User documentation – S1901 EvoTRAC

Table of contents

Product description

Table of contents

The EvoTRAC ™ S1901 system platform features an Intel ® high performance processor. It is designed to meet the future needs of in-vehicle for Artificial Intelligence, Deep Learning and HPEC by providing two high performance GPU or accelerator card options. The platform is built to accelerate the deployment of demanding applications made for heavy-duty mobile machinery operating in the construction, agriculture or mining industries. This compact platform's flexibility is unmatched as it includes, in a single IP67 enclosure, high speed I/Os , USB 2.0 and 3.0, SMA connectors for RF and configurable digital I/Os. Storage requirements are met using M.2 NVMe slots or high capacity 2.5-inch SSD slots (removable). Furthermore, it is possible to stay connected with the platform's optional Wi-Fi and/or LTE cellular modem features.
The platform offers monitoring functionalities. When the platform includes a mezzanine with a safety MCU, clients have access to monitoring features enabled by the AURIX ™ safety MCU. These features can be leveraged by clients to trigger actions that will bring the platform to a state in which failures have no effect on the system. The hardware actions that can be triggered include the following:

- Reboot or disable the CPU
- Reboot or disable the network switch
- Disable CAN bus communication

Main applications

- COM Express® Intel® High Performance computer-on-module technology
- Expansion slots for multiple GPUs, FPGA accelerators or video modules
- Multiple I/O high speed interfaces and high capacity storage options
- Wireless connectivity options for LTE/5G cellular and Wi-Fi (optional)
- In-vehicle deployment with built-in AI

Main features

- Ruggedized and modular platform
- Three selectable cooling options
- DC input power
- COM Express® Intel® High Performance processor
- Seven Ethernet ports (10/100/1000 MbE)
- Two Ethernet ports (one at 2.5 GbE and one at 10 GbE)
- PTP/1588 with 1PPS, GNSS
- Built-in support for:
  - Up to four CAN buses
  - Up to four CAN buses with an AURIX ™ safety MCU
- Two MXM slots for GPU, FPGA-based or specialized processing modules
- Up to two optional hot-swappable 2.5-inch drives
- Three external USB 2.0
- Up to two optional display ports with USB 3.0
- Four RS-232 ports, two RS-485/422 ports and two configurable RS-232 or RS-485/422 ports
- Nine GPIs, eight GPOs and seven GPIOs with isolated ground
- Two balanced audio In and two balanced audio Out
- Other functionalities: one Ignition key switch, one Power button, one Reset button, one thermistor input
- Expansion slots that can be used for cellular connectivity, Wi-Fi connectivity, storage, I/O, Automotive Ethernet and more:
  - Two M.2 M key expansion slots
  - One M.2 B key expansion slot
  - Two mini-PCIe slots
- 9 axis IMU
- Support for Linux Ubuntu 18.04 and 20.04 LTS 64-bit operating systems

Revision history

Revision	Brief description of changes	Date of issue
1.0	First product release version 0 EFT	April 2021
2.0	Preliminary version for rev 1	October 2021
2.1	Preliminary version for rev 1 - corrections made to voltages	November 2021
2.2	Preliminary version for rev 1 - various corrections made throughout the document	March 2022
3.0	Supplemented preliminary version for rev 1 • Mechanical changes to risers • Front panel changed to a tray • USB #1 port no longer available	December 2022
3.1	General update EvoTRAC	July 2023
4.0	CAN bus mezzanine with the AURIX safety MCU added to the platform Automotive Ethernet capacities added to the platform mPCIe Dual CAN bus added to the platform	June 2024

Warranty, disclaimer and support

Table of contents

Limited warranty

Please refer to the full terms and conditions of the Standard Warranty on Kontron's website at:
https://www.kontron.com/support-and-services/rma/canada/standard_warranty_policy_canada.pdf .

Disclaimer

Kontron would like to point out that the information contained in this manual may be subject to alteration, particularly as a result of the constant upgrading of Kontron products. This document does not entail any guarantee on the part of Kontron with respect to technical processes described in the manual or any product characteristics set out in the manual. Kontron assumes no responsibility or liability for the use of the described product(s), conveys no license or title under any patent, copyright or mask work rights to these products and makes no representations or warranties that these products are free from patent, copyright or mask work right infringement unless otherwise specified. Applications that are described in this manual are for illustration purposes only. Kontron makes no representation or warranty that such application will be suitable for the specified use without further testing or modification.
Kontron expressly informs the user that this manual only contains a general description of processes and instructions which may not be applicable in every individual case. In cases of doubt, please contact Kontron.

Customer support

Kontron's technical support team can be reached through the following means:

- By phone: 1-888-835-6676
- By email: support-na@kontron.com
- Via the website: www.kontron.com

Customer service

Kontron, a trusted technology innovator and global solutions provider, uses its embedded market strengths to deliver a service portfolio that helps companies break the barriers of traditional product lifecycles.
Through proven product expertise and collaborative, expert support, Kontron provides unparalleled peace of mind when it comes to building and maintaining successful products. To learn more about Kontron's service offering—including enhanced repair services, an extended warranty, and the Kontron training academy—visit www.kontron.com/support-and-services .

Safety and regulatory information

Table of contents

NOTICE	Before working with this product or performing instructions described in the getting started section or in other sections, read the Safety and regulatory information section pertaining to the product. Assembly instructions in this documentation must be followed to ensure and maintain compliance with existing product certifications and approvals. Use only the described, regulated components specified in this documentation.

General instructions on usage

In order to maintain Kontron's product warranty, this product must not be altered or modified in any way. Changes or modifications to the product, that are not explicitly approved by Kontron and described in this user guide or received from Kontron Support as a special handling instruction, will void your warranty.
This product should only be installed in or connected to systems that fulfill all necessary technical and specific environmental requirements. This also applies to the operational temperature range of the specific product version that must not be exceeded. If batteries are present, their temperature restrictions must be taken into account.
In performing all necessary installation and application operations, only follow the instructions supplied by the present user guide.
Keep all the original packaging material for future storage or warranty shipments. If it is necessary to store or ship the product, then re-pack it in the same manner as it was delivered.
Special care is necessary when handling or unpacking the product.

High risk applications hazard notice

THIS DEVICE AND ASSOCIATED SOFTWARE ARE NOT DESIGNED, MANUFACTURED, OR INTENDED FOR USE OR RESALE FOR THE OPERATION OF NUCLEAR FACILITIES, NAVIGATION, CONTROL OR COMMUNICATION SYSTEMS FOR AIRCRAFT OR OTHER TRANSPORTATION, AIR TRAFFIC CONTROL, LIFE SUPPORT OR LIFE SUSTAINING APPLICATIONS, WEAPONS SYSTEMS, OR ANY OTHER APPLICATION IN A HAZARDOUS ENVIRONMENT, OR REQUIRING FAIL-SAFE PERFORMANCE, OR IN WHICH THE FAILURE OF PRODUCTS COULD LEAD DIRECTLY TO DEATH, PERSONAL INJURY, OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE (COLLECTIVELY, "HIGH RISK APPLICATIONS").
You understand and agree that your use of Kontron devices as a component in High Risk Applications is entirely at your risk. To minimize the risks associated with your products and applications, you should provide adequate design and operating safeguards. You are solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products. You are responsible to ensure that your systems (and any Kontron hardware or software components incorporated in your systems) meet all applicable requirements. Unless otherwise stated in the product documentation, the Kontron device is not provided with error-tolerance capabilities and cannot therefore be deemed as being engineered, manufactured, or setup to be compliant for implementation or for resale as device in High Risk Applications. All application and safety related information in this document (including application descriptions, suggested safety measures, suggested Kontron products, and other materials) is provided for reference only.

General safety warnings and cautions

Your new Kontron product was developed and tested carefully to provide all features necessary to ensure its compliance with electrical safety requirements. It was also designed for a long fault-free life. However, the life expectancy of your product can be drastically reduced by improper treatment during unpacking and installation. Therefore, in the interest of your own safety and of the correct operation of your new Kontron product, you are requested to respect the following guidelines.

	ESD sensitive device! This equipment is sensitive to static electricity. Care must therefore be taken during all handling operations and inspections of this product in order to ensure product integrity at all times.

CE mark

The CE marking on this product indicates that it is in compliance with the applicable European Union Directives: Low Voltage, EMC, Radio Equipment and RoHS requirements.

Quality and environmental management

Kontron aims to deliver reliable high-end products designed and built for quality, and aims to comply with environmental laws, regulations, and other environmentally oriented requirements. For more information regarding Kontron's quality and environmental responsibilities, visit

https://www.kontron.com/en/quality-management .

Disposal and recycling

Kontron's products are manufactured to satisfy environmental protection requirements where possible. Many of the components used are capable of being recycled. Final disposal of this product after its service life must be accomplished in accordance with applicable country, State and/or local laws or regulations.

Waste electrical and electronic equipment directive

This product contains electrical or electronic materials. If not disposed of properly, these materials may have potential adverse effects on the environment and human health. The presence of this logo on the product means it should not be disposed of as unsorted waste and must be collected separately. Dispose of this product according to the appropriate local rules, regulations and laws.

WEEE directive logo

General power safety warnings and cautions

As a precaution and in case of danger, the power connector must be easily accessible. The power connector is the product's main disconnect device.

⚠CAUTION	All operations on this product must be carried out be sufficiently skilled personnel only.

	Electric Shock! Before installing a non-hot-swappable Kontron product into a system always ensure that your main power is switched off. This also applies to the installation of piggybacks. Serious electrical shock hazards can exist during all installation, repair, and maintenance operations of this product. Therefore, always unplug the power cables and any other cables which provide external voltages before performing any work on this product. An earth ground connection to the vehicle's chassis or a central grounding point shall remain connected. The earth ground cable shall be the last cable to be disconnected or the first cable to be connected when performing installation or removal procedures on this product.

⚠CAUTION	Risk of explosion if battery is replaced by an incorrect type. Dispose of used batteries according to the instructions.

Circuit overloading

Do not overload the circuits when connecting this product to the supply circuit as this can adversely affect overcurrent protection and supply wiring. Check the supply equipment nameplate ratings for correct use.
Do not connect Vbat+ as a source on pins rated for lower voltage. This will damage the platform interface.
Do not connect Vbat+ as a source to USB, display port and audio pins. This will damage the platform interface.

Reliable earth-grounding

Always maintain reliable grounding of equipment.

Overview

Specifications

Table of contents

K ey hardware features

Feature	Description
System	• Intelligent high-performance platform • Compact and flexible • Extended lifecycle (5-7 years)
Chassis	• Ruggedized • IP67 enclosure
Front panel LEDs	• Power • Status (customer configurable) • L1 (customer configurable) • L2 (customer configurable) LEDs are configurable: infrared for night vision, visible or ON/OFF
Storage	• Up to two 2.5-inch SATA SSDs or one 2.5-inch NVMe SSD • Up to three M.2 NVMe SSDs • Standard drive sizes available: 256GB, 512GB, 1TB, 2TB (other sizes on demand)
Power	• 9-36 VDC , 250 W • Standby current is 10 mA
Processor	• COMe Type7: Xeon ® D-1539 (optional) • COMe Type7: Xeon ® D-1559 (default)
Memory	• SODIMM Memory DDR4 • Up to 64 GB
Network	• One integrated network switch with 1588/PTP ◦ Six 1 GbE ports ◦ One 2.5 GbE port ◦ One 10 GbE port • One 1 GbE port with a direct link to the CPU on COMe
Serial	• One USB 3.0 port (a second USB 3.0 port can be available as a custom option) • Three USB 2.0 ports • Up to six RS-232 ports • Up to four RS-485/422 ports
I/O	• Ignition key switch input • Reset input • Power button input • Temperature input • Nine digital inputs (GPI) • Eight digital outputs (GPO) • Seven digital inputs/outputs (GPIO) • Two balanced audio inputs • Two balanced audio outputs • Eight pass-through signals that can be connected to an M.2 or an mPCIe
Interfaces	• One display port (a second display port can be available as a custom option) • Mezzanine (optional) ◦ Four CAN buses ◦ AURIX ^{TM} TC387 MCU with f our CAN buses • LTE/5G (optional) • Wi-Fi/Bluetooth (optional) • Other interfaces provided by M.2 or mPCIe cards can be available on demand (custom application) ◦ M.2 automotive Ethernet dual channel 1000Base-T1 ◦ mPCIe CAN bus FD dual channel
Expansion	• Two MXM GPUs (Type A or Type B) • Two M.2 2230/2260/2280/22110 M key • One M.2 2242/ 3052 B key • Two mPCIe full length
Battery	• Up to two RTC batteries (one installed at the factory) A capacitor keeps the time for up to 8 hours when replacing the battery.
SIM cards	• Up to 4 SIM cards (2 SIM cards with the M.2 B key and 1 SIM card per mPCIe port/slot)
Special features	• One GNSS input SMA • One PPS input or output SMA • Four optional SMA inputs for LTE/5G and Wi-Fi/Bluetooth modules • Nine axis IMU: accelerometer, magnetometer and gyroscope

K ey software features

Feature	Description
Platform management	System UEFI/BIOS
Operating system	Linux Ubuntu 18.04 or 20.04 LTS, 64-Bit

P hysical dimensions

Chassis	Measurements (mm [in])	Notes
Depth	284 [11.2]	Body
Width	330 [13]	Body
Height	115 [4.5]	Body

Packaging physical dimensions

Depth (mm [in])	Width (mm [in])	Height (mm [in])
440 [17.32]	370 [14.57]	250 [9.84]

Automotive Ethernet card weight

Weight (g [oz])
6.4 [0.225]

Shipping weights

Component	Weight (kg)	Weight (lb)
System weight – base configuration	8.8	19.4
System weight – base configuration + Packaging	10.2	22.05

E nvironmental specifications

Environment	Specification
Temperature, operating	-40°C to +71°C (-40°F to +160°F)*
Temperature, non-operating	-40°C to +85°C (-40°F to +185°F)
Temperature, storage, non-operating	-40°C to +85°C (-40°F to +185°F)

* With specific configurations. Refer to Environmental considerations for details.

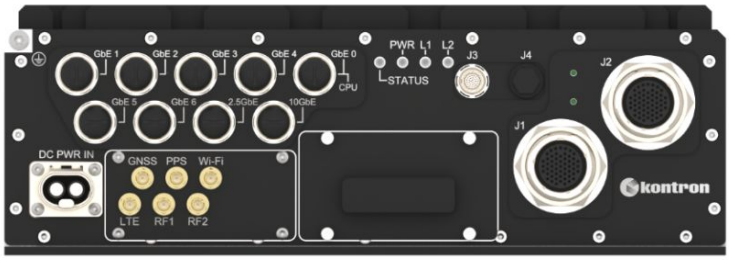
Platform components

Table of contents

Platform front panel

Overview of external connectors

For information on pinouts and electrical characteristics, refer to Connector pinouts for building custom cables .
For information on performing an ingress protection test, refer to Ingress protection test .
NOTE: Some connectors might not be present depending on product configuration.



NOTICE	Torque values are provided for mating connectors. These torques must not be exceeded as this will cause damage to the platform.

ID	Connector and function	Connector type	Supplier and P/N
GbE 0	1 Gigabit Ethernet to the COMe	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
GbE 1	1 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
GbE 2	1 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
GbE 3	1 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
GbE 4	1 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel

			shell: Harting 21033012000
GbE 5	1 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
GbE 6	1 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
2.5GbE	2.5 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
10GbE	10 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
DC PWR IN	DC power female connector	2-pole X-Coded Powerlok **Mating cycles:** 100	PCB receptacle: Amphenol PL082X-61-4
GPS	GNSS antenna	SMA female jack **Torque:** 4 lbs-in	PCB receptacle: AmphenolRF 132422-10
PPS	PPS signal (IN or OUT based on configuration)	SMA female jack **Torque:** 4 lbs-in	PCB receptacle: AmphenolRF 132422-10
Wi-Fi	Wi-Fi antenna	RP-SMA female jack **Torque:** 4 lbs-in	PCB receptacle: AmphenolRF 132422RP-10
LTE	LTE antenna	SMA female jack **Torque:** 4 lbs-in	PCB receptacle: AmphenolRF 132422-10
RF1	Project specific RF signal (typical usage second LTE antenna)	SMA female jack **Torque:** 4 lbs-in	PCB receptacle: AmphenolRF 132422-10
RF2	Project specific RF signal (typical usage second Wi-Fi antenna)	RP-SMA female jack **Torque:** 4 lbs-in	PCB receptacle: AmphenolRF 132422RP-10
J3	Display port 1 One USB 3.0	ODU AMC **Mating:** Push to connect, pull to disconnect **Mating cycles:** 5000	PCB receptacle: ODU GK1WAM-P27UB10-000L

J4	Display port 2 One USB 3.0	ODU AMC **Mating:** Push to connect, pull to disconnect **Mating cycles:** 5000	PCB receptacle: ODU GK1WAM-P27UB10-000L
J1	Four RS-232 (TX/RX) Two RS-232 (RX/TX/CTS/RTS) or RS-485/422 Four CAN FD (option) Six GPI and five GPO One USB 2.0 Power Reset Ignition key switch	MIL-DTL-38999 17-35, 55-pin straight female, Key N **Mating cycles:** 500	PCB receptacle: Amphenol AL07F17-35DN(P10)
J2	Two RS-485/422 Three GPI, three GPO and seven GPIO Two USB 2.0 Temperature sensor Balanced audio: • Two In • Two Out Eight pass-through I/O (option)	MIL-DTL-38999 17-35, 55-pin straight female, Key A **Mating cycles:** 500	PCB receptacle: Amphenol AL07F17-35DA(P10)

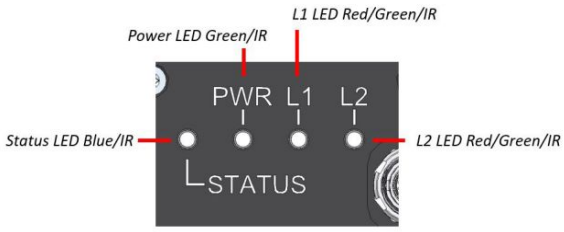
Overview of internal connectors

To access the internal connectors, remove the front tray. Refer to Components installation and assembly .

Connector description
2.5-inch SSD drive slot #1 (upper slot), only compatible with SATA
2.5-inch SSD drive slot #2 (lower slot), compatible with SATA and NVMe

Platform LEDs

To program the LEDs, refer to Controlling the LEDs .



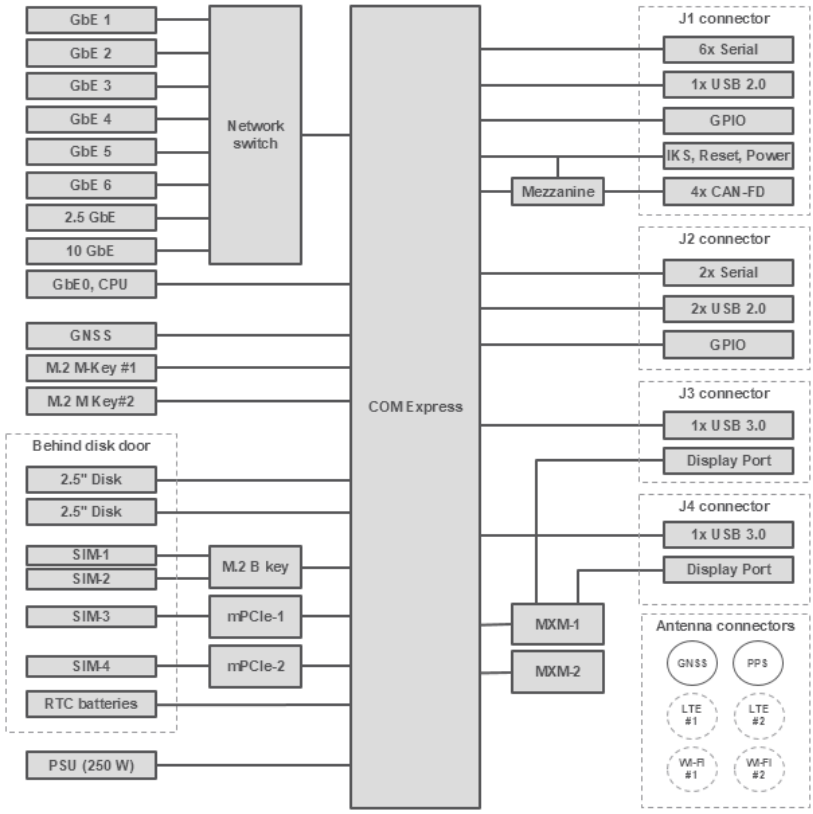
LED	Description and behavior
Status	Customer configurable LED from the operating system.
Power	Power status LED. Solid green when the power is on.
L1	Customer configurable LED from the operating system.
L2	Customer configurable LED from the operating system.

