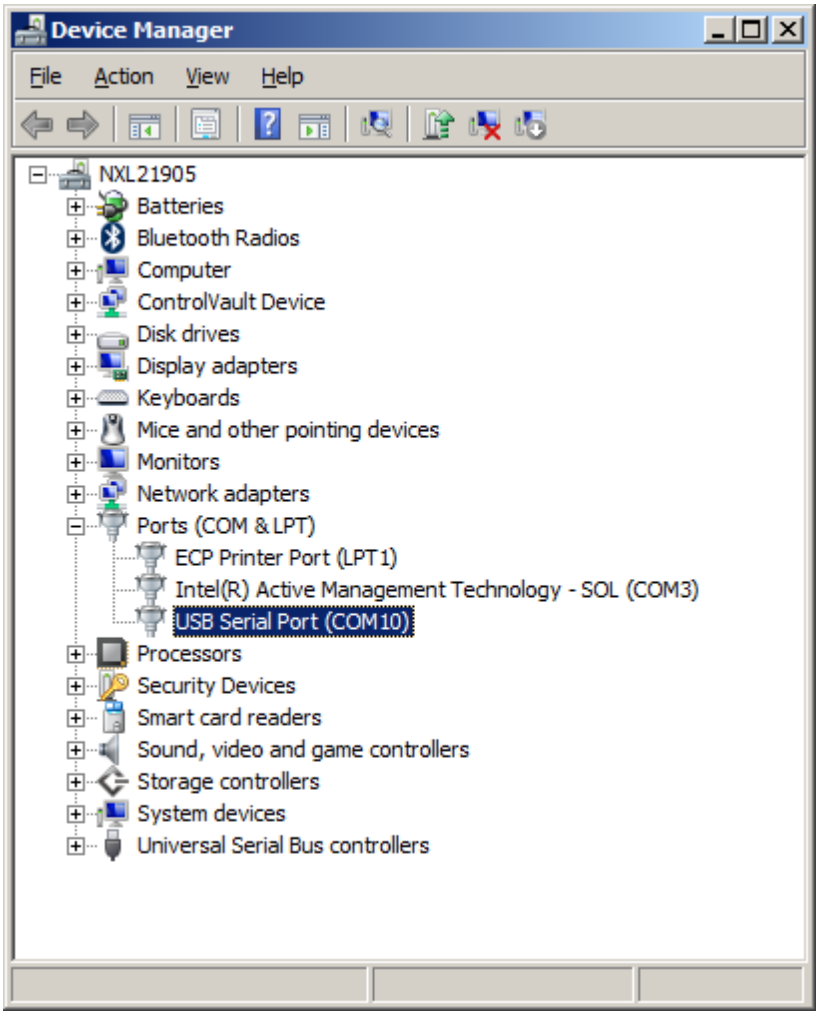

Fig 5. Windows device manager

4.5 SJA1105Q communication

The “read_deviceid.py” script gives an example on how to read register values from the SJA1105Q. It reads and prints the revision number of the µC software and the device identifier of the SJA1105Q.

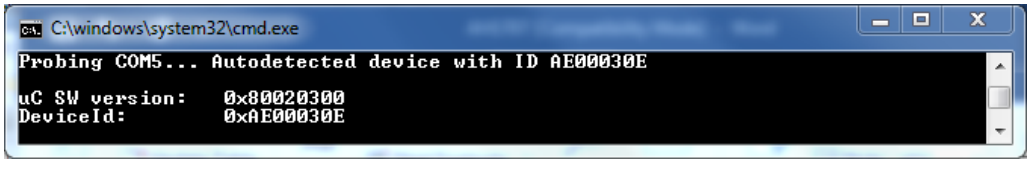

Fig 6. read_deviceid.py

4.6 TJA1102 communication

The “probe_phy.py” script from the directory “example_com” reads the PHY identification registers for all possible SMI addresses and prints them to the screen. The SMI addresses used by the TJA1102 PHYs on the SJA1105Q Evaluation Board are 8, 9, 14, 15. The PHY visible on the broadcast address 0 is caused by the fact, that NXP PHYs react on broadcast read requests.

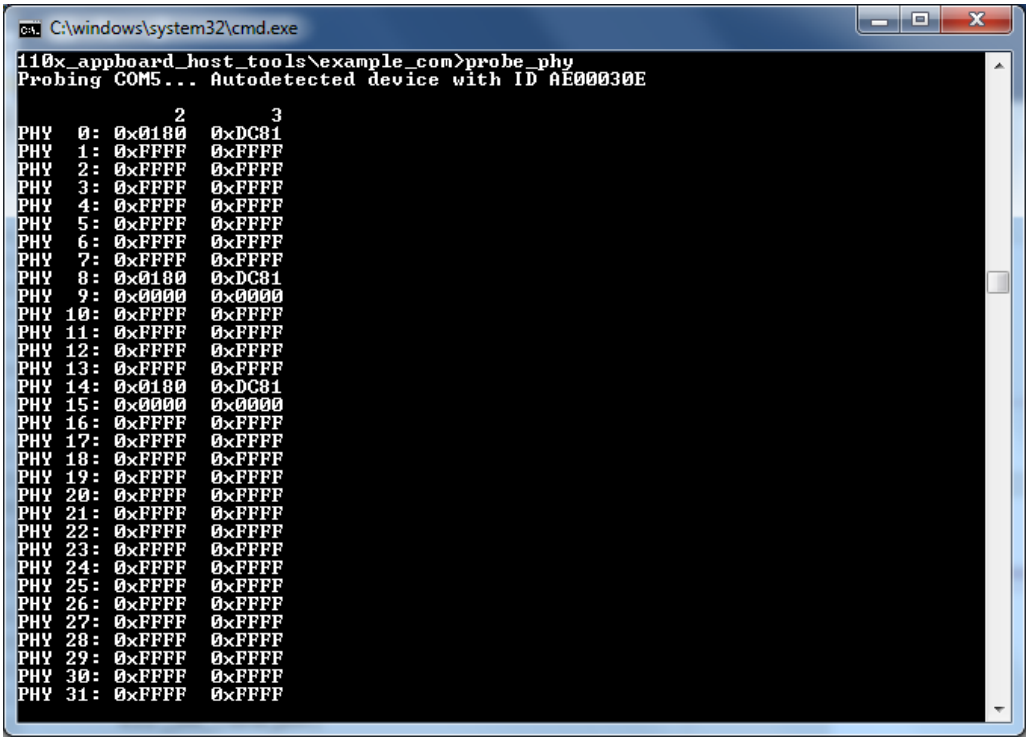

Fig 7. probe_phy.py

4.7 Example Python scripts

As a starter and to showcase some switch features the following scripts are provided with the SW package:.

Table 9. Scripts from example_com\pyScripts

Script:	Description:
read_deviceid.py	Reads the DEVICEID register of the SJA1105Q and the version of the local processor application software.
probe_phy.py	Reads the PHY identifier registers (register 2 and 3) of all possible PHY addresses
dump_phyregs.py	Reads all registers of the four PHYs on the SJA1105Q-EVB.

Script:	Description:
read_l2_address_lookup_table.py	Reads the L2 address lookup table and displays valid entries (static and dynamic)
write_l2_address_lookup_table.py	Writes a new L2 address table entry.
read_vlan_lookup_table.py	reads the VLAN lookup table and displays valid entries. This script takes some time, because there are 4096 entries which are read and checked for validity.
write_vlan_lookup_table.py	writes a new entry to the VLAN lookup table.
read_l2_management_lookup_table.py	Reads L2 management route table. The table is initially empty (no static configuration)
write_l2_management_ lookup_table.py	Writes a new L2 management table entry.
read_highlevel_diagnostic_registers.py	Reads Rx and Tx diagnostic registers for all switch ports
download_config.py	Example script that loads the new switch configuration "cfg_applicationBoard.hex" which is expected in the example_com directory.
goto_sleep.py	sends the entire board, including the local µC to sleep. On Wakeup, the µC performs a powerup boot. Special jumper settings are necessary for this to work. See chapter 7.2