Product architecture

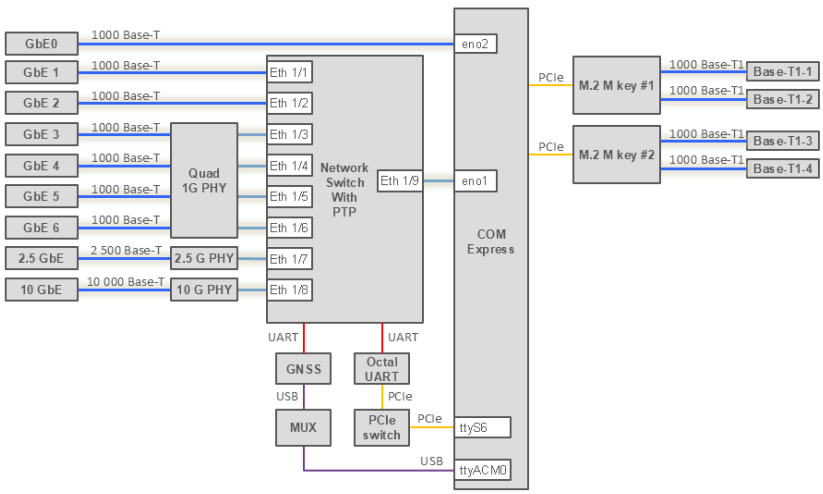
Table of contents

Block diagrams

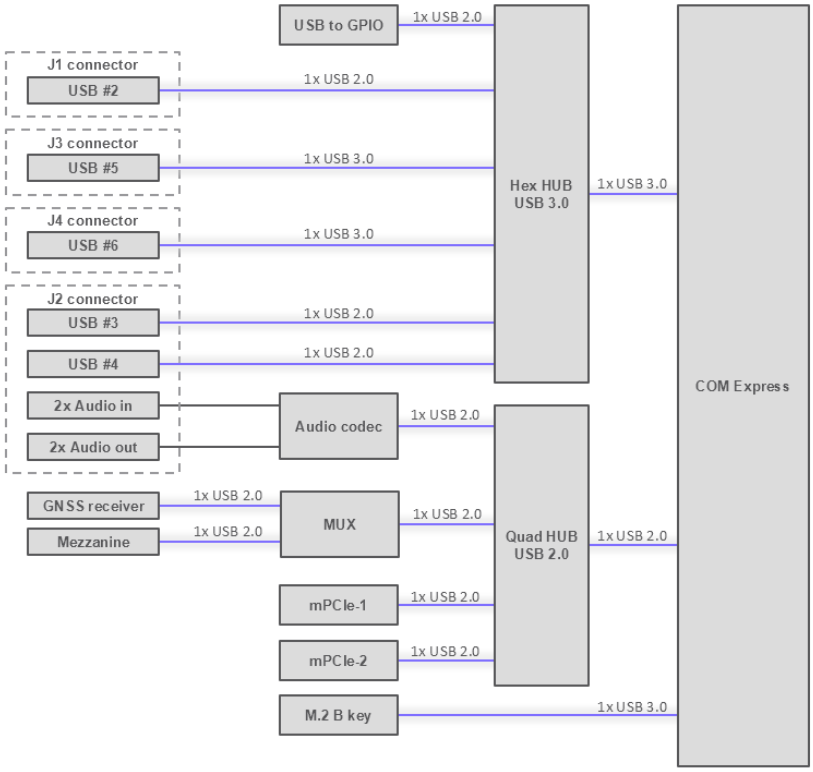
Overview



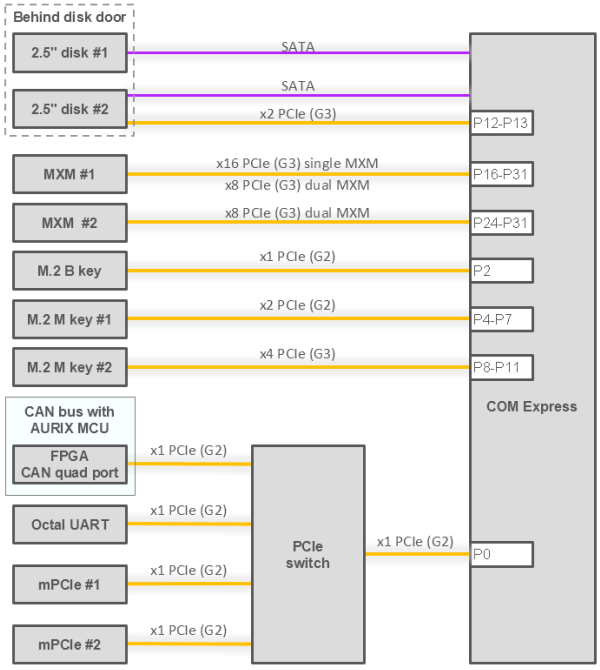
Network devices



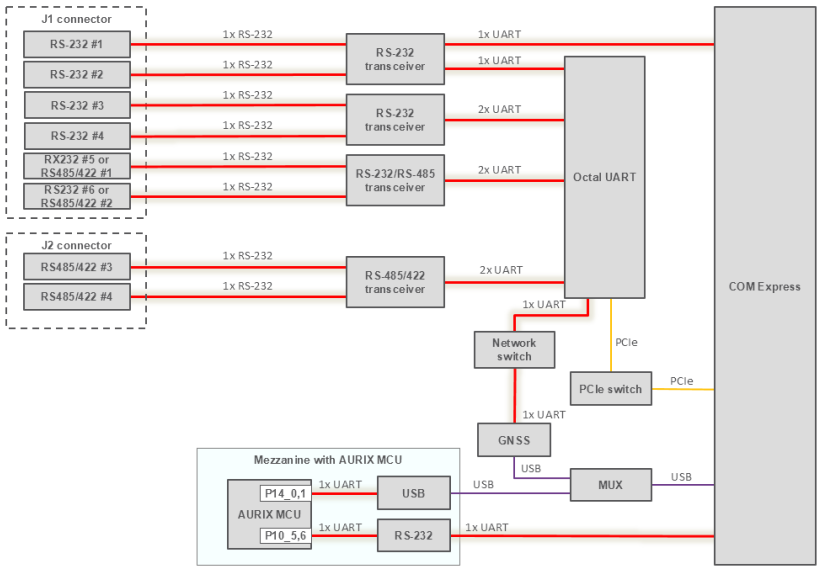
USB devices



PCIe and storage devices

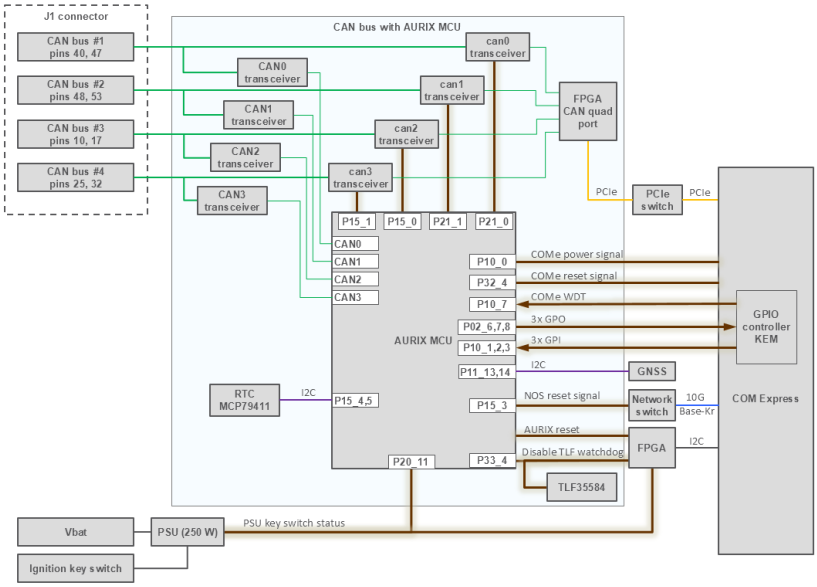


Serial connections

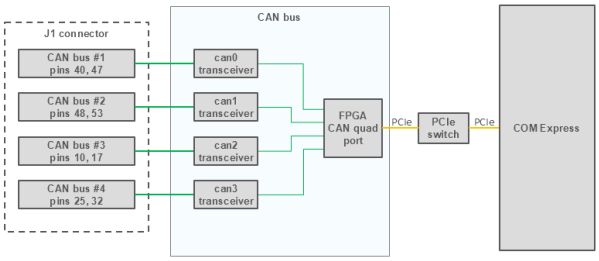


CAN bus

CAN bus and GPIOs with the AURIX MCU

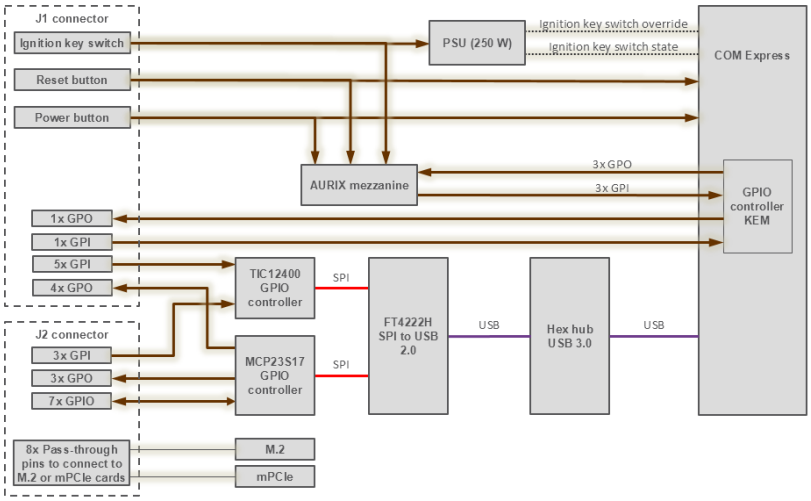


CAN bus without the AURIX MCU



GPIOs and others

This section describes the main signals available from front panel connectors J1 and J2.


J1 connector
Ignition key switch
Reset button
Power button
1x GPO
1x GPI
5x GPI
4x GPO
J2 connector
3x GPI
3x GPO
7x GPIO
8x Pass-through pins to connect to M.2 or mPCIe cards
PSU (250 W)
Ignition key switch override
Ignition key switch state
COM Express
AURIX mezzanine
3x GPO
3x GPI
GPIO controller KEM
TIC12400 GPIO controller
SPI
MCP23S17 GPIO controller
SPI
FT4222H SPI to USB 2.0
USB
Hex hub USB 3.0
USB
M.2
mPCIe

Description of system access methods

Table of contents

To configure, monitor and troubleshoot the S1901 platform and its components, several interfaces can be used:

- Operating system
- UEFI/BIOS
- Switch network operating system (NOS)
- AURIX MCU

Paths to the operating system

To access the operating system through one of the paths, refer to Accessing the operating system of a server .

Paths to the operating system	
Path description	Main reasons for use
Serial console (physical connection) *This is the recommended path for first time out-of-the-box system configuration.* *Path to access the server that is always available*	• Initial OS installation • OS network interface configuration • Unable to establish an SSH session to the OS • Troubleshooting
SSH *Ideal path once OS installation and OS network interface configurations have been performed.*	• Operating the platform under normal operation • Remote access to the OS
Remote Desktop *Ideal path once OS installation and OS network interface configurations have been performed.*	• Operating the platform under normal operation • Remote access to the OS

Paths to the UEFI/BIOS

The UEFI/BIOS is only accessible using a serial console and therefore requires a physical connection. To access the UEFI/BIOS, refer to Accessing the UEFI BIOS .

Paths to the switch network operating system (NOS)

To access the switch network operating system through one of the paths, refer to Accessing the switch network operating system .

Paths to the switch network operating system (NOS)	
Path description	Main reasons for use
Serial console from the integrated server *This is the recommended path for first time out-of-the-box NOS configuration.*	• Initial NOS configuration • Unable to establish an SSH session to the NOS • Troubleshooting the network switch
Switch Web UI *Ideal path to try and discover options or for one time manipulation, once OS installation and OS network interface configurations have been performed.*	• Switch NOS control and monitoring • Firmware upgrades
SSH from a remote computer *Ideal path once OS installation and OS network interface configurations have been performed.*	• Operating the platform under normal operation • Remote access to the OS • Automation, scripting

Paths to the AURIX MCU

To access the AURIX MCU through one of the paths, refer to Accessing the AURIX MCU .

Paths to the AURIX MCU	
Path description	Main reasons for use
Serial console from the integrated server *This is the recommended path for first time out-of-the-box use of the AURIX MCU demo code.*	• U sing the AURIX MCU demo code
USB port *Path for programming the AURIX MCU.*	• Programming the AURIX MCU • Creating an alternate serial port for the AURIX MCU shell output (this can be achieved by changing the demo code)

Recommended technical expertise

Expertise related to operating systems and network operating systems is required.
IP addresses will need to be assigned based on known MAC addresses, so appropriate IT expertise is required.

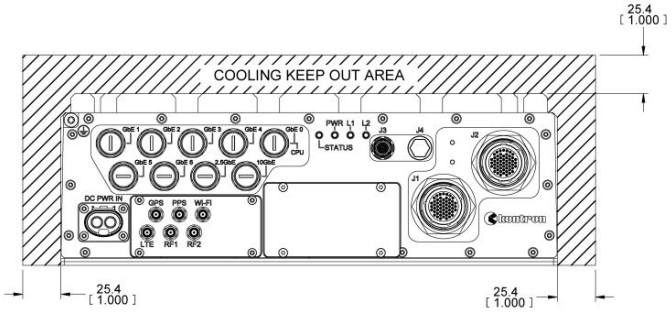
Planning

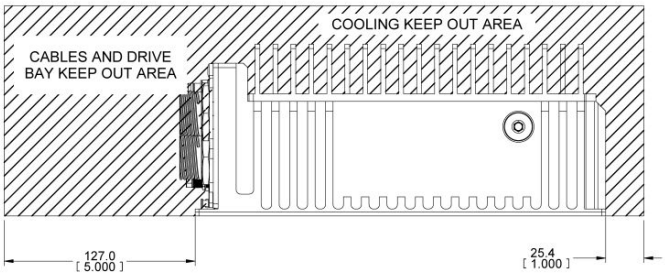
Environmental considerations

Table of contents

System cooling

Clearances





Airflow

The platform uses passive cooling and contains no fans.

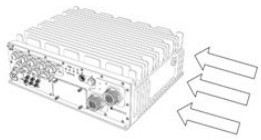
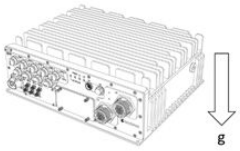
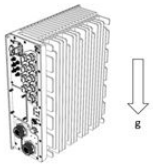
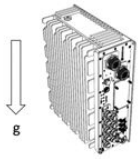
Thermal dissipation

Relevant section:
Connector pinouts for building custom cables (for the pinout relevant to the thermistor)

The platform includes a functionality allowing the installation of an external thermistor .
The system is designed to allow optimal heat transfer from major components through the upper shell body of the enclosure to the outside environment. Other hot components, such as the power supply module and other subcomponent boards, are also conducting their heat to the shell of the body.
The system enclosure adequately provides cooling up to 100 W at sea level with an approximate 20°C temperature rise at the component level when tested according to MIL-STD-810H, Method 501.7 Section 4.1b. For optimal operation, it is recommended that system not be mounted on a thermally insulating surface. Mounting over free air or on a thermally conductive surface at 71°C or less is recommended.
End application and optional components (such as M.2 or mPCIe) can also affect the maximum ambient temperature in which the system can operate. It is recommended that the customer test the system in their deployment environment using their application to assess the system operating range.

Typical orientation

The system is designed for optimal thermal performance when mounted on a flat thermally conductive horizontal surface with at least 1 inch of space between the system and any adjacent surfaces. Other orientations are possible, but the thermal performance of the unit may degrade. If the deployed environment for the system is not in the recommended orientation and spacing, and the environmental temperatures are expected to approach the upper limit of 71°C, please contact Kontron for technical assistance.
The following table provides some general installation design considerations and the values are based on simulations.

Factor	Description	Impact on temperature rise (%)		Comment
Base		0 - Baseline, sea level		CPU = D-1539 (35 W), GPU = T1000 Based on MIL-STD-810H, Method 501.7 Section 4.1b Airflow = 335 LFM from MXM side
		CPU	MXM	
Airflow	150 LFM	+40%	+40%	Airflow from MXM side
	250 LFM	+11%	+11%	Airflow from MXM side
	335 LFM	- 9%	+10%	Airflow from COMe side
	450 LFM	- 8.5%	- 8.5%	Airflow from MXM side
Natural convection (still air)		+125%	+158%	Same ambient as baseline Horizontal
		+130%	+156%	Same ambient as baseline Vertical, COMe side
		+124%	+164%	Same ambient as baseline Vertical, MXM side
		+160%	+199%	Same ambient as baseline Upside down
Altitude	10,000 FT	+18%	+18%	Same environmental conditions as baseline
	20,000 FT	+34%	+33%	
	40,000 FT	+110%	+120%	
Conductive mounting plate		- 3%	- 2.5%	Same environmental conditions as baseline 18.75 x 17.25, 1/4 thick aluminum plate
Component and usage	CPU 75% / GPU 75% load	- 19%	- 19%	Same environmental conditions as baseline
	D-1559 (45 W) / RTX3000 100% load	+32%	+7%	Same environmental conditions as baseline

Power consumption and power budget

Table of contents

Relevant section:
Platform power management

DC power supply input voltage and current requirements

DC input voltage	
Nominal	12 or 24 VDC
Minimum	9 VDC
Maximum	36 VDC
DC input current	
Maximum	31 A at 9 V
Power input	
Nominal	250 W
Maximum	288 W

Power consumption examples

This section provides power consumption values obtained in a test environment. Actual values highly depend on the application that will be used. The values provided must therefore only be used as a general reference and tests need to be performed with the actual hardware configuration and application that will be used.

System power consumption

The following S1901 configuration was used to obtain the typical power consumption values shown in the table below:

- Xeon® D-1539 processor
- Two 4 GB SODIMM
- One 256 GB M.2 NVME module
- 250 W isolated PSU power at 12 V

Status	Typical consumption (W)	Notes
Idle	50	Idle power consumption was measured in Ubuntu 20.04 once it had finished booting
Maximum application	110	Maximum power was measured in Ubuntu 20.04 running "mprime -t" as a stress application

NOTES:

- DC power supply input is at 12 VDC.
- Test was performed at ambient temperature.
- Power consumption varied during the test.
- Power consumption was measured at the DC power supply input.

Component power consumption examples

Power figures given per component in the table were measured at the DC power supply output (12 V side).

Components	Maximum power consumption (W)	Additional power required at the DC power supply input		Notes
		12 V (+12%)	24 V (+7%)	
Intel® Xeon® D-1539	35	39	37	TDP
Intel® Xeon® D-1559	45	50	48	TDP
Two 4 GB SODIMM	3.4	3.8	3.6	Under active use
Two 16 GB SODIMM	3.8	4.3	4.1	Under active use
Two 32 GB SODIMM	4.8	5.4	5.1	Under active use

CAN bus PEAK 4 port	4	4.5	4.3	Under active use
SATA 128 GB-2TB 2.5" SSD	3.5	3.9	3.7	Under active use. Idle power is 0.5 W.
NVMe 256 GB-2TB M.2 SSD	7	7.8	7.5	Under active use. Idle power is 1 W.
MXM T1000	50	70	67	Under active use, including peak power burst
MXM RTX3000	80	112	107	Under active use, including peak power burst
EM7565 M.2 LTE 4G	5	5.6	5.4	Maximum antenna power
WPEB-263ACNI(BT) Wi-Fi/Bluetooth	3	3.4	3.2	Under active use
CAN bus mezzanine with the AURIX TC387 safety MCU	13	14.5	13.9	Under active use (AURIX and CAN bus)
mPCIe CAN bus FD dual channel	1	1.1	1.1	Under active use
M.2 automotive Ethernet dual channel 100/1000Base-T1 Marvel 88Q211	1.15	1.3	1.2	Under active use

NOTICE	If all the optional components are used and operate at maximum power, the system co uld exceed its maximum power consumption.

MAC addresses

Table of contents

MAC addresses

Relevant sections:
Network device architecture
Default user names and passwords

Interface description	MAC address	Notes
COMe	MAC_GbE0	Direct to CPU; GbE 0; MAC address provided on product label sticker; called eno2 in Ubuntu
Internal 10G-kr	MAC_10G-kr	Internal MAC address going to the network switch; called eno1 in Ubuntu
Network switch	MAC_BASE	To access the NOS IP interface (VSC7440)
NOS reserved	MAC_BASE +1 to MAC_BASE +9	

Discovering the COMe MAC address

Three methods can be used to discover the COMe MAC address:

- The label
- A serial console
- The UEFI/BIOS

Discovering the COMe MAC address using the label

Step_1	Look for the MAC address on the product label located on the right side of the unit. MAC: COMe address (labeled GbE 0 on the front panel and called eno2 in Ubuntu)	

Discovering the COMe MAC address using a serial console

Prerequisites

1	An OS is installed.
2	A serial connection to the device is required.
3	(Optional) A null modem is connected.
4	A serial console tool is installed on the remote computer. • Speed (Baud): 115200 • Data bits: 8 • Stop bits: 1 • Parity: None • Flow Control: None • Recommended emulation mode: VT100+ **NOTE:** PuTTY is recommended.

Relevant sections:
Connector pinouts for building custom cables
Kontron test cables

Procedure

NOTE: If using the Kontron test harness, RS-232 port #1 is the P3 connector of the J1 harness.

Step_1	From a remote computer with a serial connection to RS-232 #1 (the default serial console to COMe CPU), open a serial console tool, and start the communication between the console and the serial port to which the device is connected.	
Step_2	The OS start screen will be displayed. **NOTE:** If the OS is not displayed, press **Enter** .	
Step_3	Open a command line interface and use the following command to discover the OS IP address. LocalServer_OSPrompt:~# **ip a** In the image, the OS IP address of COMe eno2 is 172.16.220.35 and the MAC address is 00:e0:4b :70:09:a8.	

Discovering the COMe MAC address using the UEFI BIOS

Prerequisites

1	An OS is installed.
2	A serial connection to the device is required.
3	(Optional) A null modem is connected.
4	A serial console tool is installed on the remote computer. • Speed (Baud): 115200 • Data bits: 8 • Stop bits: 1 • Parity: None • Flow Control: None • Recommended emulation mode: VT100+ **NOTE:** PuTTY is recommended.

Relevant sections:
Connector pinouts for building custom cables
Kontron test cables
Accessing the UEFI BIOS

Procedure

NOTE: If using the Kontron test harness, RS-232 port #1 is the P3 connector of the J1 harness.

Step_1	Access the UEFI BIOS using a serial console.	
Step_2	From the **Main** menu, access **Platform Information** . **NOTE:** The MAC address required is that of eno2, which is called I210 in the UEFI/BIOS.	[redacted]

Discovering the switch NOS MAC address

Prerequisites

1	An OS is installed.
2	A serial connection to the device is required.
3	(Optional) A null modem is connected.
4	A serial console tool is installed on the remote computer. • Speed (Baud): 115200 • Data bits: 8 • Stop bits: 1 • Parity: None • Flow Control: None • Recommended emulation mode: VT100+ **NOTE:** PuTTY is recommended.

Relevant sections:
Linux devices
Connector pinouts for building custom cables
Kontron test cables

Procedure

NOTE: If using the Kontron test harness, RS-232 port #1 is the P3 connector of the J1 harness.

Step_1	From a remote computer with a serial connection to RS-232 port #1 (the default serial console), open a serial console tool, and start the communication between the console and the serial port to which the device is connected.	
Step_2	The OS start screen will be displayed. **NOTE:** If the OS is not displayed, press **Enter** .	
Step_3	Open the OS CLI.	[redacted]
Step_4	Access the network switch. LocalServer_OSPrompt:~# **sudo minicom -D /dev/ttyS6**	[redacted]
Step_5	Log in using the appropriate credentials. The prompt displayed is the MAC_BASE (the MAC of the network switch). In the image, the MAC address is 00:A0:A5:E0:1C:F4.	[redacted]

Linux devices

Table of contents

Relevant section:
Connector pinouts for building custom cables

To communicate with various platform components, the Linux port is required.

Serial ports

NOTE: The serial port tables below apply to COMe bBD7 Broadwell modules. The Linux device paths will be different for other module families.

Serial ports without an mPCIe CAN bus FD dual channel

Serial port type	Port	Linux device path	Location
RS-232	RS-232 #1	/dev/ttyS0	J1
	RS-232 #2	/dev/ttyS7	J1
	RS-232 #3	/dev/ttyS8	J1
	RS-232 #4	/dev/ttyS9	J1
RS-232 or RS-485/422	RS-232 #5 or RS-485/422 #1	/dev/ttyS4	J1
	RS-232 #6 or RS-485/422 #2	/dev/ttyS5	J1
RS-485/422	RS-485/422 #3	/dev/ttyS10	J2
	RS-485/422 #4	/dev/ttyS11	J2
CAN bus	CAN bus #1	/dev/can0	J1
	CAN bus #2	/dev/can1	J1
	CAN bus #3	/dev/can2	J1
	CAN bus #4	/dev/can3	J1
AURIX MCU	IfxAsclin2_TX_P10_5 IfxAsclin2_RXD_P10_6	/dev/ttyS1	Internal
UART	Network switch	/dev/ttyS6	Internal

Serial ports with an mPCIe CAN bus FD dual channel

Serial port type	Port	Linux device path	Location
RS-232	RS-232 #1	/dev/ttyS0	J1
	RS-232 #2	/dev/ttyS7	J1
	RS-232 #3	/dev/ttyS8	J1
	RS-232 #4	/dev/ttyS9	J1
RS-232 or RS-485/422	RS-232 #5 or RS-485/422 #1	/dev/ttyS4	J1
	RS-232 #6 or RS-485/422 #2	/dev/ttyS5	J1
RS-485/422	RS-485/422 #3	/dev/ttyS10	J2
	RS-485/422 #4	/dev/ttyS11	J2
CAN bus	CAN bus #1	/dev/can2	J1
	CAN bus #2	/dev/can3	J1
	CAN bus #3	/dev/can4	J1
	CAN bus #4	/dev/can5	J1
	CAN bus #5	/dev/can0	J2
	CAN bus #6	/dev/can1	J2
AURIX MCU	IfxAsclin2_TX_P10_5 IfxAsclin2_RXD_P10_6	/dev/ttyS1	Internal
UART	Network switch	/dev/ttyS6	Internal

USB ports

The Linux device path depends on the actual configuration.

USB port list

USB port type	Port	Location
USB 3.0	USB #1	Removed from the product
	USB #5	J3
	USB #6	J4
USB 2.0	USB #2	J1
	USB #3	J2
	USB #4	J2
	/dev/ttyUSB **x** OR /dev/ttyACM0 **NOTE:** In ttyUSB **x** , **x** will depend on detection order. Use the commands below to get the variable value.	Internal to AURIX MCU OR Internal to GNSS

Typical Linux commands to get information on USB devices

Result description	Command
List of the paths of all the USB devices connected	LocalServer_OSPrompt:~# **ls -als /sys/bus/usb/devices/**
List of all USB devices connected	LocalServer_OSPrompt:~# **lsusb**
List of all USB devices connected with detailed information on each device	LocalServer_OSPrompt:~# **usb-devices**
Status of all USB devices connected	LocalServer_OSPrompt:~# **dmesg | grep -i usb**
List of all USB devices connected with Vendor and ProdID information	LocalServer_OSPrompt:~# **cat /sys/kernel/debug/usb/devices | more**

Automotive Ethernet ports

NOTE: In enp **XX** s0, **XX** will depend on detection order. Use the command below to get the variable value.

Automotive Ethernet port	Port	Linux device path	Location
1000Base-T1	M.2 #1, port #1	enp **XX** s0	J2
	M.2 #1, port #2	enp **XX** s0	J2

Command to obtain Linux device path of automotive Ethernet ports

Note that the Linux device path ID (**XX**) is based on the MAC address and that the port with the highest MAC address will have the lowest path ID:

- enp**17** s0 = 00:a0:a5:e4:54:**b5** = M.2 #1, **port #2**
- enp**18** s0 = 00:a0:a5:e4:54:**b4** = M.2 #1, **port #1**

Step_1	LocalServer_OSPrompt:~$ **ip addr | grep enp -A 1**	

Balanced audio ports

Audio type	Port	Linux device path	Location
Input	Capture	/dev/snd/pcmC1D0c	J2
Output	Control	/dev/snd/controlC1	Internal
	Playback (audio out)	/dev/snd/pcmC1D0p	J2

GNSS port

Relevant section:
Configuring the GNSS

Port	Linux device path	Location
GNSS U-Blox NEO-M9N	/dev/ttyACM0	Internal; Muxed with AURIX MCU

Material, information and software required

Table of contents

Material and information required

Component installation and assembly

Relevant section:
Components installation and assembly

Item_1	T10 Torx screwdriver

Configuration material

Item_1	One laptop with a serial port with a null modem connection

Power cables and tooling

NOTICE	All mating connectors and cables used with the S1901 platform must have an IP67 rating. When custom cables are built, the mating connector manufacturer's instructions related to the IP67 rating must be followed. When S1901 connectors are not used, the protection caps of the connectors must be installed, as this ensures the platform complies with the IP67 rating. Note that the power input is covered with a dust cap that is not rated IP67 . If no mating connector is connected to this connector, the platform does not comply with the IP67 rating. An ingress protection test can be performed to confirm all connectors are correctly mated and the IP67 rating is achieved.

Relevant sections:
Platform components
Connector pinouts for building custom cables
Kontron test cables
Cabling

Item_1	One DC power connector
Item_2	Up to 10 AWG black stranded wire to build the power cable based on the length required **NOTE:** Use a wire gauge appropriate based on current and local wiring codes.
Item_3	Up to 10 AWG red stranded wire to build the power cable based on the length required **NOTE:** Use a wire gauge appropriate based on current and local wiring codes.
Item_4	(Optional) One Kontron test cable kit or one Kontron DC power input cable

Network cables and modules

NOTICE	All mating connectors and cables used with the S1901 platform must have an IP67 rating. When custom cables are built, the mating connector manufacturer's instructions related to the IP67 rating must be followed. When S1901 connectors are not used, the protection caps of the connectors must be installed, as this ensures the platform complies with the IP67 rating. Note that the power input is covered with a dust cap that is not rated IP67 . If no mating connector is connected to this connector, the platform does not comply with the IP67 rating. An ingress protection test can be performed to confirm all connectors are correctly mated and the IP67 rating is achieved.

Relevant sections:
Platform components
Connector pinouts for building custom cables
Kontron test cables
Cabling

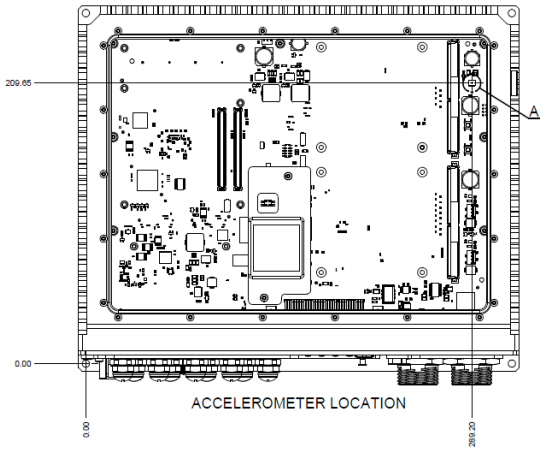
NOTE: If a part number is specified, all items in the part list may be replaced by equivalents only after approval by Kontron.

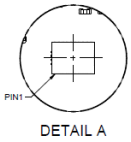
Item_1	Flat screwdriver
Item_2	(Optional) Kontron test cable kit
Item_3	Custom cables built by customers based on their application

Accelerometer location

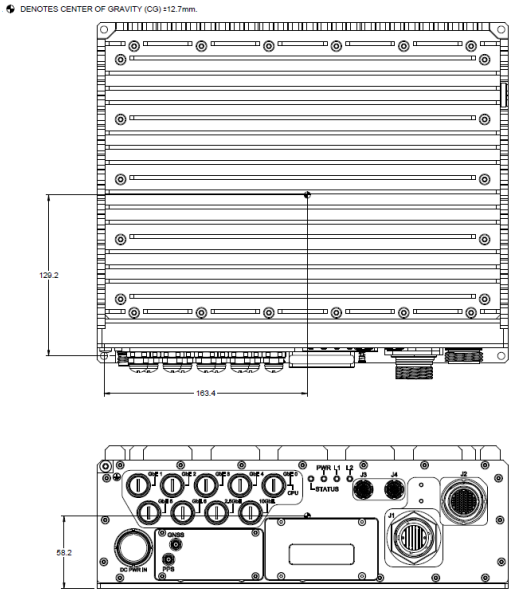
Relevant section:
Installing the board support package

The platform includes a nine axis IMU (accelerometer, magnetometer and gyroscope). The component related to this device is the BNO055.
NOTE: The dimensions on the drawing are in millimeters and refer to the center of the BNO055 chip.





Center of gravity location



Software required

Relevant section:
Installing the AURIX MCU development environment and demo code

Item_1	A terminal emulator such as **PuTTY** is installed on a remote computer.
Item_2	A Remote Desktop client is installed on a remote computer.
Item_3	Software to program the AURIX MCU (MemTool from Infineon running on Windows inside a virtual machine is the preferred method)

NOTE: Item_3 is required only when the CAN bus mezzanine with the AURIX safety MCU is installed.

Platform, modules and accessories

This section provides the complete list of compatible parts and components that can be ordered from Kontron.

Description	Kontron P/N
RSSD 1TB SATA	1068-1217
RSSD 2TB SATA	1068-1125
Test cable kit, 2 m	1069-7746
DC power input cable, 2 m	1067-8665
Network cable, 2 m	1064-9369
I/O harness for J1, 2 m	1068-5060
I/O harness for J2, 2 m	1068-5062
Display port and USB 3.0 cable for J3 or J4 , 240 mm	1068-5064

Connector pinouts for building custom cables

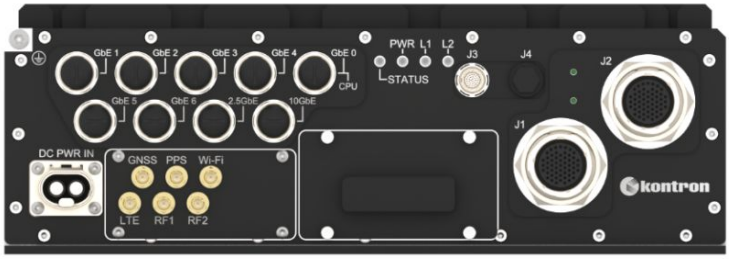
Table of contents

Customers can build custom cables based on the information provided in this section.

NOTICE	All mating connectors and cables used with the S1901 platform must have an IP67 rating. When custom cables are built, the mating connector manufacturer's instructions related to the IP67 rating must be followed. When S1901 connectors are not used, the protection caps of the connectors must be installed, as this ensures the platform complies with the IP67 rating. Note that the power input is covered with a dust cap that is not rated IP67 . If no mating connector is connected to this connector, the platform does not comply with the IP67 rating. An ingress protection test can be performed to confirm all connectors are correctly mated and the IP67 rating is achieved.

NOTICE	Mating connectors are provided as a reference. Users are responsible for ensuring the mating connectors selected are appropriate for the use environment and comply with their environmental requirements.

Platform external connectors



Description and pinout of external connectors

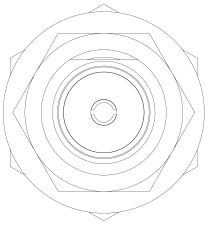
This section describes the following connectors and lists their pinouts:

- SMA GNSS RF input
- SMA PPS
- 55-pin I/O connector – J1
- 55-pin I/O connector – J2
- Ethernet port connectors
- Display port and USB 3.0 connectors – J3 and J4
- DC power supply input connector

Relevant sections:

Platform components
Grounding
Controlling GPIOs

SMA GNSS RF input

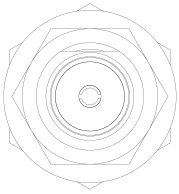


Mating connector: SMA Male

Description:

- Integrated NEO-M9N GNSS receiver antenna input
- Can be used with passive and active antennas (the antenna must be matched to the requisite 50 ohms)
- Suitable for connection to external outdoor antennas
- RF input
  - Maximum input power is < 0 dBm
  - Good antenna with > 4 dBic gain recommended
  - Good low noise amplifier (LNA) with a noise figure of less than 2 dB recommended
  - Active antenna gain of 15 dB to 35 dB (maximum) recommended
- DC bias output
  - 5 V ± 5%
  - Up to 150 mA
  - Over-current protected (< 350 mA)
  - Thermally protected
- Includes surge protection (IEC 61000-4-5 class 2, 1 kV)

SMA PPS



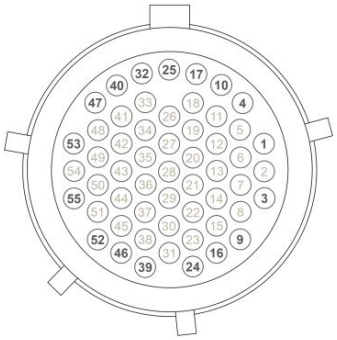
Mating connector: SMA Male

Description:

- Compliant with ITU-G.703, section 19.2
- Output is 5 V source terminated (50 ohms)
- Output duty cycle is 10% (100 ms)
- Suitable for use with unterminated loads:
  - V_{OH} > 2.6 V at I_{OH} = -12 mA
  - V_{OL} < 0.7 V at I_{OH} = 12 mA
- Suitable for use with 50 ohms to ground terminated loads:
  - V_{OH} > 1.2 V
  - V_{OL} < 0.3 V
- PPS rising edge (at SMA) aligned within ± 5 ns from internal time of day (ToD) counter

55-pin I/O connector – J1

Description: 55-pin Straight Plug 17-35 I/O female connector, Key N
Mating connector: Amphenol D38999/26FE35PN
External connector pinout:



This connector includes the following functionalities:

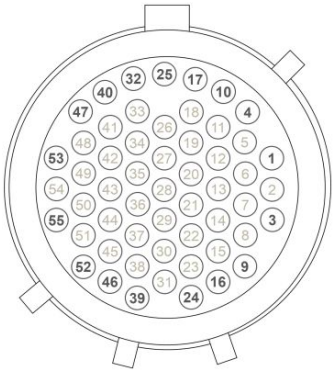
- RS-232 #1 (default serial console), #2, #3 and #4
- RS-232 #5 or RS-485/422 #1 (default is full-duplex RS-485)
- RS-232 #6 or RS-485/422 #2 (default is full-duplex RS-485)
- CAN bus #1, #2, #3 and #4
- General Purpose Input #1 – direct to COMe
  - Electrical characteristics: Logic level is 5 V CMOS (ISO7742 isolator Vcc 5 V input).
- General Purpose Output #1 – direct to COMe
  - Electrical characteristics: Voh 4.6 V maximum at 4 mA (ISO7742 isolator Vcc 5 V output).
- General Purpose Input #2, #3, #4, #5 and #6
  - Electrical characteristics: 0 to 36 V (input only). TIC12400 functionalities available, such as programmable thresholds and analog input reading.
- General Purpose Output #2, #3, #4 and #5
  - Electrical characteristics: 0 to 40 V (open collector output only). A 22 AWG wire required and the GPIO isolated GND will take all return current. The current must not exceed the rating of the GPIO isolated GND. The chip used for GPOs is a TPL7407LDR.
- USB 2.0 #2
- Power, Reset
- Ignition key switch (50mA max)

Pin	J1 signal description	Pin	J1 signal description	Pin	J1 signal description
1	USB #2 Vcc	20	GND	39	RS-232 #5 CTS or RS-485/422 #1 Rx- (B)
2	Input 5 Isolated	21	Input 6 Isolated	40	CAN bus #1 High
3	GPIO Isolated GND	22	Output 1 Isolated	41	GND
4	USB #2 D-	23	GND	42	RS-232 #1 Tx
5	USB #2 D+	24	RS-232 #5 RTS or RS-485/422 #1 Tx- (Z)	43	RS-232 #2 Tx
6	Power Button Input **NOTE:** Internal pull-up resistor. Lab only use. In production, use the ignition key switch.	25	CAN bus #4 High	44	RS-232 #2 Rx
7	Input 4 Isolated	26	RS-232 #4 Tx	45	GND
8	Input 3 Isolated	27	RS-232 #4 Rx	46	RS-232 #5 Rx or RS-485/422 #1 Rx+ (A)
9	Output 4 Isolated	28	GPIO Isolated GND	47	CAN bus #1 Low
10	CAN bus #3 High	29	Output 5 Isolated	48	CAN bus #2 High
11	GND	30	Output 2 Isolated	49	GND
12	RS-232 #3 Rx	31	RS-232 #5 Tx or RS-485/422 #1 Tx+ (Y)	50	GND
13	Reset Button Input **NOTE:** Internal pull-up resistor	32	CAN bus #4 Low	51	RS-232 #6 CTS or RS-485/422 #2 Rx- (B)
14	Output 3 Isolated	33	GND	52	RS-232 #6 Rx or RS-485/422 #2 Rx+ (A)
15	Input 2 Isolated	34	RS-232 #1 Rx	53	CAN bus #2 Low
16	Input 1 Isolated	35	GND	54	RS-232 #6 RTS or RS-485/422 #2 Tx- (Z)
17	CAN bus #3 Low	36	GND	55	RS-232 #6 Tx or RS-485/422 #2 Tx+ (Y)
18	GND	37	Ignition Key Switch Input **NOTE:** Internal pull-down resistor to Vbat-		
19	RS-232 #3 Tx	38	Vbat+ to drive Ignition Key Switch		

For information on how to use the Ignition key switch input, Reset button input and Power button input, refer to Platform power management.

55-pin I/O connector – J2

Description: 55-pin Straight Plug 17-35 I/O female connector, Key A
Mating connector: Amphenol D38999/26FE35PA
External connector pinout:



This connector includes the following functionalities:

- RS-485/422 #3 and #4 (default is full-duplex RS-485)
- General Purpose Input #7, #8, #9 (use with GPIO Isolated GND)
  - Electrical characteristics: 0 to 36 V (input only). TIC12400 functionalities available, such as programmable thresholds and analog input reading.
- General Purpose Output #6, #7, #8 (use with GPIO Isolated GND)
  - Electrical characteristics: 0 to 40 V (open collector output only). A 22 AWG wire required and the GPIO isolated GND will take all return current. The current must not exceed the rating of the GPIO isolated GND. The chip used for GPOs is a TPL7407LDR.
- General Purpose I/O #1, #2, #3, #4, #5, #6 and #7
  - Electrical characteristics: 0 to 5 V (Vil is 1 V, Voh is 4.3 V at 3 mA, Vol is 0.6 V at 8 mA (max 25 mA)). The chip used for GPIOs is a MCP23S17.
- M.2 or mPCIe pass-through I/O #1, #2, #3, #4, #5, #6, #7 and #8
- USB 2.0 #3 and #4
- Temperature sensor
  - NTC thermistor, 10 Kohm, 3976K Bead (recommended thermistor TE Connectivity GA10K3A1IA)
- Audio input left and right
- Audio output left and right

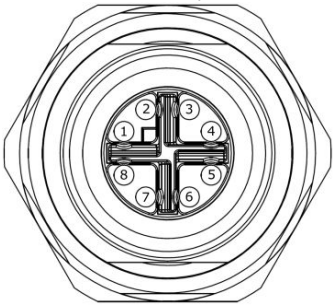
Pin	J2 signal description	Pin	J2 signal description	Pin	J2 signal description
1	Balanced Audio Left In-	20	GPIO 4 Isolated	39	RS-485/422 #4 Rx+ (A)
2	Balanced Audio Left In+	21	Input 7 Isolated	40	Balanced Audio Right Out-
3	RS-485/422 #3 Tx+ (Y)	22	M2 or mPCIe pass-through 5 **NOTE:** See table below for information specific to hardware configuration.	41	GPIO 3 Isolated
4	Balanced Audio Right In-	23	RS-485/422 #4 Isolated GND	42	GPIO 7 Isolated
5	Balanced Audio Right In+	24	RS-485/422 #4 Tx+ (Y)	43	GPIO Isolated GND
6	GND	25	USB 2.0 #3 Vcc	44	Input 8 Isolated
7	RS-485/422 #3 Isolated GND	26	GPIO 1 Isolated	45	GND
8	RS-485/422 #3 Tx- (Z)	27	M2 or mPCIe pass-through 3 **NOTE:** See table below for information specific to hardware configuration.	46	RS-485/422 #4 Rx- (B)
9	RS-485/422 #3 Rx+ (A)	28	M2 or mPCIe pass-through 4 **NOTE:** See table below for information specific to hardware configuration.	47	USB 2.0 #4 Vcc
10	Balanced Audio Left Out+	29	M2 or mPCIe pass-through 8 **NOTE:** See table below for information specific to hardware configuration.	48	GPIO Isolated GND
11	GND	30	M2 or mPCIe pass-through 7 **NOTE:** See table below for information specific to hardware configuration.	49	Output 7 Isolated
12	GPIO 5 Isolated	31	RS-485/422 #4 Tx- (Z)	50	GND
13	M2 or mPCIe pass-through 2 **NOTE:** See table below for information specific to hardware configuration.	32	Balanced Audio Right Out+	51	USB 2.0 #3 D+
14	M2 or mPCIe pass-through 1 **NOTE:** See table below for information specific to hardware configuration.	33	GND	52	USB 2.0 #3 D-
15	Output 6 Isolated	34	Input 9 Isolated	53	Output 8 Isolated
16	RS-485/422 #3 Rx- (B)	35	GPIO 6 Isolated	54	USB 2.0 #4 D+
17	Balanced Audio Left Out-	36	Thermistance	55	USB 2.0 #4 D-
18	GND	37	Thermistance		
19	GPIO 2 Isolated	38	M2 or mPCIe pass-through 6 **NOTE:** See table below for information specific to hardware configuration.		

M.2 or mPCIe pass-through specific hardware configuration

Name	Pin	mPCIe #1 dual CAN bus only	M.2 #1 automotive only	mPCIe #1 dual CAN bus and M.2 #1 automotive	M.2 #1 automotive and M.2 #2 automotive
M2 or mPCIe pass-through 1	14	/dev/can0: Low	-	/dev/can0: Low	M.2 #2, port 1: 1GBase-T1: Plus
M2 or mPCIe pass-through 2	13	/dev/can0: High	-	/dev/can0: High	M.2 #2, port 1: 1GBase-T1: Minus
M2 or mPCIe pass-through 3	27	-	M.2 #1, port 1: 1GBase-T1: Plus	M.2 #1, port 1: 1GBase-T1: Plus	M.2 #1, port 1: 1GBase-T1: Plus
M2 or mPCIe pass-through 4	28	-	M.2 #1, port 1: 1GBase-T1: Minus	M.2 #1, port 1: 1GBase-T1: Minus	M.2 #1, port 1: 1GBase-T1: Minus
M2 or mPCIe pass-through 5	22	/dev/can1: Low	-	/dev/can1: Low	M.2 #2, port 2: 1GBase-T1: Plus
M2 or mPCIe pass-through 6	38	/dev/can1: High	-	/dev/can1: High	M.2 #2, port 2: 1GBase-T1: Minus
M2 or mPCIe pass-through 7	30	-	M.2 #1, port 2: 1GBase-T1: Plus	M.2 #1, port 2: 1GBase-T1: Plus	M.2 #1, port 2: 1GBase-T1: Plus
M2 or mPCIe pass-through 8	29	-	M.2 #1, port 2: 1GBase-T1: Minus	M.2 #1, port 2: 1GBase-T1: Minus	M.2 #1, port 2: 1GBase-T1: Minus

Ethernet port connectors

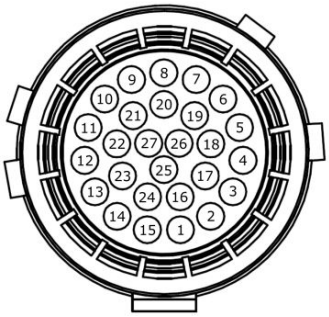
Description: M12 8-pin X-Coded female connector
Mating connector: Harting 21330100850 series or equivalent
External connector pinout:



Pin	Signal description
1	TxRx A +
2	TxRx A -
3	TxRx B +
4	TxRx B -
5	TxRx D +
6	TxRx D -
7	TxRx C -
8	TxRx C +

Display port and USB 3.0 connectors – J3 and J4

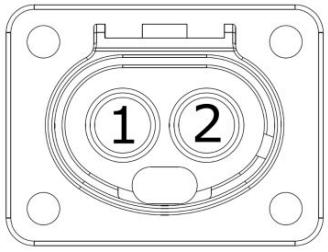
NOTE: Connector J4 is a custom option
Description: ODU AMC female display port connector
Mating connector: A11WAM-P27XBC0-0000
External connector pinout:



Pin	Signal description	Pin	Signal description	Pin	Signal description
1	ML_Lane 1(p)	10	StdA_SSRX–	19	USB GND_DRAIN
2	ML_Lane 0(p)	11	DP_PWR	20	AUX_CH(n)
3	GND Lane 1	12	ML_Lane 3(n)	21	DP_PWR Return
4	D+	13	ML_Lane 2(n)	22	GND Lane 3
5	VBUS	14	ML_Lane 2(p)	23	ML_Lane 3(p)
6	StdA_SSTX +	15	ML_Lane 1(n)	24	GND Lane 2
7	StdA_SSTX–	16	ML_Lane 0(n)	25	Hot Plug
8	GND AUX	17	GND Lane 0	26	USB GND
9	StdA_SSRX+	18	D-	27	AUX_CH(p)

DC power supply input connector

Description: 2-pole X-Coded Powerlok DC power female connector
Mating connector: Amphenol TPI PL182X-61-6P11 or equivalent
External connector pinout:



Pin	Signal description
1	Power input+ (Vbat+)
2	Power input- (Vbat-)

Kontron test cables

Table of contents

NOTICE	The Kontron test cables are not IP67 rated. They are designed for use in a lab or test environment.

Test cable kit

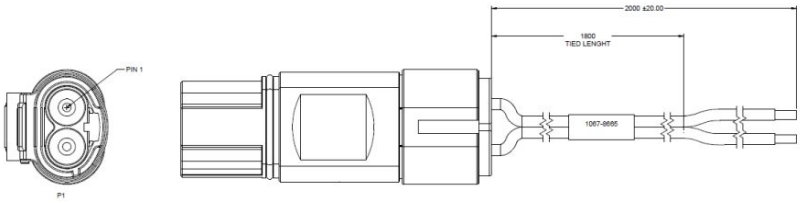
A test cable kit (P/N 1069-7746) is offered by Kontron to test the S1901 EvoTRAC product.
The test cable kit includes the following cables:

- One DC power input cable, 2 m (P/N 1067-8665)
- Nine network cables, 2 m (P/N 1064-9369)
- One I/O harness for J1, 2 m (P/N 1068-5060)
- One I/O harness for J2, 2 m (P/N 1068-5062)
- One display port and USB 3.0 cable for J3, 240 mm (P/N 1068-5064)

Relevant sections:
Platform components
Connector pinouts for building custom cables
Grounding
Cabling

Power cable

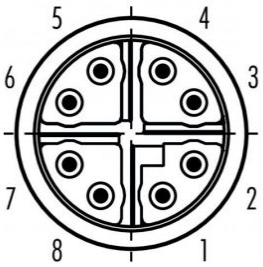
A 2-meter DC power input cable is included in the Kontron test cable kit or can be purchased separately by customers (P/N 1067-8665).
It is also possible for customers to build their own cable.

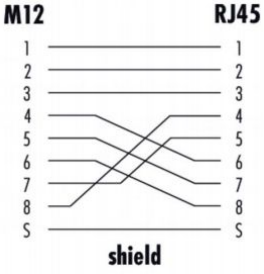


Pin	Signal description	Cable color
1	Power input+ (Vbat+)	Red
2	Power input- (Vbat-)	Black

Network cables

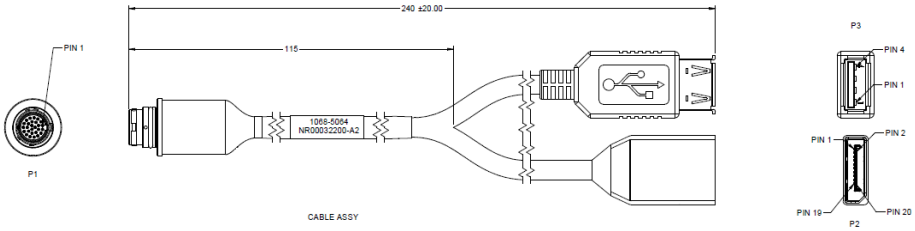
Nine 2-meter M12 to RJ45 shielded IP67 network cables are included in the Kontron test cable kit or can be purchased separately by customers (P/N 1064-9369).
It is also possible for customers to build their own cables.





Display port and USB 3.0 cable

A 240-millimeter display port and USB 3.0 cable for J3 (and J4 as custom option) can be purchased by customers (P/N 1068-5064) .
It is also possible for customers to build their own cable.



The Kontron test harness for J3 includes 2 devices:

- Display port (P2)
- USB 3.0 (P3) – US B Type A female co nnector

P1 is connected to the platform.

ODU AMC connector (P1)

Pin	Signal description	Pin	Signal description	Pin	Signal description
1	ML_Lane 1(p)	10	StdA_SSRX–	19	USB GND_DRAIN
2	ML_Lane 0(p)	11	DP_PWR	20	AUX_CH(n)
3	GND Lane 1	12	ML_Lane 3(n)	21	DP_PWR Return
4	D+	13	ML_Lane 2(n)	22	GND Lane 3
5	VBUS	14	ML_Lane 2(p)	23	ML_Lane 3(p)
6	StdA_SSTX +	15	ML_Lane 1(n)	24	GND Lane 2
7	StdA_SSTX–	16	ML_Lane 0(n)	25	Hot Plug
8	GND AUX	17	GND Lane 0	26	USB GND
9	StdA_SSRX+	18	D-	27	AUX_CH(p)

Display port (P2)

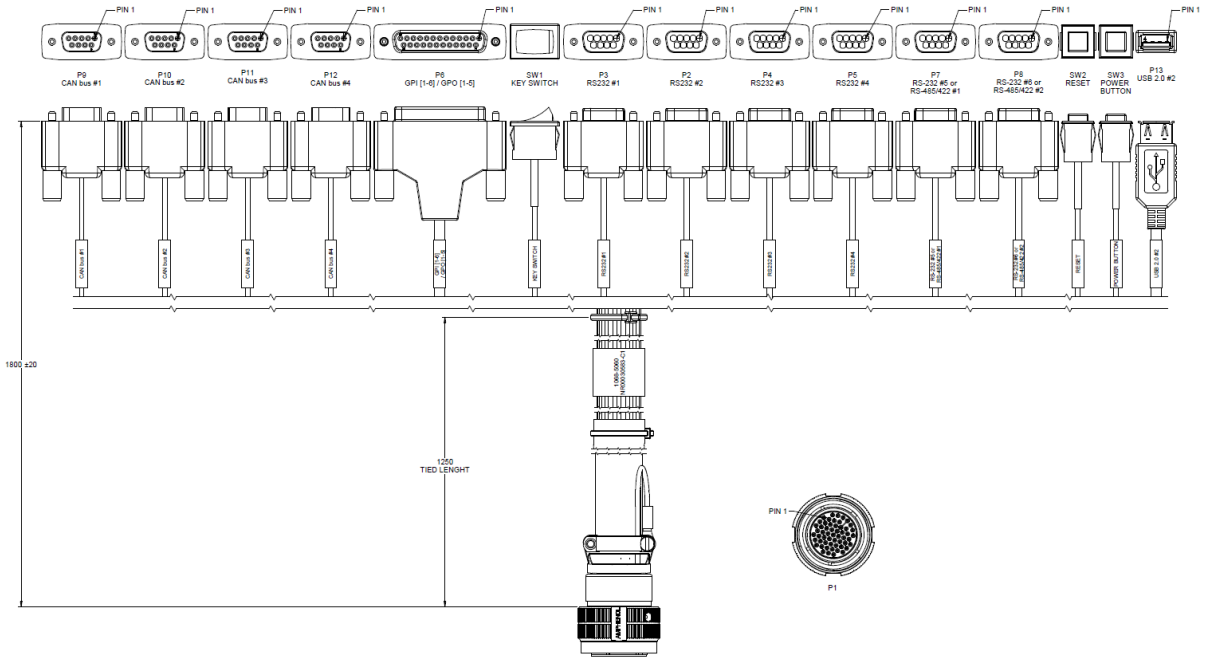
Pin	Signal description	Pin	Signal description
1	ML_Lane 0(p)	11	GND Lane 3
2	GND Lane 0	12	ML_Lane 3(n)
3	ML_Lane 0(n)	13	Reserved
4	ML_Lane 1(p)	14	Reserved
5	GND Lane 1	15	AUX CH(p)
6	ML_Lane 1(n)	16	GND AUX
7	ML_Lane 2(p)	17	AUX CH(n)
8	GND Lane 2	18	Hot Plug
9	ML_Lane 2(n)	19	DP_PWR Return
10	ML_Lane 3(p)	20	DPDP_PWR

USB 3.0 (P3)

Pin	Signal description	Pin	Signal description
1	VBUS	6	StdA_SSRX+
2	D-	7	GND_DRAIN
3	D+	8	StdA_SSTX-
4	GND	9	StdA_SSTX+
5	StdA_SSRX-		

I/O harness for J1

A 2-meter I/O test harness for J1 is included in the Kontron test cable kit or can be purchased separately by customers (P/N 1068-5060).
It is also possible for customers to build their own cable.