4.8 Creating a new SJA1105Q configuration

Creating a new configuration for the SJA1105Q involves the following steps:

1. Obtain the "Configuration Generation Tool" from www.docstore.nxp.com and extract to a custom directory (e.g. in switch_configurations)
2. Create a copy of one of the provided example configuration sources and rename it as desired. There is a minimalistic example “simplePQRS”, a more elaborate example “examplePQRS” and an example making use of the TSN features, like CBS, policer, MAC filtering for gPTP, etc. in “examplePQRS_TSN”.
3. Modify the remainder of the configuration source as desired.
4. Save the configuration source.
5. Open cmd.com (DOS box) and navigate to the switch_configurations directory
6. Execute the newly created configuration source file making use of the Python environment
7. A new configuration in the Intel HEX file format should appear with the name of the new configuration file.

It is possible to test the new configuration by copying it to the root directory of the Evaluation board tools and calling the download_config.py script. The default filename in the script must be exchanged by the name of the new .hex file. The script will reset the switch and download the newly created switch configuration.

5. Python scripting

5.1 Python support files

Besides the example scripts already mentioned, the SW package also contains supporting Python modules and libraries. These can be used to build own scripts.

Table 10. SW package contents

module	Description:
modules/uart.py	communication platform via UART to the SJA1105Q-EVB uC. The methods implemented in this module are the foundation of the scripting interface described in section 5.2.
modules/prettytable.py	printing switch configuration tables in a nice ASCII form
modules/progress.py	a simple text based progress bar visualization
modules/scanwin32.py	for Windows: query COM ports from the registry

5.2 Scripting interface

The file modules/uart.py implements the scripting interface for accessing the board via the serial interface. This public interface and its methods are described in this section.

5.2.1 __init__(serial_port)

Initializes the Uart_Platform object.

Table 11. Arguments of __init__(serial_port)

Argument:	Description
serial_port	Either “auto” for automatic port detection, or the name of the serial port to which the SJA1105Q Evaluation Board is connected (e.g. ‘COM22’ on a Windows computer).

5.2.2 spi_config(config_file)

Configures the SJA1105Q with the configuration contained in an Intel HEX-file.

Table 12. Arguments of spi_config(config_file)

Argument:	Description:
config_file	The path to the aforementioned Intel HEX-file containing the desired configuration for the SJA1105Q.

5.2.3 spi_write(address, data)

Writes a value to a SJA1105Q register defined by the argument address.

Table 13. Arguments of spi_write(address, data)

Argument:	Description:
address	The address of the register to write to. The range of valid addresses is 0x0000_0000 to 0x001F_FFFF.
data	The unsigned 32-bit integer data value to write to the register.

5.2.4 spi_read (address)

Returns the value of a SJA1105Q register (unsigned 32-bit integer) with the given address.

Table 14. Arguments of spi_read(address)

Argument:	Description:
address	The address of the register to read from. The range of valid addresses is 0x0000_0000 to 0x001F_FFFF.

5.2.5 spi_read_print(address)

Reads the value of a SJA1105Q register defined by the address and prints this value to the standard output as a formatted string.

Table 15. Arguments of spi_read_print(address)

Argument:	Description:
address	The address of the register to read from. The range of valid addresses is 0x0000_0000 to 0x001F_FFFF.

5.2.6 smi_write(phy, address, data)

Writes to an SMI register inside one of the PHYs,

Table 16. Arguments of smi_write(phy, address, data)

Argument:	Description:
phy	The address of the PHY. The range of valid PHY addresses is 0 to 31.
address	The address of the register to write to. The range of valid addresses is 0 to 31.
data	The unsigned 16-bit integer data value to write to the register.

5.2.7 smi_read(phy, address)

Reads from an SMI register inside one of the PHYs and returns this value (unsigned 16-bit integer).

Table 17. Arguments of smi_read(phy, address)

Argument:	Description:
phy	The address of the PHY. The range of valid addresses is 0 to 31.

Argument:	Description:
address	The address of the register to write to. The range of valid addresses is 0 to 31.