The Kontron test harness for J1 includes 15 devices:

- CAN bus #1 (P9) – DB9 female connector
- CAN bus #2 (P10) – DB9 female connector
- CAN bus #3 (P11) – DB9 female connector
- CAN bus #4 (P12) – DB9 female connector
- GPIs 1 to 6, GPOs 1 to 5 (P6) – DB25 female connector
- Ignition key switch (SW1) – Switch rocker ON-OFF
- Power button (SW3) – Switch push button OFF(ON)
- Reset (SW2) – Switch push button OFF(ON)
- RS-232 #1 (P3) – DB9 male DTE connector
- RS-232 #2 (P2) – DB9 male DTE connector
- RS-232 #3 (P4) – DB9 male DTE connector
- RS-232 #4 (P5) – DB9 male DTE connector
- RS-232 #5 or RS-485/422 #1 (P7) – DB9 male DTE connector
- RS-232 #6 or RS-485/422 #2 (P8) – DB9 male DTE connector
- USB 2.0 #2 (P13) – USB Type A connector

17-35, 55-pin connector (P1)

Pin	J1 signal description	Pin	J1 signal description	Pin	J1 signal description
1	USB #2 Vcc	20	GND	39	RS-232 #5 CTS or RS-485/422 #1 Rx- (B)
2	Input 5 Isolated	21	Input 6 Isolated	40	CAN bus #1 High
3	GPIO Isolated GND	22	Output 1 Isolated	41	GND
4	USB #2 D-	23	GND	42	RS-232 #1 Tx
5	USB #2 D+	24	RS-232 #5 RTS or RS-485/422 #1 Tx- (Z)	43	RS-232 #2 Tx
6	Power Button Input **NOTE:** Internal pull-up resistor. Lab only use. In production, use the ignition key switch.	25	CAN bus #4 High	44	RS-232 #2 Rx
7	Input 4 Isolated	26	RS-232 #4 Tx	45	GND
8	Input 3 Isolated	27	RS-232 #4 Rx	46	RS-232 #5 Rx or RS-485/422 #1 Rx+ (A)
9	Output 4 Isolated	28	GPIO Isolated GND	47	CAN bus #1 Low
10	CAN bus #3 High	29	Output 5 Isolated	48	CAN bus #2 High
11	GND	30	Output 2 Isolated	49	GND
12	RS-232 #3 Rx	31	RS-232 #5 Tx or RS-485/422 #1 Tx+ (Y)	50	GND
13	Reset Button Input **NOTE:** Internal pull-up resistor	32	CAN bus #4 Low	51	RS-232 #6 CTS or RS-485/422 #2 Rx- (B)
14	Output 3 Isolated	33	GND	52	RS-232 #6 Rx or RS-485/422 #2 Rx+ (A)
15	Input 2 Isolated	34	RS-232 #1 Rx	53	CAN bus #2 Low
16	Input 1 Isolated	35	GND	54	RS-232 #6 RTS or RS-485/422 #2 Tx- (Z)
17	CAN bus #3 Low	36	GND	55	RS-232 #6 Tx or RS-485/422 #2 Tx+ (Y)
18	GND	37	Ignition Key Switch Input **NOTE:** Internal pull-down resistor to Vbat-		
19	RS-232 #3 Tx	38	Vbat+ to drive Ignition Key Switch		

For information on how to use the Ignition key switch input, Reset button input and Power button input, refer to Platform power management .

CAN bus #1 (P9) – DB9 female connector

Pin	Signal description
1	Reserved
2	CAN bus L
3	GND
4	Reserved
5	Reserved
6	Reserved
7	CAN bus H
8	Reserved
9	Reserved

CAN bus #2 (P10) – DB9 female connector

Pin	Signal description
1	Reserved
2	CAN bus L
3	GND
4	Reserved
5	Reserved
6	Reserved
7	CAN bus H
8	Reserved
9	Reserved

CAN bus #3 (P11) – DB9 female connector

Pin	Signal description
1	Reserved
2	CAN bus L
3	GND
4	Reserved
5	Reserved
6	Reserved
7	CAN bus H
8	Reserved
9	Reserved

CAN bus #4 (P12) – DB9 female connector

Pin	Signal description
1	Reserved
2	CAN bus L
3	GND
4	Reserved
5	Reserved
6	Reserved
7	CAN bus H
8	Reserved
9	Reserved

GPIs 1 to 6, GPOs 1 to 5 (P6) – DB25 female connector

Pin	Signal description	Pin	Signal description
1	GPI Isolated 1	14	GPIO Isolated GND
2	GPO Isolated 1	15	GPIO Isolated GND
3	GPI Isolated 2	16	GPIO Isolated GND
4	GPO Isolated 2	17	GPIO Isolated GND
5	GPI Isolated 3	18	GPIO Isolated GND
6	GPO Isolated 3	19	GPIO Isolated GND
7	GPI Isolated 4	20	GPIO Isolated GND
8	GPO Isolated 4	21	GPIO Isolated GND
9	GPI Isolated 5	22	GPIO Isolated GND
10	GPO Isolated 5	23	GPIO Isolated GND
11	GPI Isolated 6	24	GPIO Isolated GND
12	GPIO Isolated GND	25	GPIO Isolated GND
13	GPIO Isolated GND		

Ignition key switch (SW1) – Switch rocker ON-OFF

Wire color	Signal description
White-Blue	Ignition key switch input
White	Vbat+

Power button (SW3) – Switch push button OFF(ON)

Wire color	Signal description
White-Blue	Power button input
White	GND

Reset (SW2) – Switch push button OFF(ON)

Wire color	Signal description
White-Blue	Reset button input
White	GND

RS-232 #1 (P3) – DB9 male DTE connector

This is the default serial console.

Pin	Signal description
1	Reserved
2	RS-232 Rx
3	RS-232 Tx
4	Reserved
5	GND
6	Reserved
7	Reserved
8	Reserved
9	Reserved

RS-232 #2 (P2) – DB9 male DTE connector

Pin	Signal description
1	Reserved
2	RS-232 Rx
3	RS-232 Tx
4	Reserved
5	GND
6	Reserved
7	Reserved
8	Reserved
9	Reserved

RS-232 #3 (P4) – DB9 male DTE connector

Pin	Signal description
1	Reserved
2	RS-232 Rx
3	RS-232 Tx
4	Reserved
5	GND
6	Reserved
7	Reserved
8	Reserved
9	Reserved

RS-232 #4 (P5) – DB9 male DTE connector

Pin	Signal description
1	Reserved
2	RS-232 Rx
3	RS-232 Tx
4	Reserved
5	GND
6	Reserved
7	Reserved
8	Reserved
9	Reserved

RS-232 #5 or RS-485/422 #1 (P7) – DB9 male DTE connector

For information on connector configuration, refer to Configuring serial ports .

Pin	Signal description - RS-232	Signal description - RS-485/422
1	Reserved	Reserved
2	RS-232 Rx	RS-485/422 Rx+ (A)
3	RS-232 Tx	RS-485/422 Tx+ (Y)
4	Reserved	Reserved
5	GND	GND
6	Reserved	Reserved
7	RS-232 RTS	RS-485/422 Tx- (Z)
8	RS-232 CTS	RS-485/422 Rx- (B)
9	Reserved	Reserved

RS-232 #6 or RS-485/422 #2 (P8) – DB9 male DTE connector

For information on connector configuration, refer to Configuring serial ports .

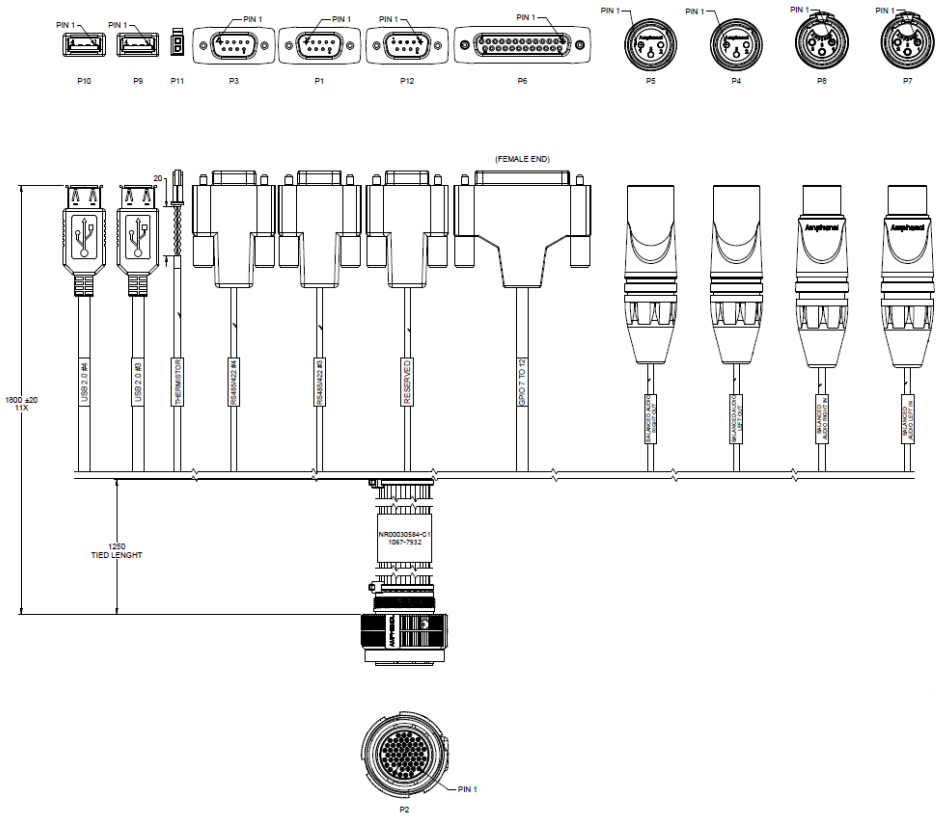
Pin	Signal description - RS-232	Signal description - RS-485/422
1	Reserved	Reserved
2	RS-232 Rx	RS-485/422 Rx+ (A)
3	RS-232 Tx	RS-485/422 Tx+ (Y)
4	Reserved	Reserved
5	GND	GND
6	Reserved	Reserved
7	RS-232 RTS	RS-485/422 Tx- (Z)
8	RS-232 CTS	RS-485/422 Rx- (B)
9	Reserved	Reserved

USB 2.0 #2 (P13) – USB Type A connector

Pin	Signal description
1	Vcc
2	D-
3	D+
4	GND

I/O harness for J2

A 2-meter I/O test harness for J2 is included in the Kontron test cable kit or can be purchased separately by customers (P/N 1068-5062).
It is also possible for customers to build their own cable.



The Kontron test harness for J2 includes 11 devices:

- Balanced audio left out (P4) – XLR male connector
- Balanced audio right out (P5) – XLR male connector
- Balanced audio left in (P7) – XLR female connector
- Balanced audio right in (P8) – XLR female connector
- GPIs 7 to 9, GPOs 6 to 8, GPIOs 1 to 7 (P6) – DB25 female connector
- M.2 or mPCIe pass-through I/O 1 to 8 (P12) – DB9 female connector
- RS-485/422 #3 (P1) – DB9 male DTE connector
- RS-485/422 #4 (P3) – DB9 male DTE connector
- Temperature (P11) – 2-pin micro-fit female connector
- USB 2.0 #3 (P9) – USB Type A connector
- USB 2.0 #4 (P10) – USB Type A connector

17-35, 55-pin connector (P2)

Pin	J2 signal description	Pin	J2 signal description	Pin	J2 signal description
1	Balanced Audio Left In-	20	GPIO 4 Isolated	39	RS-485/422 #4 Rx+ (A)
2	Balanced Audio Left In+	21	Input 7 Isolated	40	Balanced Audio Right Out-
3	RS-485/422 #3 Tx+ (Y)	22	M2 or mPCIe pass-through 5 **NOTE:** See table below for information specific to hardware configuration.	41	GPIO 3 Isolated
4	Balanced Audio Right In-	23	RS-485/422 #4 Isolated GND	42	GPIO 7 Isolated
5	Balanced Audio Right In+	24	RS-485/422 #4 Tx+ (Y)	43	GPIO Isolated GND
6	GND	25	USB 2.0 #3 Vcc	44	Input 8 Isolated
7	RS-485/422 #3 Isolated GND	26	GPIO 1 Isolated	45	GND
8	RS-485/422 #3 Tx- (Z)	27	M2 or mPCIe pass-through 3 **NOTE:** See table below for information specific to hardware configuration.	46	RS-485/422 #4 Rx- (B)
9	RS-485/422 #3 Rx+ (A)	28	M2 or mPCIe pass-through 4 **NOTE:** See table below for information specific to hardware configuration.	47	USB 2.0 #4 Vcc
10	Balanced Audio Left Out+	29	M2 or mPCIe pass-through 8 **NOTE:** See table below for information specific to hardware configuration.	48	GPIO Isolated GND
11	GND	30	M2 or mPCIe pass-through 7 **NOTE:** See table below for information specific to hardware configuration.	49	Output 7 Isolated
12	GPIO 5 Isolated	31	RS-485/422 #4 Tx- (Z)	50	GND
13	M2 or mPCIe pass-through 2 **NOTE:** See table below for information specific to hardware configuration.	32	Balanced Audio Right Out+	51	USB 2.0 #3 D+
14	M2 or mPCIe pass-through 1 **NOTE:** See table below for information specific to hardware configuration.	33	GND	52	USB 2.0 #3 D-
15	Output 6 Isolated	34	Input 9 Isolated	53	Output 8 Isolated
16	RS-485/422 #3 Rx- (B)	35	GPIO 6 Isolated	54	USB 2.0 #4 D+
17	Balanced Audio Left Out-	36	Thermistance	55	USB 2.0 #4 D-
18	GND	37	Thermistance		
19	GPIO 2 Isolated	38	M2 or mPCIe pass-through 6 **NOTE:** See table below for information specific to hardware configuration.		

Balanced audio left out (P4) – XLR male connector

Pin	Signal description
1	GND
2	Balanced Audio Out+
3	Balanced Audio Out-

Balanced audio right out (P5) – XLR male connector

Pin	Signal description
1	GND
2	Balanced Audio Out+
3	Balanced Audio Out-

Balanced audio left in (P7) – XLR female connector

Pin	Signal description
1	GND
2	Balanced Audio In+
3	Balanced Audio In-

Balanced audio right in (P8) – XLR female connector

Pin	Signal description
1	GND
2	Balanced Audio In+
3	Balanced Audio In-

GPIs 7 to 9, GPOs 6 to 8, GPIOs 1 to 7 (P6) – DB25 female connector

Pin	Signal description	Pin	Signal description
1	GPI Isolated 7	14	GPIO Isolated GND
2	GPO Isolated 7	15	GPIO Isolated GND
3	GPI Isolated 8	16	GPIO Isolated GND
4	GPO Isolated 8	17	GPIO Isolated GND
5	GPI Isolated 9	18	GPIO Isolated GND
6	GPIO Isolated 1	19	GPIO Isolated GND
7	GPIO Isolated 5	20	GPIO Isolated GND
8	GPIO Isolated 2	21	GPIO Isolated GND
9	GPIO Isolated 6	22	GPIO Isolated GND
10	GPIO Isolated 3	23	GPIO Isolated GND
11	GPIO Isolated 7	24	GPIO Isolated GND
12	GPIO Isolated 4	25	GPIO Isolated GND
13	GPO Isolated 6		

M.2 or mPCIe pass-through I/O 1 to 8 (P12) – DB9 female connector

Pin	Signal description
1	M.2 or mPCIe pass-through 1
2	M.2 or mPCIe pass-through 2
3	M.2 or mPCIe pass-through 3
4	M.2 or mPCIe pass-through 4
5	M.2 or mPCIe pass-through 5
6	M.2 or mPCIe pass-through 6
7	M.2 or mPCIe pass-through 7
8	M.2 or mPCIe pass-through 8
9	GND

NOTE: Contact Kontron for specific pinout information when M.2 or mPCIe cards are installed. Refer to the following section for pinout information based on specific hardware configuration: M.2 or mPCIe pass-through specific hardware configuration.

RS-485/422 #3 (P1) – DB9 male DTE connector

Pin	Signal description
1	RS-485/422 Tx- (Y)
2	RS-485/422 Tx+ (Z)
3	Reserved
4	Reserved
5	RS-485 Isolated GND
6	RS-485/422 Rx- (B)
7	RS-485/422 Rx+ (A)
8	Reserved
9	Reserved

RS-485/422 #4 (P3) – DB9 male DTE connector

Pin	Signal description
1	RS-485/422 Tx- (Y)
2	RS-485/422 Tx+ (Z)
3	Reserved
4	Reserved
5	RS-485 Isolated GND
6	RS-485/422 Rx- (B)
7	RS-485/422 Rx+ (A)
8	Reserved
9	Reserved

Temperature (P11) – 2-pin micro-fit female connector

Pin	Signal description
1	Thermistance
2	Thermistance

USB 2.0 #3 (P9) – USB Type A connector

Pin	Signal description
1	Vcc
2	D-
3	D+
4	GND

USB 2.0 #4 (P10) – USB Type A connector

Pin	Signal description
1	Vcc
2	D-
3	D+
4	GND

Validated operating systems

Table of contents

Status description

Status legend	Description
CERTIFIED	The product is certified by the OS vendor as compliant hardware
VALIDATED	The product was internally tested
TESTED CERT	The unit passed the certification tests, but the official OS vendor certificate was not published
PLANNED	Certification is planned
IN PROCESS	Certification has started

OS certification status

Operating system	Status
Ubuntu Server LTS 18.04.5 with kernel 5.4	VALIDATED
Ubuntu Server LTS 20.04.1 with kernel 5.4	VALIDATED

Security

- Establish a plan to change default user names and passwords. Refer to Configuring and managing users .
- The platform features a Trusted Platform Module (TPM). Determine your requirement with regards to hardware-based, security-related functions. Refer to Configuring the TPM in section Configuring UEFI BIOS options .
- Kontron can provide an encrypted UEFI/BIOS with an SSH key. Contact Kontron for more information.

For more information on security features, contact Kontron.

Getting started - configuring the CAN bus mezzanine with the AURIX safety MCU

Table of contents

Safety and regulatory

NOTICE	Before working with this product or performing instructions described in the getting started section or in other sections, read the Safety and regulatory information section pertaining to the product. Assembly instructions in this documentation must be followed to ensure and maintain compliance with existing product certifications and approvals. Use only the described, regulated components specified in this documentation.

Installing the AURIX MCU development environment and demo code

Perform all the configurations and tasks listed in page Installing the AURIX MCU development environment and demo code .

Executing the AURIX MCU demo code

Refer to page AURIX MCU demo code to execute the demo code.

Getting started - Platform integration and system access

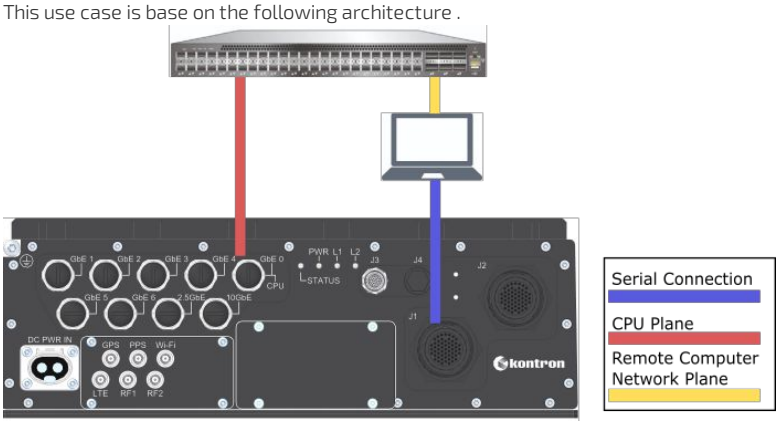
Table of contents

Safety and regulatory

NOTICE	Before working with this product or performing instructions described in the getting started section or in other sections, read the Safety and regulatory information section pertaining to the product. Assembly instructions in this documentation must be followed to ensure and maintain compliance with existing product certifications and approvals. Use only the described, regulated components specified in this documentation.

Introduction

This getting started section describes the assembly instructions for system components , interfaces and operating system accessing steps required to start operating the platform.
If the platform is equipped with the CAN bus mezzanine with the AURIX safety MCU , additional configuration steps will be required once the procedures in this getting started section are completed. Refer to Getting started - configuring the CAN bus mezzanine with the AURIX safety MCU for the additional steps required.
This use case is base on the following architecture .



To visualize the complete platform architecture or for further details, refer to Product architecture .

Assumptions

The scenario described in this getting started section is based on the following assumptions:

- The main connections of the system are as follows:
  - One 1GbE via Ethernet port 0 (**GbE 0**) connected to the CPU
  - One serial connection via the COMe RS-232 serial port of the J1 connector
- The OS access method must be through a serial connection
- The default IPv4 scheme is DHCP
- The Kontron test cable kit is used (includes a power input cable, an I/O harness for J1 and a network cable)
- Ubuntu 20.04 and the board support package (BSP) are installed on the platform
- Remote Desktop connectivity from a laptop to the S1901

Unboxing the platform

What's in the box

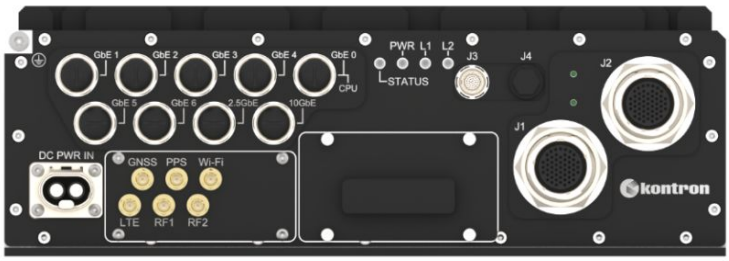
The box includes **one preassembled system platform** .

Step_1	Carefully remove the platform from its packaging.
Step_2	Remove the platform from the ESD bag using all the appropriate ESD protection measures.

NOTE: Additional material may be required to proceed with installation and configuratio n (refer to Material and information required for more information).

External connectors

This section is an overview of the connectors used for application benchmarking.
NOTE: Some connectors might not be present depending on product configuration.



NOTICE	Torque values are provided for mating connectors. These torques must not be exceeded as this will cause damage to the platform.

ID	Connector and function	Connector type	Supplier and P/N
GbE 0	1 Gigabit Ethernet to the COMe	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
GbE 1	1 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
GbE 2	1 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
GbE 3	1 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
GbE 4	1 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
GbE 5	1 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
GbE 6	1 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
2.5GbE	2.5 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in **Mating cycles:** 500	PCB receptacle: Harting 21033812806 Front panel shell: Harting 21033012000
10GbE	10 Gigabit Ethernet to the network switch	M12 8-pin X-Coded female **Torque:** 5 lbs-in	PCB receptacle:

		Mating cycles: 500	Harting 21033812806 Front panel shell: Harting 21033012000
DC PWR IN	DC power female connector	2-pole X-Coded Powerlok **Mating cycles:** 100	PCB receptacle: Amphenol PL082X-61-4
GPS	GNSS antenna	SMA female jack **Torque:** 4 lbs-in	PCB receptacle: AmphenolRF 132422-10
PPS	PPS signal (IN or OUT based on configuration)	SMA female jack **Torque:** 4 lbs-in	PCB receptacle: AmphenolRF 132422-10
Wi-Fi	Wi-Fi antenna	RP-SMA female jack **Torque:** 4 lbs-in	PCB receptacle: AmphenolRF 132422RP-10
LTE	LTE antenna	SMA female jack **Torque:** 4 lbs-in	PCB receptacle: AmphenolRF 132422-10
RF1	Project specific RF signal (typical usage second LTE antenna)	SMA female jack **Torque:** 4 lbs-in	PCB receptacle: AmphenolRF 132422-10
RF2	Project specific RF signal (typical usage second Wi-Fi antenna)	RP-SMA female jack **Torque:** 4 lbs-in	PCB receptacle: AmphenolRF 132422RP-10
J3	Display port 1 One USB 3.0	ODU AMC **Mating:** Push to connect, pull to disconnect **Mating cycles:** 5000	PCB receptacle: ODU GK1WAM-P27UB10-000L
J4	Display port 2 One USB 3.0	ODU AMC **Mating:** Push to connect, pull to disconnect **Mating cycles:** 5000	PCB receptacle: ODU GK1WAM-P27UB10-000L
J1	Four RS-232 (TX/RX) Two RS-232 (RX/TX/CTS/RTS) or RS-485/422 Four CAN FD (option) Six GPI and five GPO One USB 2.0 Power Reset Ignition key switch	MIL-DTL-38999 17-35, 55-pin straight female, Key N **Mating cycles:** 500	PCB receptacle: Amphenol AL07F17-35DN(P10)
J2	Two RS-485/422 Three GPI, three GPO and seven GPIO Two USB 2.0 Temperature sensor Balanced audio: • Two In • Two Out Eight pass-through I/O (option)	MIL-DTL-38999 17-35, 55-pin straight female, Key A **Mating cycles:** 500	PCB receptacle: Amphenol AL07F17-35DA(P10)

Material and information required

Configuration material

Relevant section:
Cabling

Item_1	One I/O harness for J1 (included in the Kontron test cable kit)
Item_2	One network cable to connect the laptop to the network switch
Item_3	One laptop with a serial port with a null modem connection and a network port

DC power cable and tooling

Relevant sections:
Platform components
Cabling

Item_1	One DC power input cable (included in the Kontron test cable kit)

Network cables and modules

Relevant section:
Platform components

Item_1	One network cable (included in the Kontron test cable kit)

Software required

Item_1	A terminal emulator such as **PuTTY** is installed on a remote computer.
Item_2	A Remote Desktop client is installed on a remote computer.
Item_3	Software to program the AURIX MCU (MemTool from Infineon running on Windows inside a virtual machine is the preferred method)

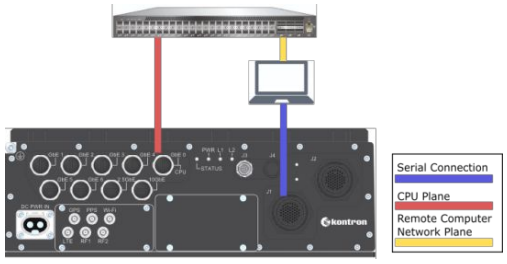
NOTE: Item_3 is required only when the CAN bus mezzanine with the AURIX safety MCU is installed.