5.2.8 smi_read_print(phy, address)

Reads from an SMI register inside one of the PHYs and prints this value to the standard output as a formatted string.

Table 18. Arguments of smi_read_print(phy, address)

Argument:	Description:
phy	The address of the PHY. The range of valid PHY addresses is 0 to 31.
address	The address of the register to write to. The range of valid addresses is 0 to 31.

5.2.9 reset()

Resets the SJA1105Q.

5.2.10 get_version()

Returns the µC software revision number (unsigned 32-bit integer).

Table 19. Version number elements

Bits	Contents	Description
31	legacy bit	compatibility bit
30..24	feature bitset	bitset for extra features contained. Not used here.
23..16	product type	03h for SJA1105Q Application Board
15..12	major version	
11..8	mid version	
7..0	minor version	

An example for a µC software version number is 0x80031000: V1.0.0 for SJA1105Q-EVB RevB.

5.2.11 set_leds(red, green)

Turns the red and green GPIO LEDs (D16, D17) on or off.

Table 20. Arguments of set_leds(red, green)

Argument:	Description:
red	turns the Red LED on (True) or off (False)
green	turns the Green LED on (True) or off (False)

6. Serial commands and responses

The various commands accepted by the µC software are detailed in this section.

6.1 Basic structure

The interface implemented by the µC software is based on the command-response pattern. The computer will send a command to the µC and, if the command is accepted, a response is returned.

Each command is delimited by the '<' and '>' characters. Responses are delimited by the '[' and ']' characters.

Each command and response is identified by a 4 ASCII character string (e.g., 'LEDR'). Command and response arguments are also ASCII encoded according to these conversion rules:

- Boolean values are converted to 'T' or 'F' character to indicate a true or false value respectively.
- Unsigned integer values are encoded into a sequence of ASCII characters that match the hexadecimal representation of the value. Unsigned 8-bit (e.g., 0x12 → '12'), 16-bit (e.g., 0x9abc → '9ABC') and 32-bit (e.g., 0x12345678 → '12345678') values are supported.

6.2 Read µC software revision number

This command read the revision number of the µC software.

Command: <VERS>

Response: [VERSaaaaaaaa]

Table 21. Arguments of the VERS command and response

Argument:	Description:
a	The revision number of the µC software (32-bit unsigned integer).

6.3 SJA1105Q reset

This command enables resetting the SJA1105Q from software.

Command: <PRSTa>

Response: [PRSTa]

Table 22. Arguments of the PRST command and response

Argument:	Description:
a	Boolean value controlling the SJA1105Q reset line: assert ('T') or de-assert ('F').

6.4 SJA1105Q register write

This command writes to a register inside the SJA1105Q.

Command: <SPIWaaaaaaaabbbbbbbb>

Response: [SPIWaaaaaaaabbbbbbbb]

Table 23. Arguments of the SPIW command and response

Argument:	Description:
a	The register to write to (32-bit unsigned integer).
b	The value to write (32-bit unsigned integer).

6.5 SJA1105Q register read

This command reads from a register inside the SJA1105Q.

Command: <SPIRaaaaaaaa>

Response: [SPIRaaaaaaaabbbbbbbb]

Table 24. Arguments of the SPIR command and response

Argument:	Description:
a	The register to read from (32-bit unsigned integer).
b	The value read (32-bit unsigned integer).

6.6 SJA1105Q low level SPI access

This command implements a low-level interface to the SJA1105Q. Its primary use is for writing configurations in a more efficient way.

Command: <SPIX(aaaa)+>

Response: [SPIX(aaaa)+]

Table 25. Arguments of the SPIX command and response

Argument:	Description:
(aaaa)+	One or more 16-bit unsigned integer values to transfer on the SPI interface.

6.7 PHY register write

This command writes to a register inside an Ethernet PHY.

Command: <SMIWaabbcccc>

Response: [SMIWaabbcccc]

Table 26. Arguments of the SMIW command and response

Argument:	Description:
a	The SMI address of the PHY (8-bit unsigned integer, range 0 to 31).
b	The register to write to (8-bit unsigned integer, range 0 to 31).
c	The value to write (16-bit unsigned integer).

6.8 PHY register read

This command reads from a register inside an Ethernet PHY.

Command: <SMIRaabb>

Response: [SMIRaabbcccc]

Table 27. Arguments of the SMIR command and response

Argument:	Description:
a	The SMI address of the PHY (8-bit unsigned integer, range 0 to 31).
b	The register to read from (8-bit unsigned integer, range 0 to 31).
c	The value read (16-bit unsigned integer).

6.9 LED control

This command controls the Red or Green LED.

Command: <LEDab>

Response: [LEDab]

Table 28. Arguments of the LED command and response

Argument:	Description:
a	'R' for Red LED, 'G' for Green LED
b	'T' for 'ON', 'F' for 'OFF'

7. Board Use Cases

The SJA1105Q-EVB can be used in various scenarios and use cases besides the standard standalone operation mode described in the previous chapters:

7.1 Standard standalone operation mode

In this mode, the application in the local µC controls the board, its initialization and its behavior. User access is accomplished via the USB serial link to a host computer.

The required cabling and jumper settings are shown in Fig 3 and Table 8.

7.2 Standalone Operation mode with sleep support

This mode showcases the capability of the TJA1102 PHYs to send the entire network or parts of it to sleep and wake it up again via user interaction or via network communication.

For remote wakeup capability, nodes featuring the TJA1102 PHY must be connected at least to one Ethernet port of the board[5].

Use the following jumper settings:

Table 29. Jumper settings for sleep mode capable standalone operation

Jumper	Setting
J4	none installed
J8	1-2
J9	1-2
J16	not installed

[5] SJA1105Q-EVB RevA has no wakeup button. Local wakeup is therefore not possible.

The script “goto_sleep.py” demonstrates, how to put the board to sleep: After writing the sleep request into the PHY registers, the sleep request is sent to the connected nodes via PHY negotiation protocol, and then the board’s power-supply (now under control of the PHYs due to Jumper J9) is switched off.

Once a connected node sends a wakeup signal to one of the PHYs, or the user presses wht WAKE button, the power supply is switched on again, and the local µC starts due to power being available again.

7.3 Communication daughter board for SABRE CPUs

The SJA1105Q-EVB features a host CPU connector (J15, bottom side), which allows it to be used as a communication card for certain i.MX and S32 computing boards. The connector and the geometry of the SJA1105Q-EVB are compatible to the SABRE[6] standard.

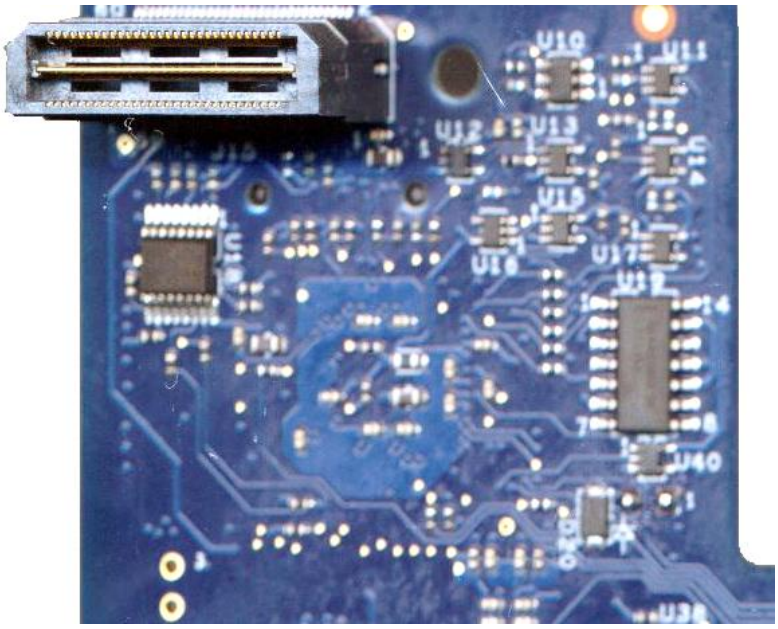

Fig 8. Host CPU connector (SABRE)