> You now have the material and software required. Proceed with building and connecting the cables.

Connecting the cables

Connecting the serial cable and network cables

Step_1	Unscrew the plastic protector from the J1 connector.
Step_2	Connect the Kontron I/O harness for J1 to the J1 connector .
Step_3	Locate the DB9 connector of the I/O harness for J1 (labeled RS-232 #1) and connect it to the laptop. (Purple connection) This is the RS-232 serial port identified as ttyS0 in Linux. If necessary, connect a null modem to the laptop (remote computer).
Step_4	Connect the network cable from the laptop to the network switch. (Yellow connection)
Step_5	Using a flat screwdriver, unscrew the plastic protector from port GbE 0.
Step_6	Connect the network cable provided in the Kontron test cable kit in port GbE 0 and to the network switch. (Red connection) This port is identified as eno2 in Lin ux.

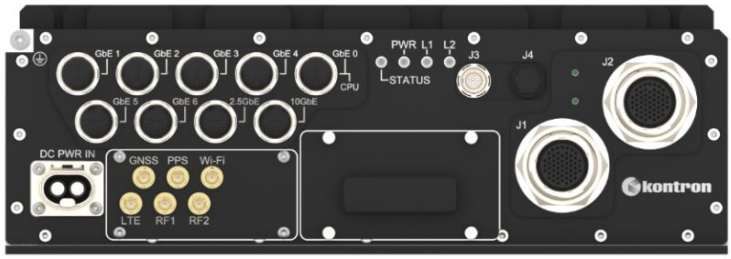


Kontron I/O harness for J1

In the Kontron test harness for J1, the DB9 male DTE adapter labeled P3 is connection RS-232 #1. The following table describes the pinout of the DB9 adapter.

Pin	Signal description
1	Reserved
2	RS-232 Rx
3	RS-232 Tx
4	Reserved
5	GND
6	Reserved
7	Reserved
8	Reserved
9	Reserved

Connecting the power cable

Step_1	Connect the DC power input cable to the platform.	

Powering ON the platform

NOTE: Pin 37 of connector J1 is the Ignition Key Switch Input pin.
This action is performed using the ignition key switch.

Electrical prerequisites

1	The power supply cable must be plugged in.
2	The ignition key switch input pin is not connected.

Procedure

Step_1	Connect the ignition key switch input pin to Vbat+. Resulting actions: • PSU: Powers ON • COMe: Powers ON • Network switch: Powers ON • Power LED: Turns ON • COMe serial port (console): Becomes available • AURIX MCU: Powers ON (if installed)

Relevant section:
Platform power management

>Proceed with operating the platform.

Logging into the operating system

Relevant sections:
Default user names and passwords
Accessing the operating system of a server

Getting the IP address of the OS

Prerequisites

1	An OS is installed.
2	A serial connection to the device is required.
3	A serial console tool is installed on the remote computer. • Speed (Baud): 115200 • Data bits: 8 • Stop bits: 1 • Parity: None • Flow Control: None • Recommended emulation mode: VT100+ **NOTE:** PuTTY is recommended.

Relevant section:
Platform components

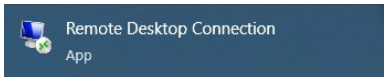
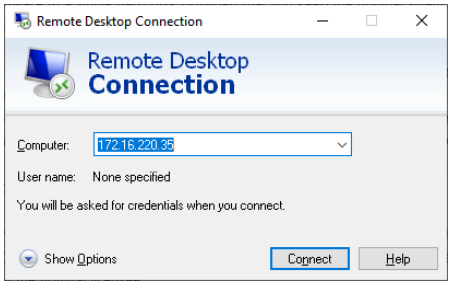
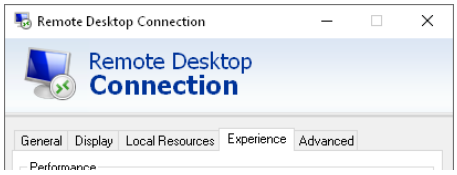
Procedure

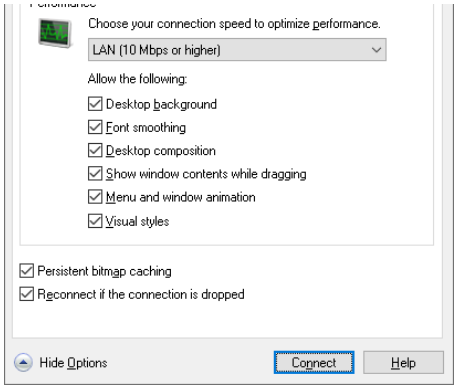
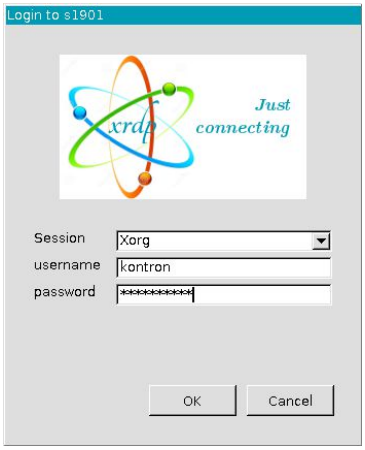
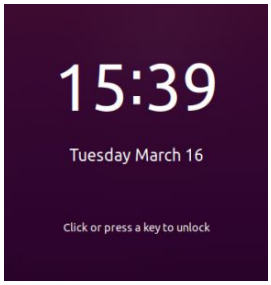
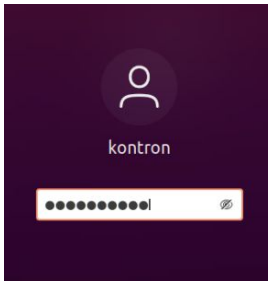
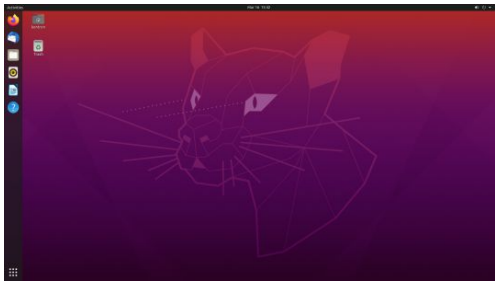
To obtain the list of default user names and passwords, refer to Default user names and passwords.
NOTE: In the Kontron test harness, RS-232 port #1 is the P3 connector of the J1 harness.

Step_1	From a remote computer with a serial connection to RS-232 #1 (the default serial console to COMe CPU), open a serial console tool, and start the communication between the console and the serial port to which the device is connected.	
Step_2	The OS start screen will be displayed. **NOTE:** If the OS is not displayed, press **Enter** .	
Step_3	Open a command line interface and use the following command to discover the OS IP address. LocalServer_OSPrompt:~# **ip a** In the image, the OS IP address of COMe eno2 is 172.16.220.35 and the MAC address is 00:e0:4b :70:09:a8.	

>With the OS IP address, you can access the platform's GNOME ™ desktop and the platform's CBIT Web interface.

Accessing the platform GNOME desktop

Step_1	Open Remote Desktop on the remote computer.	
Step_2	Enter the IP of the remote computer and click on **Show Options.**	
Step_3	Under tab **Experience** , select **LAN (10 Mbps or higher)** as the connection speed. Click on **Connect** .	

		
Step_4	Enter the appropriate credentials and click on **OK** .	
Step_5	Click or press any key.	
Step_6	Enter the password of the OS and press **Enter** .	
Step_7	The OS screen is displayed.	

Accessing the platform CBIT Web interface

CBIT is installed in demo mode by default and will start a Web service on port 80.

Step_1	Open a browser and enter the IP of eno2 to access the CBIT/kehm Web interface.

Accessing the network switch

Relevant sections:
Default user names and passwords
Accessing the operating system of a server
Linux devices

Getting the IP address of the network switch

Prerequisites

1	A network connection to the platform is required.
2	A serial console tool is installed on the OS. • Speed (Baud): 115200 • Data bits: 8 • Stop bits: 1 • Parity: None • Flow Control: None • Recommended emulation mode: VT100+ **NOTE:** Minicom is recommended.

Procedure

Access the OS.

Step_1	From the OS, o pen the OS CLI.	[redacted]
Step_2	Access the network switch. LocalServer_OSPrompt:~# **resize; sudo minicom -w -D /dev/ttyS6**	[redacted]
Step_3	Log in using the appropriate credentials.	[redacted]
Step_4	Discover the NOS IP address. LocalSwitchNOS_OSPrompt:~# **show ip interface brief** **NOTE:** If no DHCP server is found 1 minute after starting, the switch will default to 192.168.0.1/24.	[redacted]
Step_5	(Optional) To retry a DHCP request and get a NOS IP address that is not the default. LocalSwitchNOS_OSPrompt:~# **ip dhcp retry interface vlan 1** Wait one minute. LocalSwitchNOS_OSPrompt:~# **show ip interface brief**	[redacted]

Accessing the switch NOS Web UI

Prerequisites

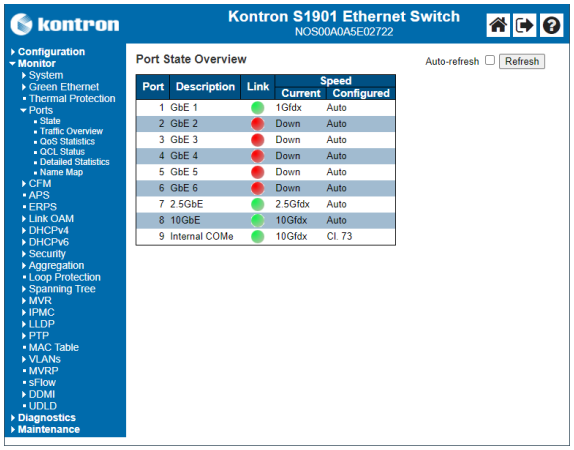
1	The network switch NOS IP address is known.
2	The remote computer has access to the switch NOS network subnet.

Browser considerations

HTTPS self-signed certificate	Upon connection to the Web UI, it is mandatory to accept the HTTPS self-signed certificate. For further information about accepting HTTPS self-signed certificates, please refer to your Web browser's documentation.
File download permission	File download from the site needs to be permitted. For further information about file download permission, please refer to your Web browser's documentation.
Cookies	Cookies must be enabled in order to access the website. For further information about enabling cookies, please refer to your Web browser's documentation.

NOTE: The procedure may vary depending on the browser used. Examples provided use Firefox.

Access procedure

Step_1	From a remote computer that has access to the switch network, open a browser window and enter the IP address discovered for the switch. *http://[SWITCH_IP]*	

Configuring the CAN bus mezzanine with the AURIX safety MCU

If the platform is equipped with the AURIX MCU, go to section Getting started - configuring the CAN bus mezzanine with the AURIX safety MCU and proceed with configurations.

Mechanical installation and precautions

ESD protections

Electrostatic discharge (ESD) can damage electronic components (e.g. disk drives and boards).
Look for this warning in the documentation as it indicates that the device is ESD sensitive and that precautions must be taken.

	ESD sensitive device! This equipment is sensitive to static electricity. Care must therefore be taken during all handling operations and inspections of this product in order to ensure product integrity at all times.

We recommend that you perform all the installation procedures described in the documentation at an ESD workstation. If this is not possible, apply ESD protections such as the following:

- Wear an antistatic wrist strap attached to a chassis ground (any unpainted metal surface) on the equipment when handling parts.
- Touch the metal chassis before touching an electronic component (e.g. a DIMM or board).
- Keep a part of your body (e.g. a hand) in contact with the metal chassis to dissipate the static charge while handling the electronic component.
- Avoid moving around unnecessarily.
- Use a ground strap attached to the front panel (with the bezel removed).
- Read and follow the safety precautions provided for a specific component by the manufacturer.

Unboxing

What's in the box

The box includes **one preassembled system platform** .

Step_1	Carefully remove the platform from its packaging.
Step_2	Remove the platform from the ESD bag using all the appropriate ESD protection measures.

Components installation and assembly

Table of contents

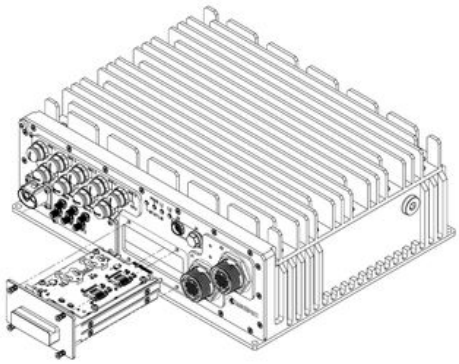
	**ESD sensitive device!** This equipment is sensitive to static electricity. Care must therefore be taken during all handling operations and inspections of this product in order to ensure product integrity at all times.

	When handling components, follow the precautions described in section ESD protections .

	Disconnect the power supply cord before servicing the product to avoid electric shock. If the product has more than one power supply cord, disconnect them all.

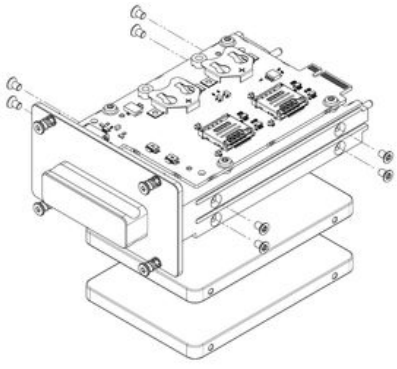
Installing components located in the front tray

Removing the front tray

Step_1	Using a T10 Torx screwdriver, unscrew the 4 captive screws and slide the front tray out of the enclosure.	

Installing or replacing one or two 2.5-in SSDs

Refer to Platform, modules and accessories for ordering information.
Up to two 2.5-inch SSDs can be installed in the front tray of the platform.

Step_1	**If installing an SSD for the first time** , insert the SSD in one of the front tray slots and secure the M3 screws using a T10 Torx screwdriver (6 lbs-in torque). **If replacing an SSD** , remove the M3 screws using a T10 Torx screwdriver, remove the SSD to replace from the tray, insert the new SSD in the tray and secure the M3 screws using a T10 Torx screwdriver (6 lbs-in torque) . **NOTE:** Slot #1 is the top slot and is compatible with SATA and NVMe, but the only signal available is SATA. Slot #2 is the bottom slot and is compatible with SATA and NVMe, and both signals are available.	

Replacing a battery

⚠CAUTION	**Risk of explosion if battery is replaced by an incorrect type.** Dispose of used batteries according to the instructions.

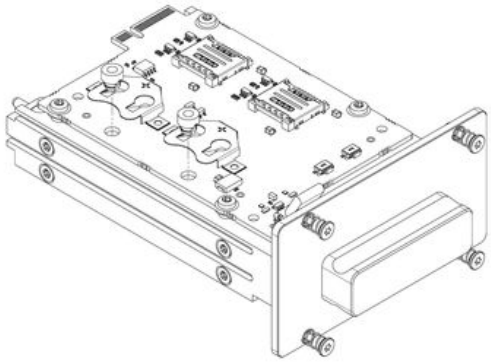
One battery is installed at the factory, but the S1901 platform can house up to two (type: CR2032; operating temperature range: -40°C to 85°C) .
If the platform is never powered, t he RTC circuit consumes a maximum of 10 µA/h with a 200 mA CR2032 battery:

- When one battery is installed and the platform is not powered, the battery can last approximately 20,000 hours (2.2 years).
- When two batteries are installed and the platform is not powered, the batteries can last approximately 40,000 hours (4.5 years).
- When the platform is powered, the battery is not used, extending its shelf life. For example, if the platform is powered 16 hours per day, one battery

will last approximately 7 to 8 years. When using two batteries, the shelf life of each battery remains the same.

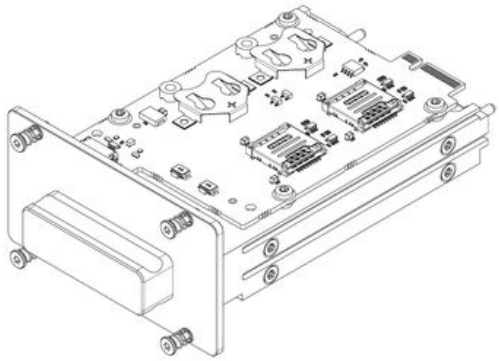
A capacitor located inside the platform keeps the RTC time for up to 8 hours when the front tray is taken out to replace one or both batteries.

- As a good practice, when two batteries are present, it is recommended to replace both at the same time.

Step_1	Lift one or both plastic pins with a small tool, depending on the number of batteries to replace. **NOTE:** Battery 1 is located in front of the tray, closer to the tray handle. Battery 2 is located at the back, away from the tray handle.	
Step_2	Remove the batteries by gently pushing on them from the back.	
Step_3	Insert new ones, making sure to respect the appropriate orientation and polarity.	
Step_4	Push the plastic pins back in place.	

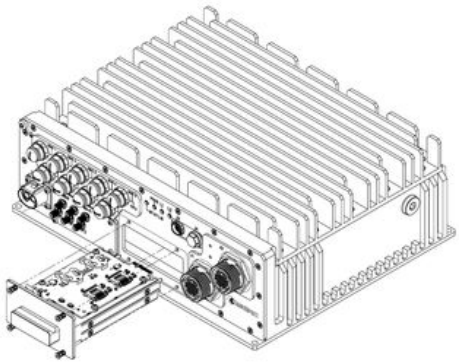
Replacing SIM cards

The S1901 platform contains up to four SIM cards.
The SIM cards are located on the same board as the batteries. **A capacitor located inside the platform keeps the RTC time for up to 8 hours** when replacing one or more SIM cards.

Step_1	Gently push on the SIM card with your finger to release it from the holding mechanism of the housing. Take it out of the housing.
	NOTE: The SIM cards are positioned as follows.
	• SIM 3 and SIM 4 are located in front, closer to the tray handle.
	  ◦ Top: SIM 4
	  ◦ Bottom: SIM 3
	• SIM 1 and SIM 2 are located in the back, away from the tray handle.
	  ◦ Top: SIM 2
	  ◦ Bottom: SIM 1
		

Step_2	Insert a new SIM card in the housing and gently push it until you hear a click.	
Step_3	(Optional) Repeat steps 1 and 2 if other SIM cards have to be replaced.	

Inserting the front tray

Step_1	Insert the front tray in the platform. Using a T10 Torx screwdriver, fasten the 4 captive screws to secure the front tray to the enclosure (6 lbs-in torque) .	

Installing the platform

Table of contents

NOTICE	Follow the corresponding instructions in this manual when installing/mounting the system. Observe all specified dimensions required for mounting included in the drawing with outline dimensions.

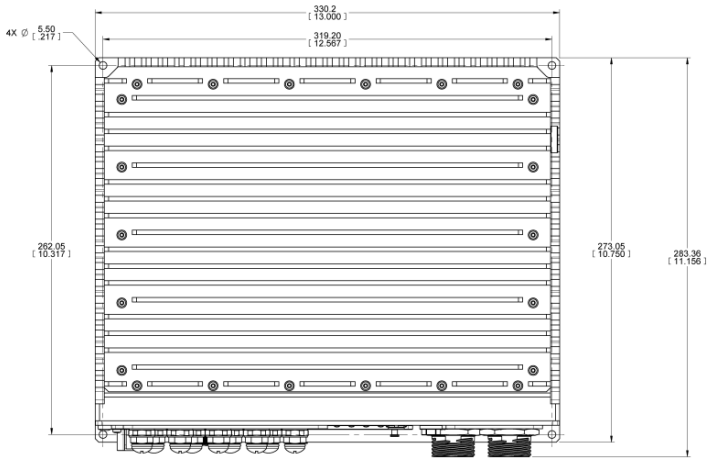
Precautions when installing the platform

- Respect the cooling clearances. Refer to the Clearances section.
- Make sure the mounting surface type and the mounting solution can safely support the load of the system. Refer to Specifications for dimensions and weights.
- Firmly attach the platform to a clean, flat and solid mounting surface using fastening materials suitable for the mounting surface.
- Follow the local/national regulations for grounding. A ground bonding measurement (between the system chassis ground and the mounting surface) should be conducted to ensure proper safety and EMI characteristics are maintained. Refer to Grounding for more information.
- The voltage feeds must not be overloaded. Adjust the cabling and the external overcharge protection to correspond with the electrical data indicated on the type label. Refer to Cabling for further details.
- If an ingress protection test must be performed once the platform is installed, make sure the clearances and mounting location are appropriate. Refer to Ingress protection test for information on the location of the hole to perform the test.

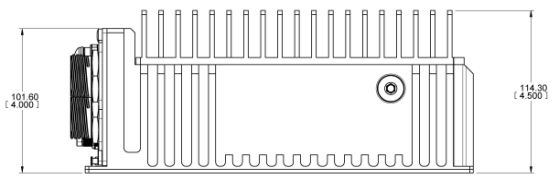
Mounting locations

There are 4 mounting thru-holes on the bottom with tool access from the top for mounting to any flat surface.
The system moun ting hole size is 5.5 mm (0.217 in) and is designed to support 10-24 (in), 10-32 (in), or M5 (metric) mounting hardware.
The recommended mounting configuration is to use a flat washer/locking washer/bolt stack, a self-locking bolt, or a simple bolt with a Loctite-like material.

Top view

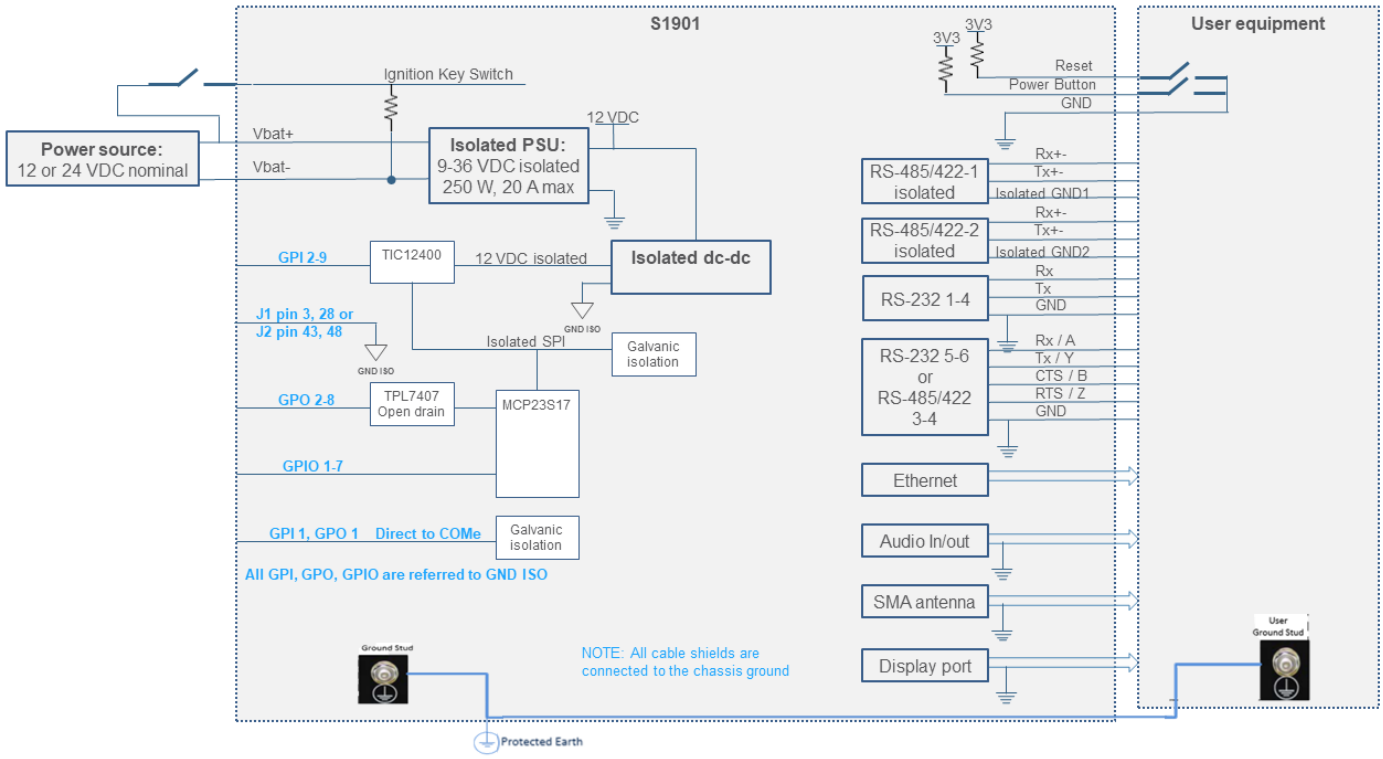


Side view



Grounding

The S1901 platform uses isolation between the power source and internal components. It also provides isolated GPIOs that must be referred to their local isolated ground.
Isolation prevents current flow between two communicating points, while still allowing the transmission of data and power signals between these points. Its purpose is to prevent high voltages from causing damages to sensitive electronic components or harming people. Furthermore, to ensure signal integrity, it eliminates ground loops in communication links that have big ground potential differences.
The logical ground is connected directly to the chassis ground on the faceplate for efficient ESD protection.
The torque for the ECM grounding screw located in front of the platform (top left) is 24 lbs-in.



Cabling

Table of contents

NOTICE	All mating connectors and cables used with the S1901 platform must have an IP67 rating. When custom cables are built, the mating connector manufacturer's instructions related to the IP67 rating must be followed. When S1901 connectors are not used, the protection caps of the connectors must be installed, as this ensures the platform complies with the IP67 rating. Note that the power input is covered with a dust cap that is not rated IP67 . If no mating connector is connected to this connector, the platform does not comply with the IP67 rating. An ingress protection test can be performed to confirm all connectors are correctly mated and the IP67 rating is achieved.

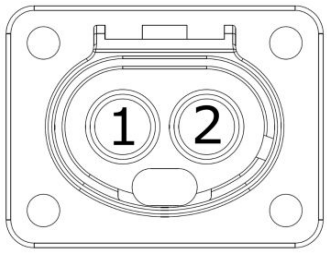
Power cables

DC power supply

A 2-meter DC power input cable is included in the Kontron test cable kit or can be purchased separately by customers (P/N 1067-8665).
It is also possible for customers to build their own cable according to the mating connector manufacturer's instructions.

DC power supply input connector

Description: 2-pole X-Coded Powerlok DC power female connector
Mating connector: Amphenol TPI PL182X-61-6P11 or equivalent
External connector pinout:



Pin	Signal description
1	Power input+ (Vbat+)
2	Power input- (Vbat-)

Material required to build the DC power cable

Item_1	One DC power connector
Item_2	Up to 10 AWG black stranded wire to build the power cable based on the length required **NOTE:** Use a wire gauge appropriate based on current and local wiring codes.
Item_3	Up to 10 AWG red stranded wire to build the power cable based on the length required **NOTE:** Use a wire gauge appropriate based on current and local wiring codes.
Item_4	(Optional) One Kontron test cable kit or one Kontron DC power input cable

Building the DC power cable

Step_1	Build the power cable using the appropriate material and according to the manufacturer's instructions and local wiring codes.

Other cables

Relevant sections:

Connector pinouts for building custom cables
Kontron test cables

Step_1	Build all the cables required using the appropriate material and according to the manufacturer's instructions and local wiring codes.

Ingress protection test

Table of contents

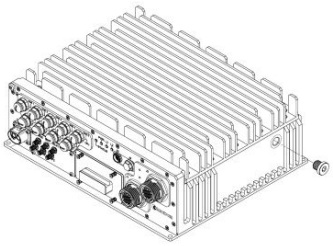
NOTICE	All mating connectors and cables used with the S1901 platform must have an IP67 rating. When custom cables are built, the mating connector manufacturer's instructions related to the IP67 rating must be followed. When S1901 connectors are not used, the protection caps of the connectors must be installed, as this ensures the platform complies with the IP67 rating. Note that the power input is covered with a dust cap that is not rated IP67 . If no mating connector is connected to this connector, the platform does not comply with the IP67 rating. An ingress protection test can be performed to confirm all connectors are correctly mated and the IP67 rating is achieved.

Equipment required

To perform the ingress protection test, the following equipment is required:

- An oil-free compressor
- An air hose
- An air spray nozzle
- A digital pressure gauge with a precision of 0.001 psi
- A custom fitting built to link the air spray nozzle and the testing hole on the platform (G 1/4)
- A 6 mm Allen key to remove and reattach the plug to the pressure port (200 lbs-in torque)
- A timer

Plug location



Procedure

Step_1	Make sure that all the connectors used have cables properly mated to them and that all unused connectors have their protection caps properly installed. Using a 6 mm Allen key , remove the plug from the pressure port.
Step_2	Attach the custom fitting built to the pressure port.
Step_3	Inject oil-free air into the platform until the reading on the pressure gauge is between 1.5 and 2.0 psi. **NOTE:** Make sure not to exceed 2.0 psi. If this occurs, bleed air from the system until the reading is below 2.0 psi.
Step_4	Wait 30 seconds for the system to stabilize.
Step_5	If the pressure gauge reading is lower than 1.5 psi, add more air into the system until the reading is higher than 1.5 psi, but lower than 2.0 psi. **NOTE:** Make sure not to exceed 2.0 psi. If this occurs, bleed air from the system until the reading is below 2.0 psi.
Step_6	Repeat steps 4 and 5 until the system is stable (gauge reading between 1.5 and 2.0 psi) after the 30-second wait period.
Step_7	Log the starting pressure indicated on the pressure gauge (in psi) and start the timer. Wait 1 minute.
Step_8	Log the end-of-test pressure indicated on the pressure gauge (in psi).
Step_9	Subtract the end-of-test pressure from the starting pressure and log the information.
Step_10	If pressure does not drop more than 0.01 psi in 1 minute, the test is a success.
Step_11	Once a test is successful, remove the custom fitting from the pressure port and reattach the plug (200 lbs-in torque). **NOTE:** It is crucial to respect the torque when reinstalling the plug. An improperly installed plug could compromise the IP67 rating of the platform.

Accessing platform components

Accessing the operating system of a server

Table of contents

An operating system can be accessed through various methods:

- Using a serial console (physical connection)
- Using SSH
- Using Remote Desktop

Refer to Description of system access methods for more information on the various paths.

Accessing the operating system from a serial console (physical connection)

Prerequisites

1	An OS is installed.
2	A serial connection to the device is required.
3	(Optional) A null modem is connected.
4	A serial console tool is installed on the remote computer. • Speed (Baud): 115200 • Data bits: 8 • Stop bits: 1 • Parity: None • Flow Control: None • Recommended emulation mode: VT100+ **NOTE:** PuTTY is recommended.

Relevant sections:

Kontron test cables

Connector pinouts for building custom cables

Access procedure

To obtain the list of default user names and passwords, refer to Default user names and passwords .

NOTE: If using the Kontron test harness, RS-232 port #1 is the P3 connector of the J1 harness.

Step_1	From a remote computer with a serial connection to RS-232 #1 (the default serial console to COMe CPU), open a serial console tool, and start the communication between the console and the serial port to which the device is connected.
Step_2	The OS start screen will be displayed. **NOTE:** If the OS is not displayed, press **Enter** .

Accessing the operating system using SSH

Prerequisites

1	An OS is installed.
2	An IP address is known: • eno2 Ethernet port IP address direct to COMe, if the network switch is not configured • eno1 Ethernet port IP address from the network switch, once the network switch is configured
3	The remote computer has access to the OS subnet.

Relevant section:

Discovering platform IP addresses

Access procedure

To obtain the list of default user names and passwords, refer to Default user names and passwords .

Step_1	Using the OS IP address, access the platform's operating system using SSH.

Accessing the operating system using Remote Desktop

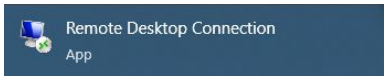
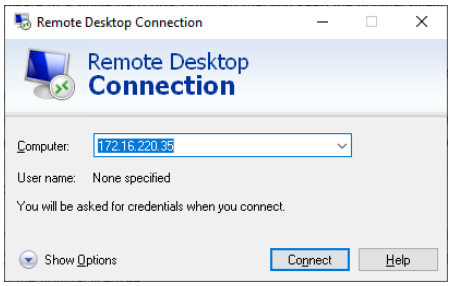
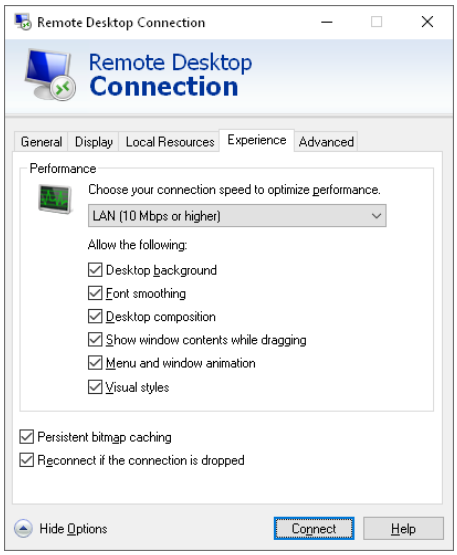
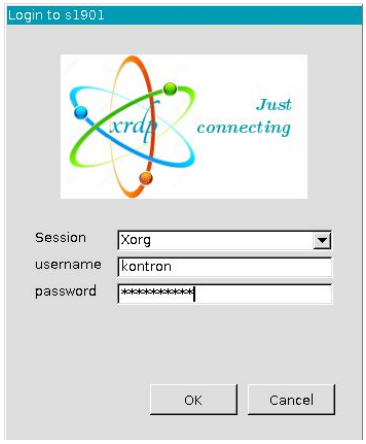
Prerequisites

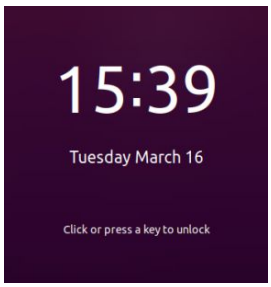
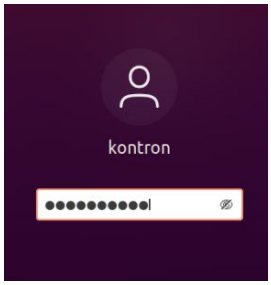
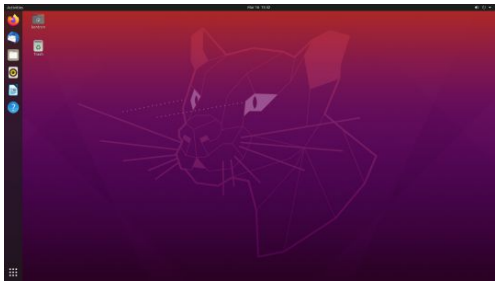
1	An OS with Remote Desktop is installed. **NOTE:** When Kontron installs a demo OS, Remote Desktop is installed by default.
2	An IP address is known: • eno2 Ethernet port IP address direct to COMe, if the network switch is not configured • eno1 Ethernet port IP address from the network switch, once the network switch is configured
3	The remote computer has access to the OS subnet.

Relevant section:
Discovering platform IP addresses

Access procedure

To obtain the list of default user names and passwords, refer to Default user names and passwords.

Step_1	Open Remote Desktop on the remote computer.	
Step_2	Enter the IP of the remote computer and click on **Show Options**.	
Step_3	Under tab **Experience** , select **LAN (10 Mbps or higher)** as the connection speed. Click on **Connect** .	
Step_4	Enter the appropriate credentials and click on **OK** .	

Step_5	Click or press any key.	
Step_6	Enter the password of the OS and press **Enter** .	
Step_7	The OS screen is displayed.	

Accessing the UEFI BIOS

Table of contents

Accessing the UEFI BIOS from a serial console (physical connection)

Prerequisites

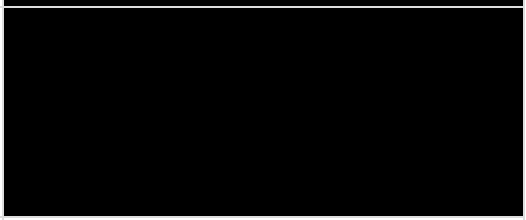
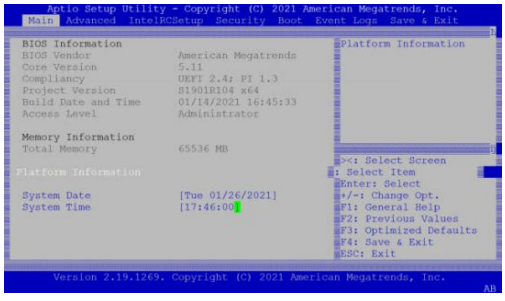
1	A serial connection to the device is required.
2	(Optional) A null modem is connected.
3	A serial console tool is installed on the remote computer. • Speed (Baud): 115200 • Data bits: 8 • Stop bits: 1 • Parity: None • Flow Control: None • Recommended emulation mode: VT100+ **NOTE:** PuTTY is recommended.

Relevant sections:

Connector pinouts for building custom cables
Kontron test cables
Platform power management

Access procedure

To obtain the list of default user names and passwords, refer to Default user names and passwords .
NOTE: If using the Kontron test harness, RS-232 port #1 is the P3 connector of the J1 harness.
NOTE: If using the Kontron test harness, the power button is the SW3 connector of the J1 harness and the reset switch is the SW2 connector of the J1 harness.

Step_1	From a remote computer with a serial connection to RS-232 #1 (the default serial console), open a serial console tool, and start the communication between the console and the port to which the device is connected.	
Step_2	Perform a power cycle on the platform or a server reset using the appropriate switch. **NOTE:** When a power cycle or server reset is performed, it may take a few seconds for the UEFI/BIOS sign on screen to display.	
Step_3	When the UEFI/BIOS sign-on screen is displayed, press the specified key to enter the UEFI/BIOS setup menu. **NOTE:** It may take a few seconds for the UEFI/BIOS sign-on screen to display the confirmation message "Entering Setup…".	
Step_4	The UEFI/ BIOS sign on screen displays "Entering Setup…". **NOTE:** It will take several seconds to display and enter the UEFI/BIOS setup menu.	
Step_5	The UEFI/BIOS setup menu is displayed.	

Accessing the switch network operating system

Table of contents

The switch NOS can be accessed through various methods:

- Using a serial console from the integrated server
- Using the switch NOS Web UI
- Using SSH from a remote computer

Refer to Description of system access methods for more information on the various paths.

Accessing the NOS using a serial console from the integrated server

This is the direct internal path from the integrated server to the network switch.

Prerequisites

1	An OS is installed.
2	A network connection to the platform is required.
3	The physical serial connection to the network switch is present within the platform.
4	A serial console tool is installed on the remote computer. • Speed (Baud): 115200 • Data bits: 8 • Stop bits: 1 • Parity: None • Flow Control: None • Recommended emulation mode: VT100+

Relevant sections:
Product architecture
Linux devices

Access procedure

To obtain the list of default user names and passwords, refer to Default user names and passwords .
Access the OS using your preferred method. Refer to Accessing the operating system of a server for access instructions.

Step_1	From the OS, o pen the OS CLI.	
Step_2	Access the network switch. LocalServer_OSPrompt:~# **resize ; sudo minicom -w -D /dev/ttyS6**	
Step_3	Log in using the appropriate credentials. **NOTE:** If "username:" is not displayed, press **Enter** .	

Accessing the switch NOS using the Web UI

Prerequisites

1	The network switch NOS IP address is known.
2	The remote computer has access to the switch NOS network subnet.

Relevant sections:
Discovering platform IP addresses
Configuring VLANs

Configuring VLANs

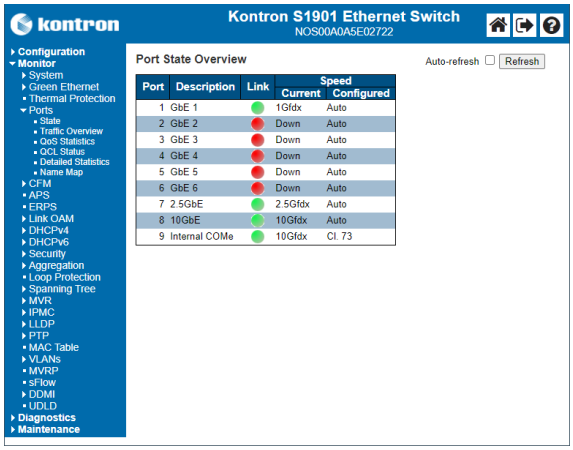
Browser considerations

HTTPS self-signed certificate	Upon connection to the Web UI, it is mandatory to accept the HTTPS self-signed certificate. For further information about accepting HTTPS self-signed certificates, please refer to your Web browser's documentation.
File download permission	File download from the site needs to be permitted. For further information about file download permission, please refer to your Web browser's documentation.
Cookies	Cookies must be enabled in order to access the website. For further information about enabling cookies, please refer to your Web browser's documentation.

NOTE: The procedure may vary depending on the browser used. Examples provided use Firefox.

Access procedure

To obtain the list of default user names and passwords, refer to Default user names and passwords .

Step_1	From a remote computer that has access to the switch network, open a browser window and enter the IP address discovered for the switch. *http://[SWITCH_IP]*	

Accessing the switch NOS using SSH from a remote computer

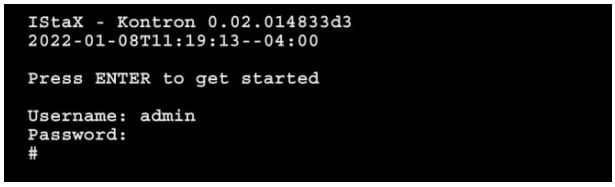
Prerequisites

1	The network switch NOS IP address is known.
2	The remote computer has access to the switch NOS network subnet.
3	An SSH client tool is installed on the remote computer. NOTE: PuTTY is recommended for Windows environments and SSH is recommended for Linux environments.

Relevant sections:
Discovering platform IP addresses
Configuring VLANs

Access procedure

To obtain the list of default user names and passwords, refer to Default user names and passwords .

Step_1	From a remote computer, open an SSH client tool and connect with the NOS IP address.	
Step_2	Log in the switch NOS CLI using the appropriate credentials.	

Accessing the AURIX MCU

Table of contents

The CAN bus mezzanine with the AURIX safety MCU can be accessed through various methods:

- Using a serial console from the integrated server
- Using a specific USB port

Refer to Description of system access methods for more information on the various paths.

Accessing the AURIX MCU using a serial console from the integrated server

Prerequisites

1	An OS is installed.
2	A network connection to the platform is required.
3	The physical serial connection to the network switch is present within the platform.
4	A serial console tool is installed on the remote computer. • Speed (Baud): 115200 • Data bits: 8 • Stop bits: 1 • Parity: None • Flow Control: None • Recommended emulation mode: VT100+
5	The backspace key parameter of the terminal emulator must be set to DEL.

Relevant sections:
Product architecture
Linux devices
Minicom problems

Procedure

To obtain the list of default user names and passwords, refer to Default user names and passwords .
Access the OS using your preferred method. Refer to Accessing the operating system of a server for access instructions.

Step_1	From the OS, o pen the OS CLI.	
Step_2	Access the AURIX MCU. LocalServer_OSPrompt:~# **resize ; sudo minicom -w -D /dev/ttyS1** **NOTE:** If "Shell>" is not displayed, press **Enter** .	

Accessing the AURIX MCU using a specific USB port

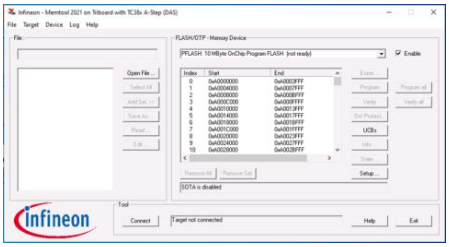
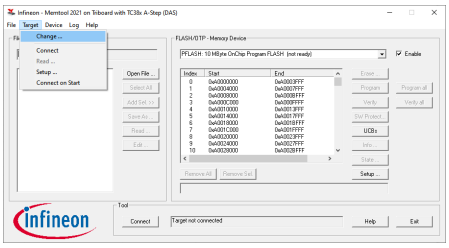
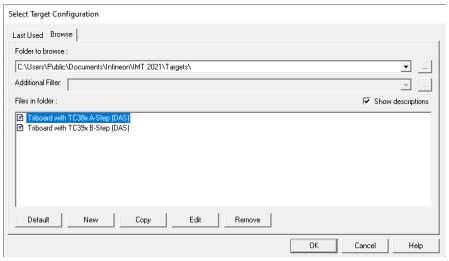
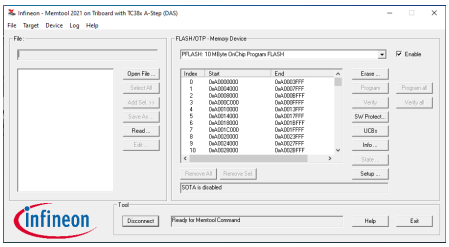
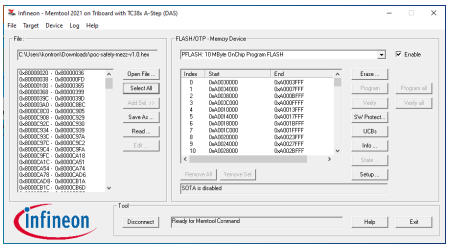
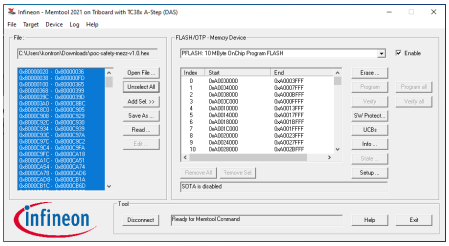
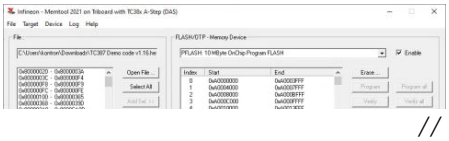
Prerequisites

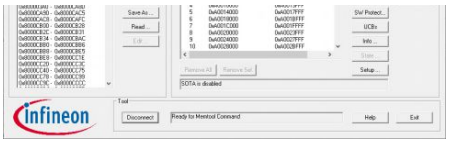
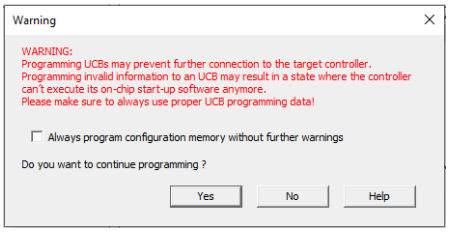
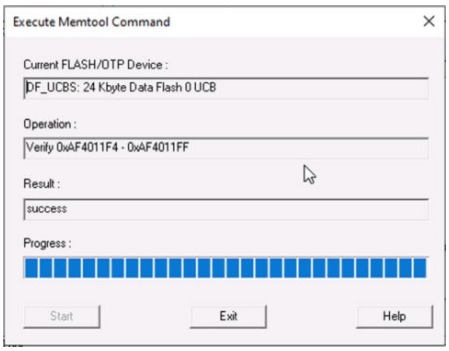
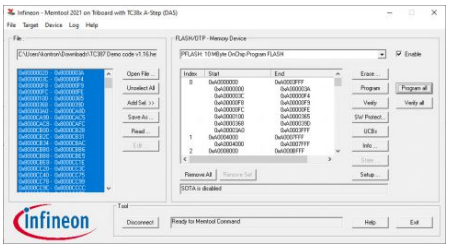
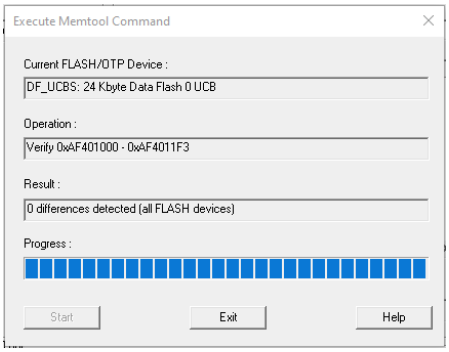
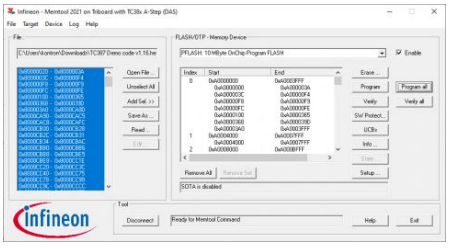
1	The programming environment of the AURIX MCU must be configured.
2	The USB port must be assigned to the virtual machine (if a virtual machine is used for programming).
3	All relevant BSP components are installed.

Relevant sections:
Installing the board support package

Procedure

To obtain the list of default user names and passwords, refer to Default user names and passwords .
Access the OS using your preferred method. Refer to Accessing the operating system of a server using Remote Desktop for access instructions .

Step_1	From the Linux CLI, run the aurix-pre-prog.sh script. This will disable the watchdog timer of the Multi Voltage Safety Micro Processor Supply (TLF35584) installed on the CAN bus mezzanine with the AURIX safety MCU and perform a reset. LocalServer_OSPrompt:~# **sudo aurix-pre-prog.sh**	
Step_2	From the VM desktop, double click on the **Infineon Memtool** icon.	
Step_3	Click on **Connect** .	
Step_4	Confirm the AURIX target is the right one. From the **Target** tab, click on **Change...**	
Step_5	Select the proper AURIX target and click on **OK** . The possible targets are: • TC387: Select Triboard with TC38x A-Step (DAS) • TC397: Select Triboard with TC39x B-Step (DAS)	
Step_6	Click on **Connect** and confirm that **Ready for MemTool Command** is displayed in the **Tool** status text box.	
Step_7	Click on **Open File** . Select the appropriate .hex file and click on **Open** .	
Step_8	Click on **Select All** .	
Step_9	Click on **Add Sel >>** and click on **Program all** .	

		
Step_10	Click on **Yes** .	
Step_11	Once **success** is displayed in the **Result** text box, click on **Exit** .	
Step_12	Click on **Verify all** .	
Step_13	Message **0 differences detected (all FLASH devices)** should be displayed in the **Result** text box. Once it is, click on **Exit** .	
Step_14	Click on **Exit** .	
Step_15	From the Linux CLI, run the aurix-post-prog.sh script. This will re-enable the watchdog timer of the Multi Voltage Safety Micro Processor Supply (TLF35584) installed on the CAN bus mezzanine with the AURIX safety MCU and perform a reset. LocalServer_OSPrompt:~# **sudo aurix-post-prog.sh**	

Discovering platform IP addresses

Table of contents

Discovering the IP address of the OS

This IP address is the minimum required to access the platform.
The OS IP address can be discovered:

- Using a serial console (physical connection)
- From the DHCP server

Discovering the IP address of the OS using a serial console (physical connection)

Prerequisites

1	An OS is installed.
2	A serial connection to the device is required.
3	(Optional) A null modem is connected.
4	A serial console tool is installed on the remote computer. • Speed (Baud): 115200 • Data bits: 8 • Stop bits: 1 • Parity: None • Flow Control: None • Recommended emulation mode: VT100+ **NOTE:** PuTTY is recommended.

Relevant sections:
Connector pinouts for building custom cables
Kontron test cables

Procedure

To obtain the list of default user names and passwords, refer to Default user names and passwords .
For details on network architecture, refer to Network devices .
NOTE: If using the Kontron test harness, RS-232 port #1 is the P3 connector of the J1 harness.

Step_1	From a remote computer with a serial connection to RS-232 #1 (the default serial console to COMe CPU), open a serial console tool, and start the communication between the console and the serial port to which the device is connected.	
Step_2	The OS start screen will be displayed. **NOTE:** If the OS is not displayed, press **Enter** .	
Step_3	Open a command line interface and use the following command to discover the OS IP address. LocalServer_OSPrompt:~# **ip a** In the image, the OS IP address of COMe eno2 is 172.16.220.35 and the MAC address is 00:e0:4b :70:09:a8.	

Discovering the IP address of the OS from the DHCP server

Prerequisites

1	The MAC address of the COMe is known.

Relevant section:

MAC addresses

Procedure

Step_1	Look in the DHCP server to find the IP address based on the MAC address.

NOTE: To discover the IP addresses of the automotive Ethernet ports, the procedure is the same as the one described above.

Discovering the IP address of the network switch

The network switch IP address can be discovered:

- Using the network switch CLI
- From the DHCP server

Discovering the network switch IP address using the network switch CLI

Prerequisites

1	A network connection to the platform is required.
2	A serial console tool is installed on the OS. • Speed (Baud): 115200 • Data bits: 8 • Stop bits: 1 • Parity: None • Flow Control: None • Recommended emulation mode: VT100+ **NOTE:** Minicom is recommended.