[6] SABRE = Smart Application Blueprint for Rapid Engineering – a standard endorsed by NXP

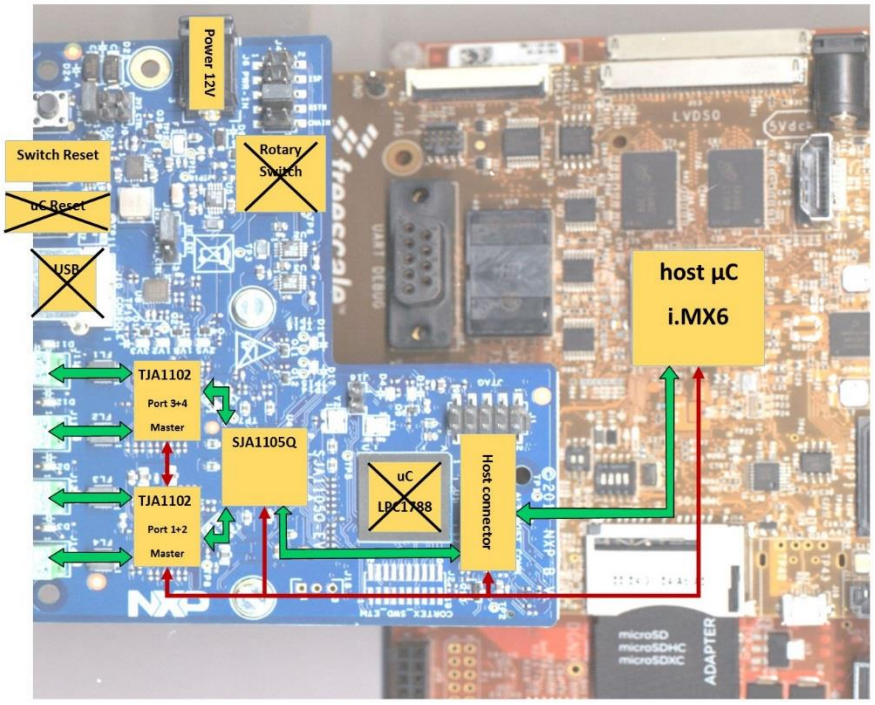

Fig 9. SJA1105Q-EVB plugged onto a i.MX6 system

In this application, the SJA1105Q-EVB is considered a communication slave card from the host CPU's perspective. The local µC is inactive, and the connected host CPU controls the switch and the PHYs. Besides, the host CPU's Ethernet port is connected to port0 of the SJA1105Q switch via RGMII with 1000MBit (fixed speed).

Table 30. Jumper settings for use as a communication daughter board

Jumper	Setting
J4	normally no special setting[7] necessary because of daughter board mode.
J8	2-3
J9	3-4
J16	On Rev-A: DO NOT INSTALL – could damage the host board

Some host boards supply the 12V for operating the SJA1105Q-EVB. If in doubt, connect a 12V power supply to J6 as in standalone operation.

The local CPU detects in an early boot phase, if it is plugged to a host CPU board. It does not continue to boot, but gives the host CPU exclusive access to SPI and SMI busses. It also blinks slowly D12, which is an indication for being in daughter board mode.

Please note, that for using the SJA1105Q in SABRE operation firmware version 1.0.1 and up is required. Currently, only FW version 1.0.0 is programmed in the factory. You can download the latest board firmware from https://docstore.nxp.com

[7] On Rev-A: install J4 7-8.

7.4 Communication daughter board for SABRE CPUs and sleep support

This use case is like the standalone mode with sleep support (ch. 7.2), except that the host processor is also sent to sleep.

This mode is supported on Rev-B boards and newer. Only newer host boards support the HW signals for sleep and wakeup.

7.5 Controlling switch and PHY from customer's processor

In situations, where a customer processor board must be used to control the switch and the PHYs, there is a simplified host connector on the top side of the SJA1105Q-EVB. Similar to the SABRE connector on the bottom side it gives access to the SPI and SMI bus, but it lacks the RGMII connection to the switch.

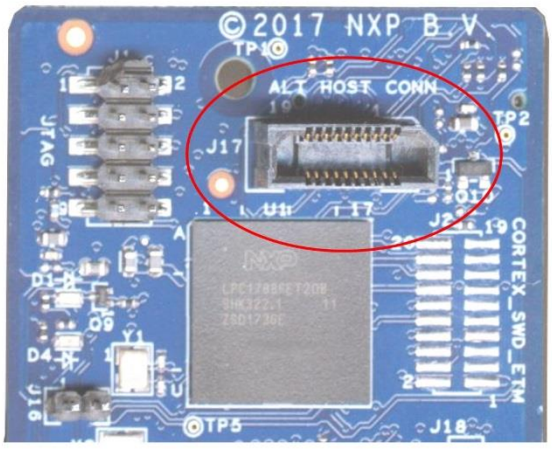

Fig 10. Alternative host CPU connector

The connector on the board is a Samtec ERF8-010-07.0-S-DV-K-TR. It mates with various connectors or cable assemblies[8].

Table 31. Pinning of the alternative host connector

Pin:	Name:	Description:
1	SPI MOSI	SPI bus to the switch; host processor is master
2	SPI MISO	SPI bus
3	SPI SCK	SPI bus
4	SPI_CS	SPI bus
5	VCC_12V0	Voltage supply for the SJA1105Q-EVB from the host processor board as an alternative to the barrel connector J6
6	n.c.	

[8] consult the connector web page at https://www.samtec.com/products/erf8

7	MDIO	SMI bus to the PHYs
8	MDC	SMI bus
9	I2C_SDA	Access to port extender MAX7322 for board identification at I2C 7bit-Address 0x68
10	INH_3V3	Wired-OR Inhibit signal from the PHYs for entering sleep mode. See TJA1102 data sheet.
11	I2C_SCL	port extender access; see Pin 9.
12	WAKE_IN_OUT	Wakeup signal from host processor to the board (PHYs)
13	RESET_N	Reset signal from host processor to the switch
14	CARD_DETECT	Must be connected to GND on the host processor side.
15	VREG_INH	Host processor can hold up going to sleep by pulling this line high.
16	n.c.	
17	n.c.	
18	n.c.	
19	GND	
20	GND	

8. Board behavior

8.1 Diagnostics

On startup, the local µC application performs some checks and outputs blink codes if it detects an error.

Table 32. Diagnostic blink codes

blink code	meaning
D1 blinking	normal operation, all checks passed
D11 slow blink	communication daughter board mode – for Rev-A install J4 7-8
D2, D3, D11, D12 fast rotating pattern	boot in progress
D2 blinking	trying to fetch the deviceid from the switch
D3 blinking	deviceID successfully read, but does not match expected value for a SJA1105Q
D4 permanent light (red)	µC is in reset, usually all other LEDs are also permanently on.

8.2 Automatic Master-Slave adaption

In rotary switch position “0” the firmware periodically toggles the PHY mode between Master and Slave for ports without an established link. This is to make it easier to initially bring up the board.

8.3 Reset behavior

The SJA1105Q switch chip can be reset by pressing the reset button SW2 (see Fig 2). This clears the configuration of the switch, so it must be configured again. When loading

a new switch configuration with set_config() (see 5.2.2), an implicit switch reset is performed by the python host tools. Other than that, an implicit switch reset is only performed during power up initalisation of the µC software.

The switch reset does not affect the PHY settings, as the PHY reset line is not connected to the switch reset.

The local µC can be reset by pressing button SW3 (see Fig 2). Other causes of a µC HW reset are power supply issues, including initial power up, or placing a jumper on J4 (7-8). During the reset initialization, the PHYs are queried. They are initialized only when they have detected a power-on situation. This is to preserve the configuration in the PHY in case of a wakeup scenario caused by a remote Ethernet device.

9. In-system programming

The µC software can be reprogrammed via the serial port.

9.1 Installation

- Install the Flash Magic application (http://www.flashmagictool.com/download.html).
- Use the “Advanced Options…” item in the “Options” menu to open the tabbed pane with advanced options. The controls in the second tab should be changed to conform to Fig 11.