Relevant section:
Linux devices

Procedure

To obtain the list of default user names and passwords, refer to Default user names and passwords .
Access the OS using your preferred method. Refer to Accessing the operating system of a server for access instructions.

Step_1	From the OS, o pen the OS CLI.	
Step_2	Access the network switch. LocalServer_OSPrompt:~# **resize; sudo minicom -w -D /dev/ttyS6**	
Step_3	Log in using the appropriate credentials.	
Step_4	Discover the NOS IP address. LocalSwitchNOS_OSPrompt:~# **show ip interface brief** **NOTE:** If no DHCP server is found 1 minute after starting, the switch will default to 192.168.0.1/24.	
Step_5	(Optional) To retry a DHCP request and get a NOS IP address that is not the default. LocalSwitchNOS_OSPrompt:~# **ip dhcp retry interface vlan 1** Wait one minute. LocalSwitchNOS_OSPrompt:~# **show ip interface brief**	

Discovering the network switch IP address from the DHCP server

Prerequisites

1	The MAC address of the network switch (MAC_BASE) is known.

Relevant section:

MAC addresses

Procedure

Step_1	Look in the DHCP server to find the IP address based on the MAC address.

Default user names and passwords

Table of contents

NOTE : For security reasons, it is important to change the default user names and passwords as soon as possible. Refer to Configuring and managing users .

Switch network operating system (NOS)

User name	Password
admin	ready2go

Operating system

The user name and password are application-specific.
However, if Kontron provided an operating system, the credentials will be the following:

User name	Password
kontron	ready2go

UEFI/BIOS

No password is set by default.

Software installation and deployment

Preparing for operating system and board support package installation

Step_1	Choose the operating system needed based on the requirements of your application (Ubuntu 18.04.5 or 20.04.1 with kernel 5.4 is recommended).
Step_2	Confirm the OS version to be installed is compatible with the board support package (BSP). If the OS used is not an OS validated by Kontron, refer to the BSP section to determine the drivers required and the configuration parameters to be adapted to your OS.
Step_3	Download the ISO file of the OS to be installed.

For a list of known compatible operating systems, refer to Validated operating systems .
For information on the board support package, refer to Installing the board support package .

Installing an operating system on a server

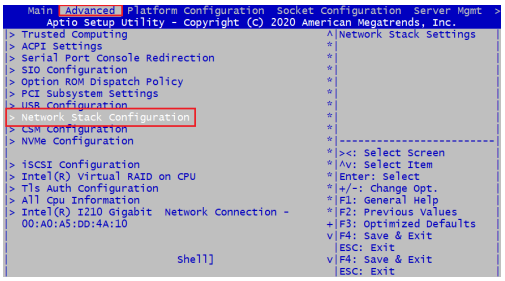
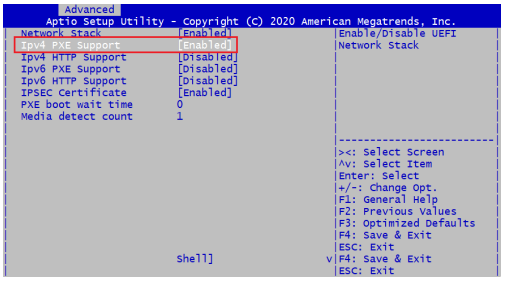
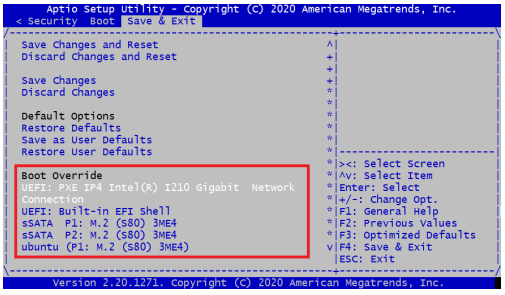
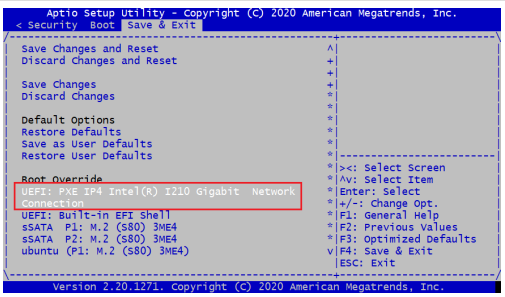
Table of contents

The operating system can be installed using the following methods:

- Using PXE (Boot from LAN)
- Using a USB storage device

Installing an OS on a server using PXE (Boot from LAN)

Relevant section:
Accessing the UEFI BIOS
NOTE: Using Boot from LAN requires a PXE server architecture.

Step_1	From the UEFI/BIOS setup menu, select the **Advanced** tab and then the **Network Stack Configuration** submenu.	
Step_2	Set **Network Stack** to **Enabled** . Set **IPv4 PXE Support** or **IPv6 PXE Support** , depending on the application, to **Enabled** .	
Step_3	Reboot the system and access the UEFI/BIOS setup menu again.	
Step_4	Navigate to the **Save & Exit** menu and then to the **Boot Override** section.	
Step_5	Choose the PXE option desired.	

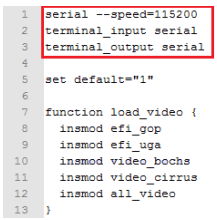
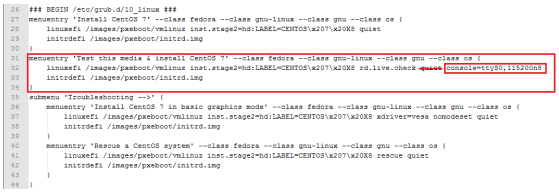
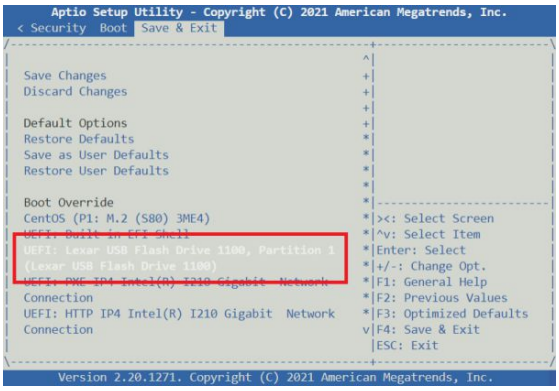
Installing an OS on a server using a USB storage device

	When the use of software tools such as Clonezilla live, a backup utility, restore Ubuntu live or any other live CD or live DVD is planned, lines must be added in the code. The following instruction "console=ttyS0,115200n8" must be added to the grub instructions. For more details on the process, refer to the OS manufacturer's documentation. Useful keywords for an Internet search are: console=ttyS0 grub.

Relevant sections:

Connector pinouts for building custom cables (to use USB ports available from the front plate connectors using custom cables)
Kontron test cables (to use USB ports available from the front plate connectors using Kontron test cables)
Components installation and assembly (to use the USB port located under the front panel)

This procedure can be performed using any of the platform's USB ports.

Step_1	Create a bootable USB key using the appropriate software. **NOTE:** RUFUS is recommended. Use ISO mode.	
Step_2	Open the USB directory in a remote computer.	
Step_3	Navigate to EFI then BOOT (e.g.: E:/EFI/BOOT/).	
Step_4	Open the grub.cfg file with any text editor.	
Step_5	Edit the file and add the following line on the top to activate the serial installation: *serial --speed=115200* *terminal_input serial* *terminal_output serial*	
Step_6	Scroll down the file and edit the menu used to start the OS installation. Edit as follows: • Remove the **quiet** argument, if present. • Add the **console=ttyS0,115200n8** argument. An example is provided in the image for CentOS 7.	
Step_7	Save the file and eject the USB key.	
Step_8	Insert the USB key into one of the USB ports of the front panel.	
Step_9	Power on the platform and access the UEFI/BIOS setup menu.	
Step_10	Navigate to the **Save & Exit** menu and then to the **Boot Override** section.	
Step_11	Choose the USB option desired.	

Enabling the ignition key switch

Relevant section:
Platform power management

When the platform is delivered with an Ubuntu OS installed by Kontron, the behavior of the OS power button is configured to handle the ignition key switch (IKS).
If a custom OS is used, the behavior of the OS power button needs to be configured to handle the IKS. This configuration is vital for proper platform operation.
The procedure below programs the power button so the following happens:

- When the IKS is set to OFF, the platform shuts down and remains shut down (there is no more power).
- When the IKS is toggled, the system reboots (the toggle time must be lower than the timeout implemented in the FPGA).

Step_1	Create a file. LocalServer_OSPrompt:~$ **sudo vi /etc/acpi/events/power**
Step_2	Open the file created and copy the following instructions in it. **event=button/power** **action=/usr/bin/logger "ACPI_POWER_BTTN: rebooting"** **action=/sbin/reboot**
Step_3	Activate the new service. LocalServer_OSPrompt:~$ **sudo service acpid restart**

Installing the board support package

Table of contents

The Board Support Package (BSP) is a collection of components to be installed on top of a vanilla operating system to gain access to system specific devices or functionalities.
The BSP is provided in the form of a .zip archive and is specific to the target OS. If the OS used is not an OS validated by Kontron, refer to the extracted BSP scripts to determine the drivers required and the configuration parameters to adapt for your OS.
BSP components associated with the current hardware configuration of the platform must be installed.
If hardware changes were made within the platform since last BSP installation, reinstall the relevant BSP components. This will ensure all the functionalities are available and operational.

	Before proceeding with BSP installation, read the Change Log section of the README file (README.md) contained in the BSP archive to view limitations and functionalities that are not operational. It is also recommended to read the release note document provided with the BSP documentation to take into consideration all the known issues related to specific devices. The document name will have a structure similar to this one: S1901-Evotrac_BSP_ubuntu20.04.pdf

Procedure for BSP installation

Access the OS. Refer to Accessing the operating system of a server for access instructions.
The BSP archive contains a README file with comprehensive details about component installation and options. Refer to it for detailed instructions or follow this series of steps for a simple installation.

Extracting the BSP

Step_1	Copy and extract the BSP on the target server. NOTE: The steps that follow must be performed from the directory where the BSP archive is extracted. Administrative privileges (sudo) are required to install BSP components.

Installing BSP components common to all S1901 platforms

For additional information on components such as MAX6581, ADS7830 and SIC451, refer to Monitoring platform components .
For the location of the accelerometer chip, refer to Accelerometer location .
The steps below are an example. They should be performed in the sequence shown.

Step_1	If secure boot is enabled on the system, it is required to enroll a new module signature key in the UEFI/BIOS. 1. Run the script **. /SECUREBOOT/secure-boot-setup.sh** . Enter a password for the key and reboot the unit to enroll the new signature key in the UEFI/BIOS from the console. 2. When the unit reboots, a blue menu will appear. 3. Press **OK** , then enter the **MOK management** utility. 4. Select **Enroll MOK** and press **Continue** . 5. Press **YES** and enter the password you created for the key. 6. Press **Reboot** . 7. When all components are installed, it is highly recommended to delete the Private key created during the secure boot setup using this command **rm /var/lib/shim-signed/mok/MOK.priv** .
Step_2	Install the COMe-BBD7 component by running the following command: **./COMe-bBD7/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.
Step_3	Install the MAX6581 component by running the following command: **./MAX6581/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.
Step_4	Install the ADS7830 component by running the following command: **./ADS7830/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.
Step_5	Install the SIC451 component by running the following command: **./SIC451/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.
Step_6	Install the PCA9698 and PCA9539 components by running the following command: **./PCA953x/install.sh** Upon successful installation, the "Installation Complete" message should be displayed. NOTE: S1901 platforms can be equipped with PCA9698 or PCA9539 chips. This step will install what is required no matter the S1901 platform.

Step_7	Install the Marvell M.2 automotive Ethernet component by running the following command: **./MARVELL_M.2_AUTO/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.
Step_8	Install the RS485 component by running the following command: **./RS485/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.
Step_9	Install the LM5056 component by running the following command: **./LM5056/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.
Step_10	If a NVidia MXM card is present, install the NVidia driver using the following command: **./NVIDIA_MxM/install.sh** Upon successful installation, the "[INFO] NVIDIA driver - Installation completed" message should be displayed.
Step_11	Install the KEAPI component by running the following command: **./KEAPI/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.
Step_12	Install the PBIT component by running the following command: **./PBIT/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.
Step_13	Install the CBIT component by running the following command: **./CBIT/install.sh --demo** Upon successful installation, the "Installation Complete" message should be displayed. Installing CBIT in demo mode starts a Web service on port 80. It will then be possible to open a browser window using the eno1 or eno2 IP address to view the CBIT Web interface. If you do not want to enable this Web service, use command **install.sh** .
Step_14	Install the BNO055 component by running the following command: **./BNO055/install.sh** Upon successful installation, the "Installation Complete" message should be displayed. See instructions in the README file to compile the driver and program.
Step_15	Install the CANBUS component by running the following command: **./CANBUS/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.
Step_16	Install the UEFI Firmware Tools component by running the following command: **./UEFI-firmware-tools/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.
Step_17	Install the PTP for Linux Tools component by running the following command: **./PTP4L/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.
Step_18	Install the TPM component by running the following command: **./TPM2/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.
Step_19	Install the ALSA component by running the following command: **./ALSA/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.

Installing BSP components specific to the S1901 EvoTRAC platform

For additional information on the components, refer to Monitoring platform components .

Step_1	Install the FT422H component by running the following command: **./FT4222-gpio/install.sh** Upon successful installation, the "Installation Complete" message should be displayed.

Verifying operating system and board support package installation

Table of contents

Relevant sections:

Product architecture

Accessing the operating system of a server

Verifying for support devices

	All the results and commands may vary depending on the operating system and the devices added.

Most commands below require administrative privileges (sudo).

Step_1	Reboot the OS as recommended, then access the OS command prompt.	
Step_2	Install **ethtool** , **pciutils** and **tpm2-tools** using the package manager, and update the operating system packages. Example for Ubuntu: LocalServer_OSPrompt:~# **apt-get update** LocalServer_OSPrompt:~# **apt-get install pciutils** LocalServer_OSPrompt:~# **apt-get install ethtool** LocalServer_OSPrompt:~# **apt-get install tpm2-tools** **NOTE:** Updating the packages may take a few minutes.	
Step_3	Verify that no error messages or warnings are displayed in **dmesg** using the following commands. LocalServer_OSPrompt:~# **dmesg | grep -i fail** LocalServer_OSPrompt:~# **dmesg | grep -i error** LocalServer_OSPrompt:~# **dmesg | grep -i warning** LocalServer_OSPrompt:~# **dmesg | grep -i "call trace"** **NOTE:** If there are any messages or warnings displayed, refer to the operating system's documentation to fix them.	
Step_4	Verify that the DIMMs are detected. LocalServer_OSPrompt:~# **free -h**	[redacted]
Step_5	Verify that all the storage devices are detected. LocalServer_OSPrompt:~# **lsblk**	[redacted]

Verifying the board support package installation - components available on all S1901 platforms

The following commands verify that most BSP components are correctly installed and configured. Most commands below require administrative privileges (sudo).

If your BSP version is higher than 17, refer to the README.md file included in your BSP package. It will contain the latest information and information on optional components.

Step_1	Run the following command to make sure sensor driver configuration files are installed. LocalServer_OSPrompt:~# **sensors** **NOTE:** The output should be similar to the following sample, although actual readings will vary.	[redacted]

		[redacted]
Step_2	If a NVidia MXM card is present, run the following command to verify the driver is installed. LocalServer_OSPrompt:~# **nvidia-smi**	
	[redacted]	
Step_3	Run the following command to verify that KEAPI is installed. LocalServer_OSPrompt:~# **ktool gen GetBoardInfo** **NOTE:** For more information about how to use KEAPI/ktool, read the "KEAPI_3_doc.pdf" file in the BSP package.	[redacted]
Step_4	Run the following command to verify that PBIT is installed and configured. LocalServer_OSPrompt:~# **kdiag stat**	[redacted]
Step_5	Run the following command to verify that CBIT/kehm is installed and configured. LocalServer_OSPrompt:~# **/usr/share/kehm/kehm_example.exe -i /usr/share/kehm/S1901-sensors.xml**	[redacted]