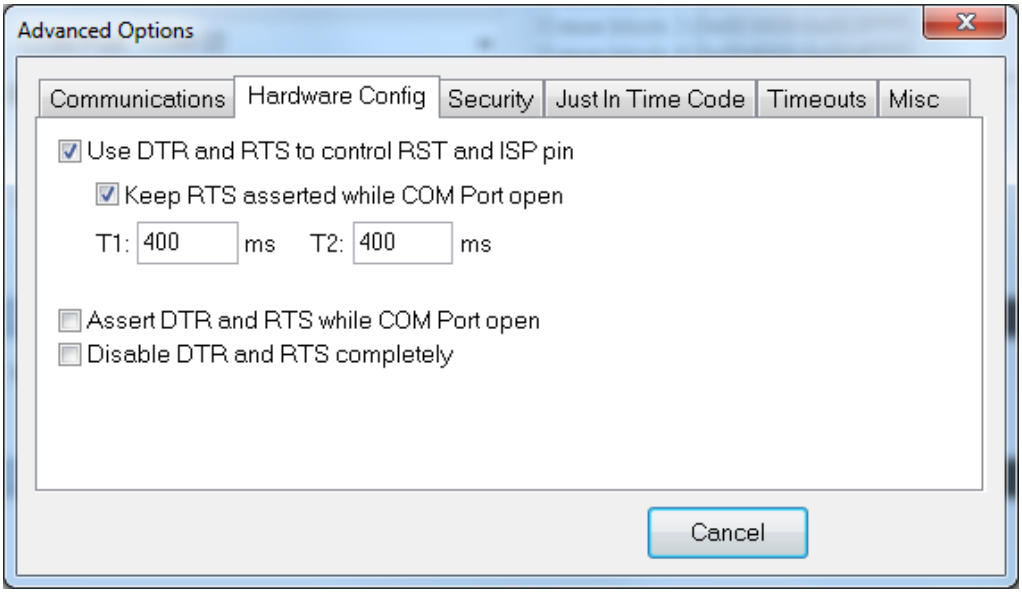

Fig 11. Flash magic advanced options

9.2 Preparation

The following actions are required to prepare the SJA1105Q Evaluation Board for reprogramming:

- Supply power to the board.

- Connect a USB cable between the board and the computer.
- Place a jumper to short pins 3 and 4 of connector J4.

9.3 Programming

The main user interface of the Flash Magic application is divided into 5 numbered steps and the actions per step are:

Step 1:

- Select LPC1788 microcontroller.
- Select the serial port used by the SJA1105Q Evaluation Board (see 4.4 for how to do this)
- Set the baudrate to 57600.
- Set interface to “None (ISP)”.
- Set the oscillator to 12 MHz.

Step 2:

- Select “Erase blocks used by Hex File”.

Step 3:

- Select the Intel Hex file to program.

Step 4:

- Optionally select “Verify after programming”.

Step 5:

- Click the Start button.

10. Abbreviations

Table 33. Abbreviations

Acronym	Description
µC	Microcontroller
ARP	Address Resolution Protocol
AVB	Audio/Video Bridging (IEEE 802.1BA)
IC	Integrated Circuit
ICMP	Internet Control Message Protocol
ISP	In-System Programming
LED	Light Emitting Diode
MII	Media Independent Interface
OABR	OPEN Alliance (IEEE 802.3bw) BroadR-Reach; aka 100BASE-T1
PHY	Physical layer
RMII	Reduced Media Independent Interface
SMI	Serial Management Interface, a.k.a. Media Independent Interface Management
SPI	Serial Peripheral Interface

Acronym	Description
TSN	Time Sensitive Networking
TT	Time-triggered
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

11. Legal information

11.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

11.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

11.3 Licenses

Purchase of NXP <xxx> components
<License statement text>

11.4 Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — owned by <Company name>

11.5 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

<Name> — is a trademark of NXP Semiconductors N.V.

12. List of figures

13. List of tables

14. Contents