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03 Driver Version: 460.32.03 CUDA Version: 11.2 |
|-------------------------------+----------------------+----------------------+
| GPU Name         Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan Temp Perf    Pwr:Usage/Cap|          Memory-Usage | GPU-Util Compute M. |
|                               |                      |              MIG M. |
|===============================+======================+======================|
|   0   Quadro T1000        Off | 00000000:06:00.0 Off |                  N/A |
| N/A    70C    P0     8W / N/A |       0MiB / 3911MiB |       0%     Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
| GPU    GI   CI        PID   Type     Process name               GPU Memory |
|        ID   ID                                                      Usage |
|=============================================================================|
| No running processes found                                                  |
+-----------------------------------------------------------------------------+
```

		[redacted]
Step_6	Run the following command to verify that BNO055 drivers and tools are installed. LocalServer_OSPrompt:~# **/opt/getbno055/getbno055 -t inf** **NOTE:** The output should be similar to the following sample, although actual readings will vary.	[redacted]
Step_7	Run the following command to verify that CAN Bus tools are installed. LocalServer_OSPrompt:~# **lspcan -a** **NOTE:** CAN Bus cards must be installed to get an output.	pcanpcifd0 CAN1 82 80MHz 500k+2M CLOSED 0.00 0 0 0 pcanpcifd1 CAN2 83 80MHz 500k+2M CLOSED 0.00 0 0 0 pcanpcifd2 CAN3 84 80MHz 500k+2M CLOSED 0.00 0 0 0 pcanpcifd3 CAN4 85 80MHz 500k+2M CLOSED 0.00 0 0 0
Step_8	Run the following command to verify that UEFI Firmware tools are installed. LocalServer_OSPrompt:~# **kbtc get all**	[redacted]
Step_9	Run the following command to verify that TPM2 tools are installed. LocalServer_OSPrompt:~# **tpm2_getrandom --hex 8** **NOTE:** Command output will vary at each execution.	[redacted]
Step_10	Run the following command to verify that audio tools are installed. LocalServer_OSPrompt:~# **alsamixer** **NOTE:** You should see **Card: USB AUDIO CODEC** on the top left of the screen.	
Step_11	Run the following command to verify that RS485 scripts are installed. LocalServer_OSPrompt:~# **ls -w1 /usr/local/bin/s1901-rs485***	[redacted]
Step_12	Run the following command to verify that PTP4Linux tools are installed. LocalServer_OSPrompt:~# **ptp4l**	[redacted]

Verifying the board support package installation - components specific to EvoTRAC

Defining the I2C bus number

All devices linked to one of the I2C buses of the platform will be assigned a number when the platform boots. For configuration and operation purposes, the number assigned to the SMBus and the kempld must be known.

Step_1	From the OS CLI, run the following command to determine the SMBus number. LocalServer_OSPrompt:~# **echo "SMBUS is on $(i2cdetect -l | grep "SMBus I801" | cut -f 1 | cut -d - -f 2)"** **NOTE:** The answer will be SMBUS is on [SMBUS_NO].
Step_2	Run the following command to determine the kempld number. LocalServer_OSPrompt:~# **echo "KEMPLD is on $(i2cdetect -l | grep "i2c-kempld" | cut -f 1 | cut -d - -f 2)"** **NOTE:** The answer will be KEMPLD is on [KEMPLD_NO].

Confirming the GPIO drivers are present

Most commands below require administrative privileges (sudo).

Step_1	Run the following command to verify that the GPIO devices are present. This will mean the drivers are installed. LocalServer_OSPrompt:~# **gpiodetect** The following GPIO devices must be present (more could be listed): • gpio-kempld • exar_gpio0 • pca9539-[SMBUS_NO]-0074 • pca9539-[KEMPLD_NO]-0077	LocalServer_OSPrompt:~ # gpiodetect gpiochip0 [gpio_ich] (76 lines) gpiochip1 [gpio-kempld] (8 lines) gpiochip2 [exar_gpio0] (16 lines) gpiochip3 [pca9539-1-0077] (16 lines) gpiochip4 [pca9539-0-0074] (16 lines)
Step_2	Run the following command to verify that the GPIO devices are present. This will mean the drivers are installed. LocalServer_OSPrompt:~# **s1901-gpiodetect** The following GPIO devices must be present: • tic12400 • mcp23s17-0 • mcp23s17-1	LocalServer_OSPrompt:~ # s1901-gpiodetect gpiochip0 [tic12400] (8 lines) [gpi 2-9] gpiochip1 [mcp23s17-0] (7 lines) [gpo 2-8] gpiochip2 [mcp23s17-1] (7 lines) [gpio 1-7]

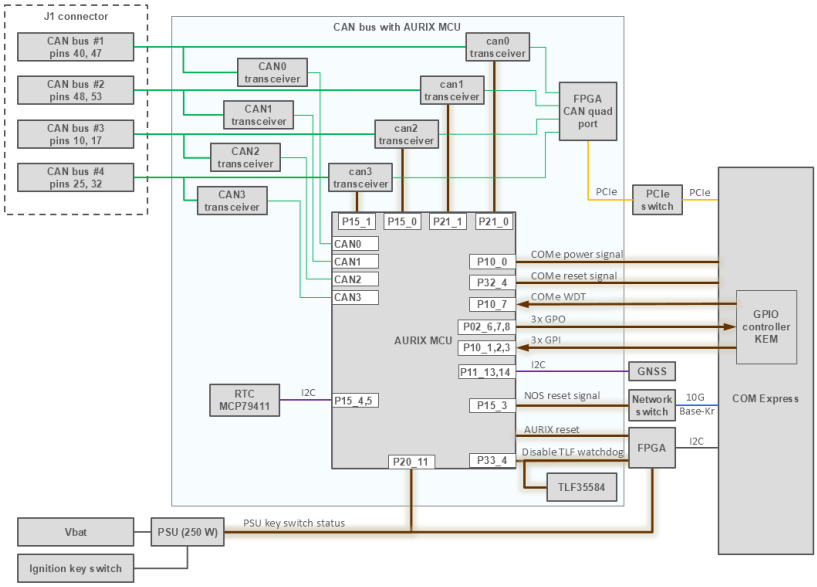
Installing the AURIX MCU development environment and demo code

Table of contents

This procedure describes how to setup MemTool from Infineon on a Windows 10 virtual machine to compile and program software on the AURIX MCU of the CAN bus mezzanine.
NOTE: The compilation process can be performed either on the server host, on a virtual machine running Windows, or on another computer. However, the programming process must be done on the host that has access to the AURIX DAS USB programming port.

Relevant sections:
Product architecture
AURIX MCU demo code

Interconnections between the AURIX MCU and the platform

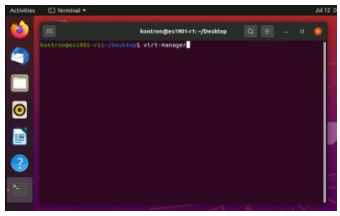
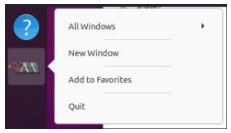
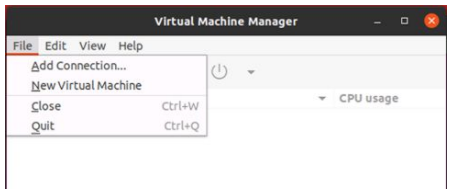
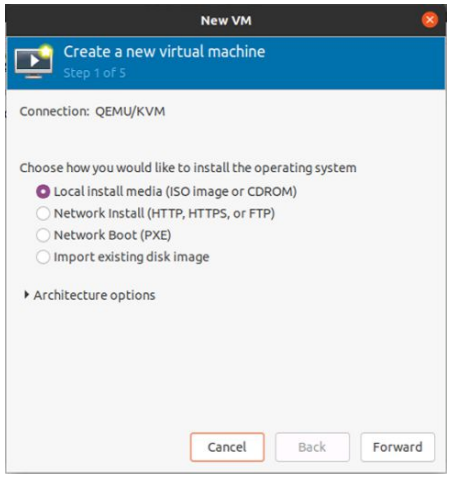
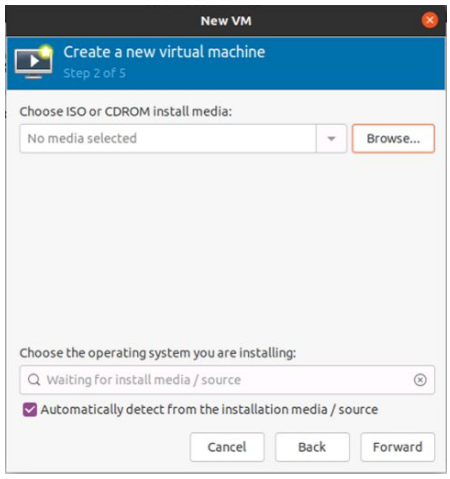


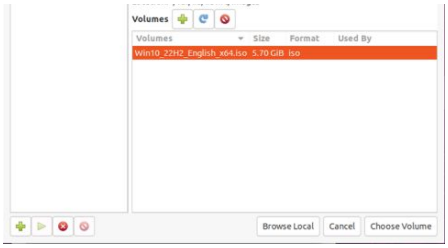
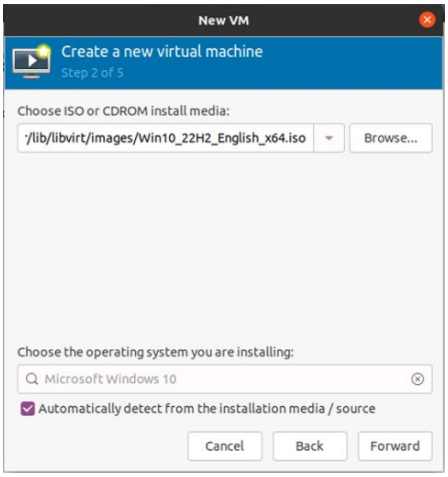
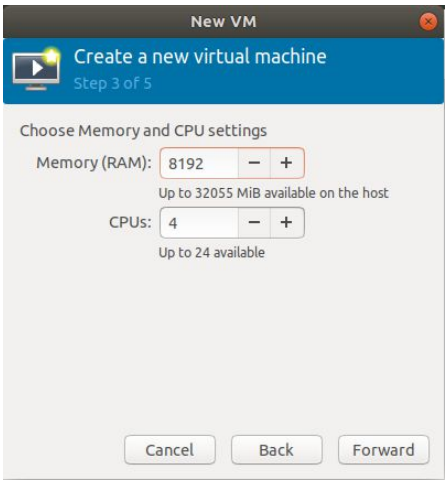
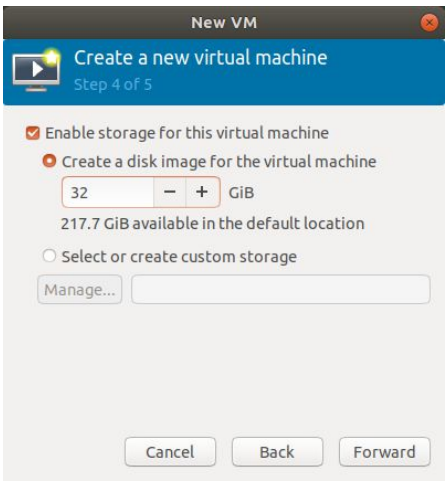
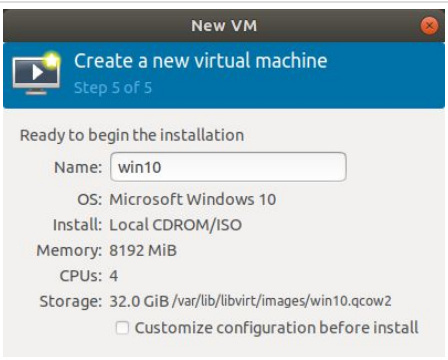
Installing the AURIX MCU development environment

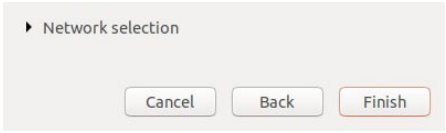
Installing the KVM hypervisor

Step_1	Update the OS. LocalServer_OSPrompt:~# **sudo apt update**
Step_2	Install qemu. LocalServer_OSPrompt:~# **sudo apt install -y qemu qemu-kvm qemu-system qemu-utils**
Step_3	Install libvirt. LocalServer_OSPrompt:~# **sudo apt install -y libvirt-clients libvirt-daemon-system virtinst virt-manager**
Step_4	Add the current user to the libvirt group. LocalServer_OSPrompt:~# **sudo usermod -aG libvirt $(whoami)**
Step_5	Reboot the platform. LocalServer_OSPrompt:~# **sudo reboot**

Creating a Windows 10 virtual machine

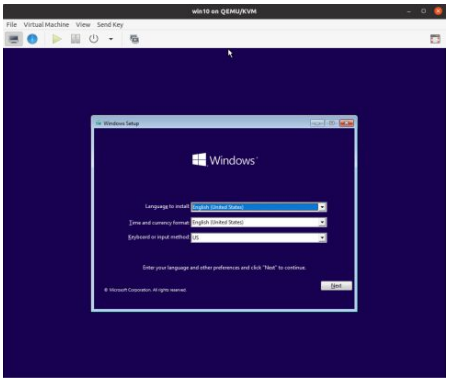
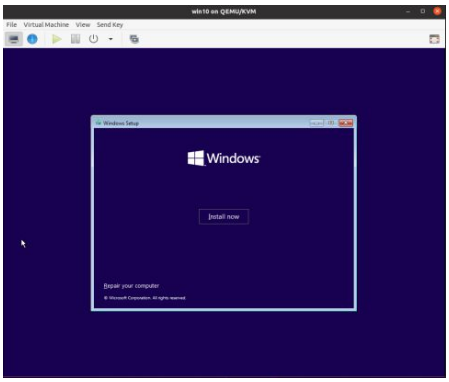
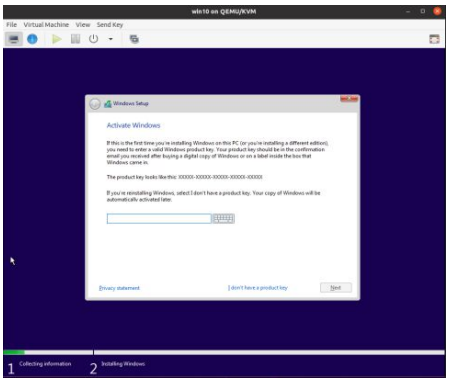
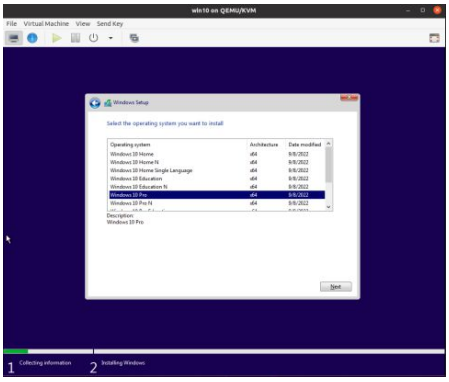
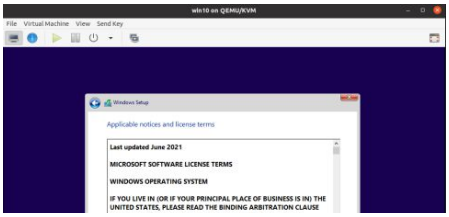
Step_1	Right click on the desktop and select **Open in Terminal** .	
Step_2	Start the KVM. LocalServer_OSPrompt:~# **virt-manager**	
Step_3	Right click on the Virtual Machine Manager icon and click on **Add to Favorites** .	
Step_4	Select **New Virtual Machine** .	
Step_5	Select **Local install media (ISO image or CDROM)** . Click on **Forward** .	
Step_6	Click on **Browse** .	
Step_7	Download the ISO file of the OS to install and save it in the following folder: **/var/lib/libvirt/images**	
Step_8	Select the OS ISO file downloaded. In this example, we use **Win10_22H2_English_x64.iso** . Click on **Choose Volume** .	

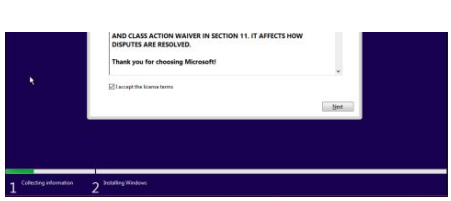
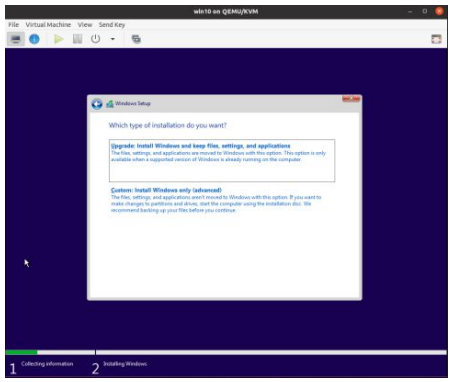
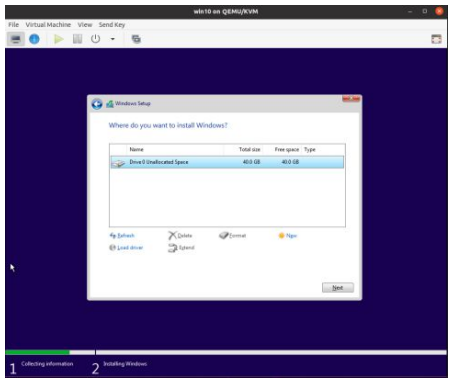
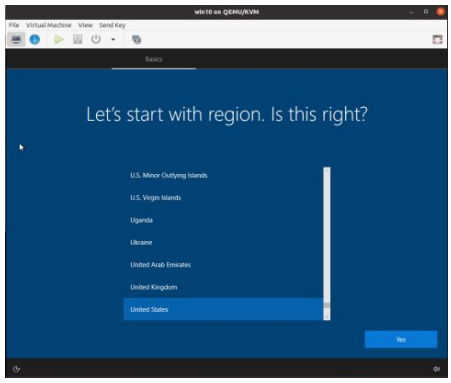
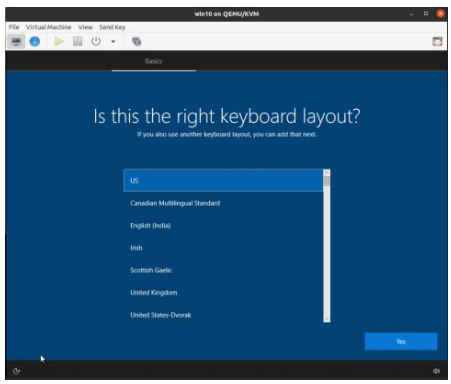
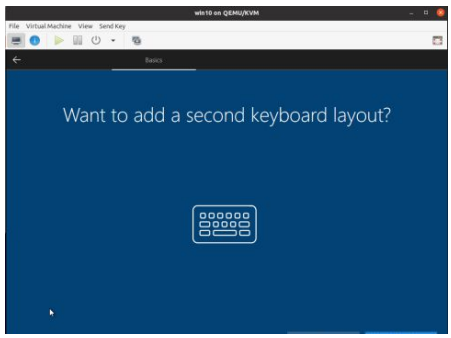
		
Step_9	Click on **Forward** .	
Step_10	Select 4 GB of RAM and 4 CPUs, or the amount needed by the operating system. Click on **Forward** .	
Step_11	Click on **Forward** .	
Step_12	Click on **Finish** . The OS installation process will start.	

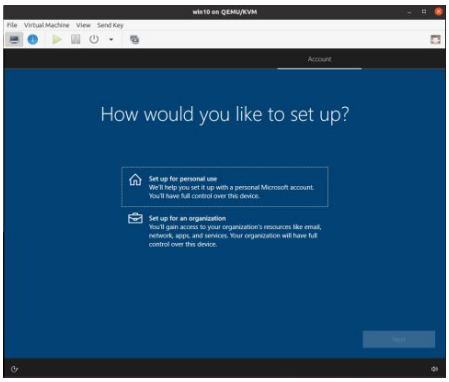
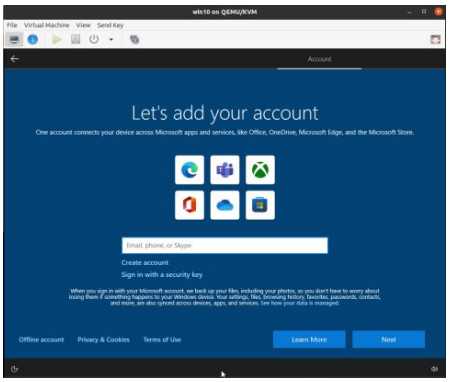
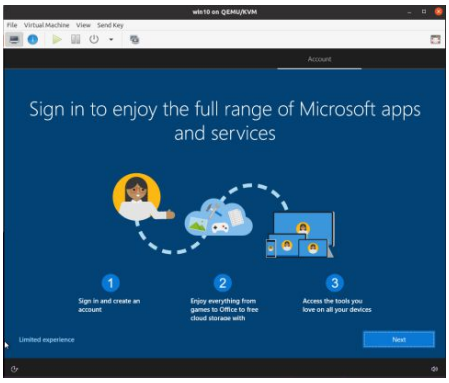
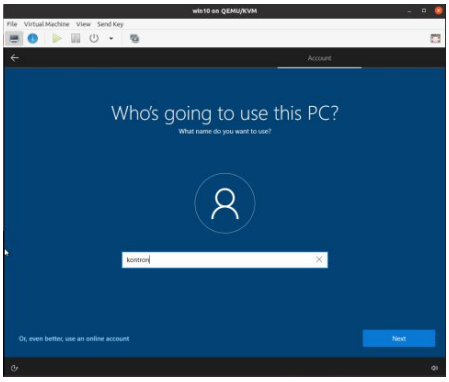
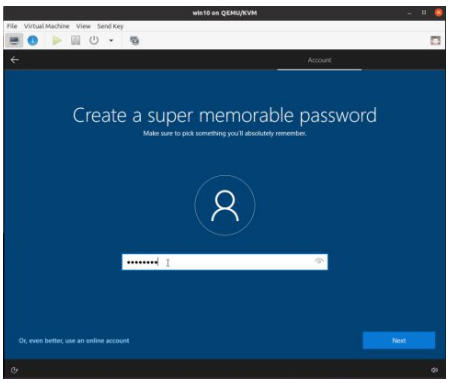
		

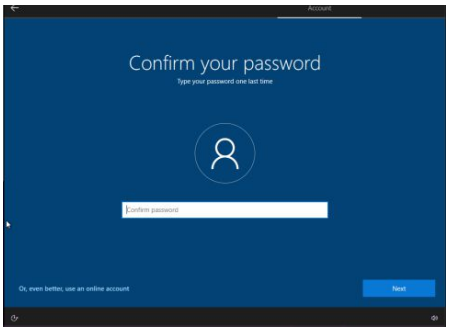
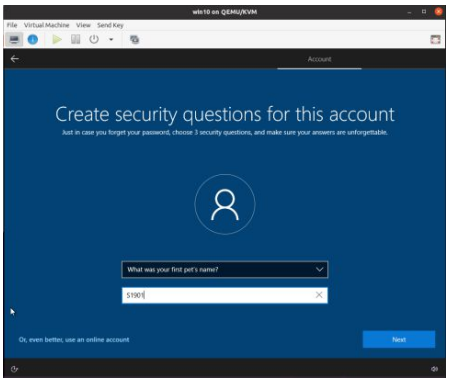
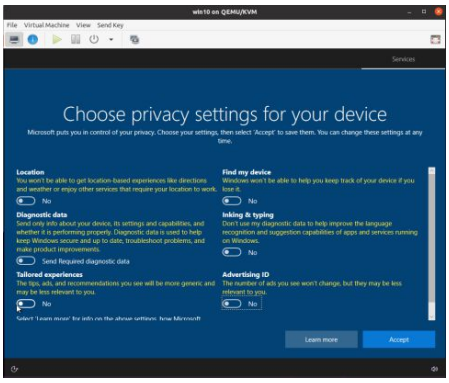
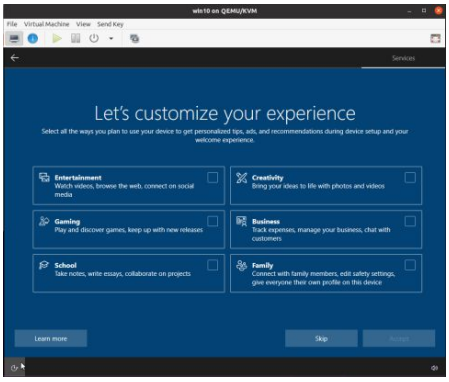
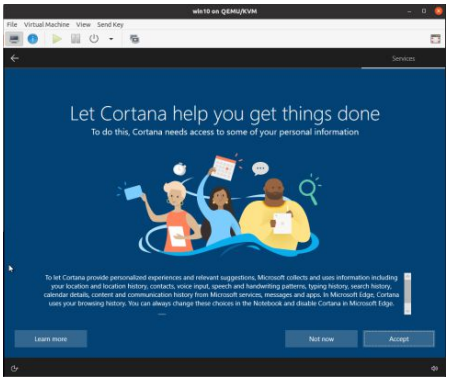
Installing Windows 10 on the virtual machine

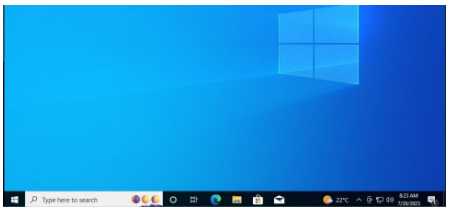
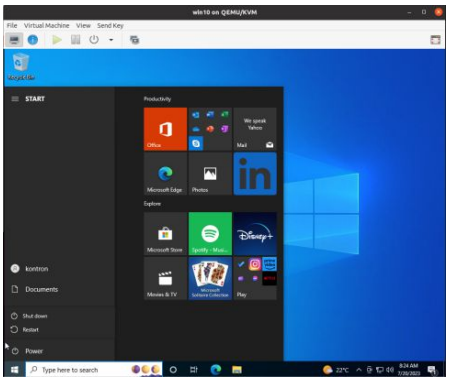
The following procedure shows how to temporarily install Windows without a license. Once the installation is completed, make sure a valid license is entered.

Step_1	Click on **Next** .	
Step_2	Click on **Install now** .	
Step_3	Click on **I don't have a product key** . **NOTE:** This will temporarily install Windows without a license. Once the installation is completed, make sure a valid license is entered.	
Step_4	Select **Windows 10 Pro** . Click on **Next** .	
Step_5	If you agree with the license terms, check the box **I accept the license terms** . Click on **Next** .	

		
Step_6	Click on **Custom: Install Windows Only (advanced)** .	
Step_7	Click on **Next** . **NOTE:** The VM will reboot to complete the installation.	
Step_8	Select your region and click on **Yes** .	
Step_9	Select your keyboard layout and click on **Yes** .	
Step_10	Click on **Skip**. **NOTE:** The VM will reboot to complete the installation.	

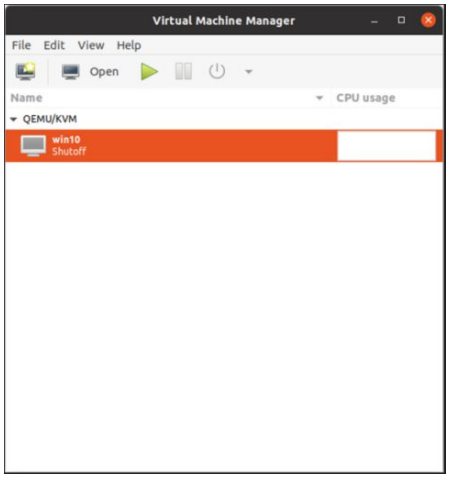
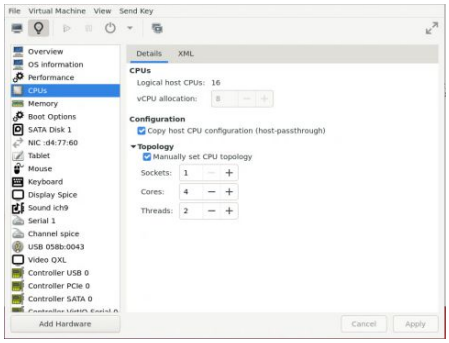
		
Step_11	Select **Set for personal use** or **Set up for an organization** , as per your company policy. For demonstration purposes, **Set for personal use** is used in this procedure. Click on **Next** .	
Step_12	Click on **Offline account** .	
Step_13	Click on **Limited experience** .	
Step_14	Enter a user name: **kontron** . Click on **Next** .	
Step_15	Enter a password: **ready2go** . Click on **Next** .	
Step_16	Re-enter the password: **ready2go** . Click on **Next** .	

		
Step_17	Complete the 3 security questions. Click on **Next** .	
Step_18	Set all privacy settings to **No** . Click on **Accept** .	
Step_19	Click on **Skip** .	
Step_20	Click on **Not now** .	
Step_21	Windows is now installed.	

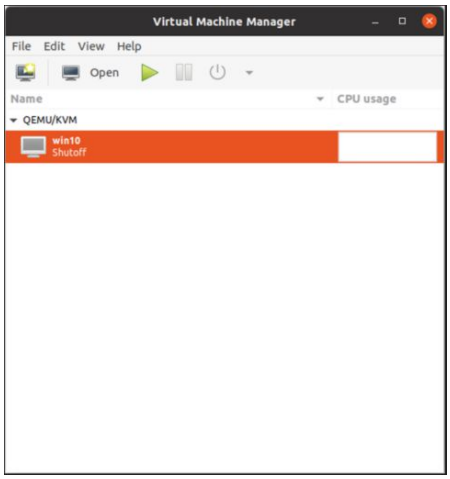
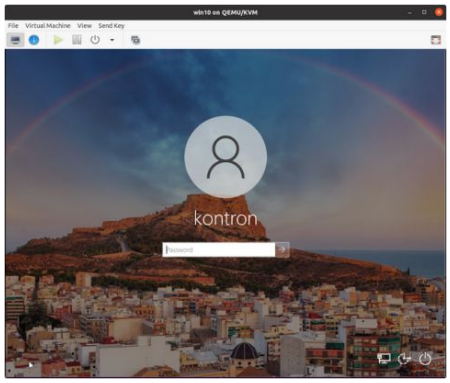
		
Step_22	Shut down the Windows VM.	

Assigning the AURIX MCU USB port to the Windows 10 virtual machine and correcting CPU parameters

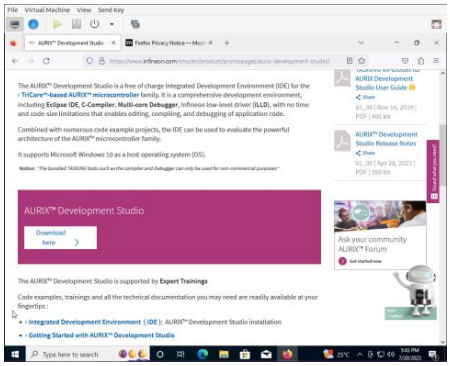
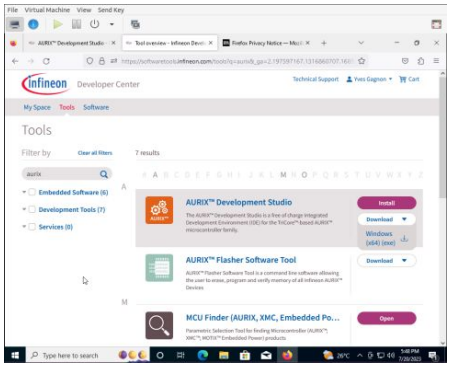
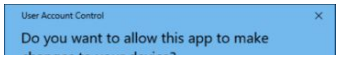
To program the AURIX MCU, its USB port must be assigned to the virtual machine.

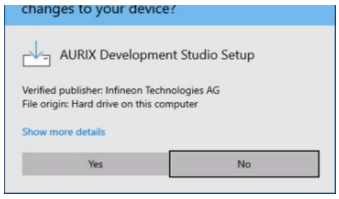
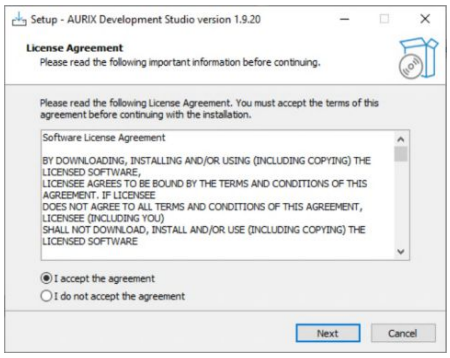
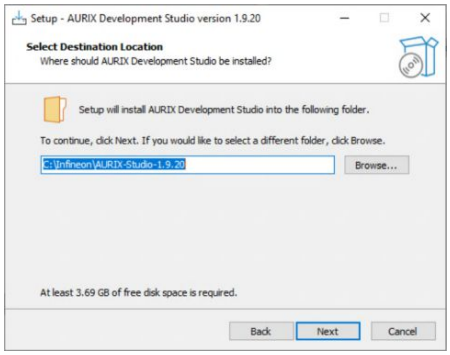
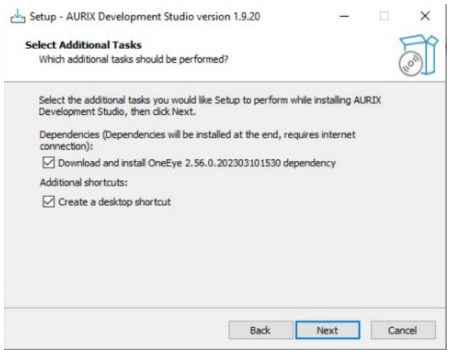
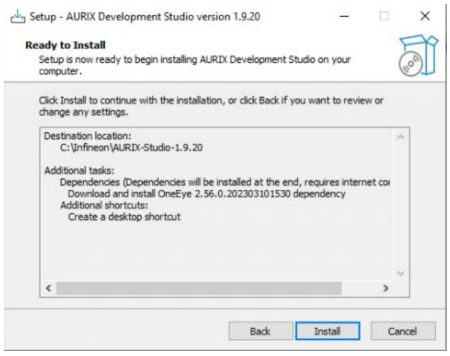
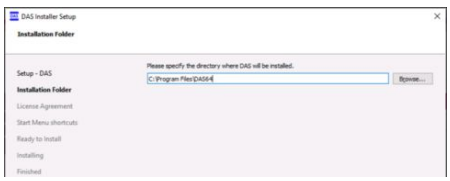
Step_1	From the VM desktop, click on the Virtual Machine Manager icon.	
Step_2	Select the VM to edit (in the image, only one VM is shown). Click on **Open** .	
Step_3	Click on the **i** icon to change the VM configuration.	
Step_4	Click on **Add Hardware** .	
Step_5	Click on **USB Host Device** . Select **001:006 Infineon Technologies DAS JDS Application Kit TC38x V1.0** . Click on **Finish** .	
Step_6	Click on **CPUs** . Select **Manually set CPU topology** . Set **Sockets** to 1. Set **Cores** to 4. Set **Threads** to 2. Click on **Apply** .	

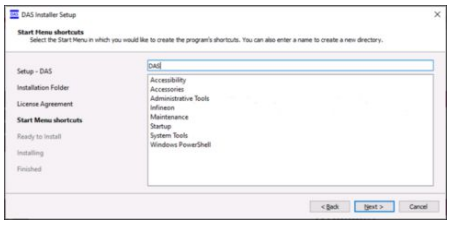
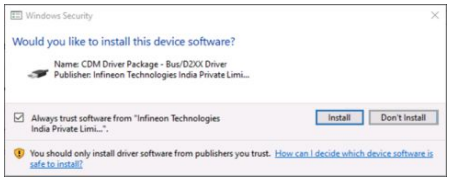
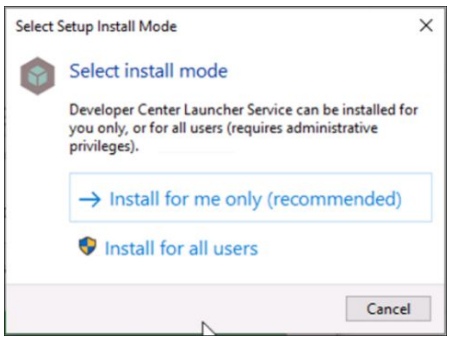
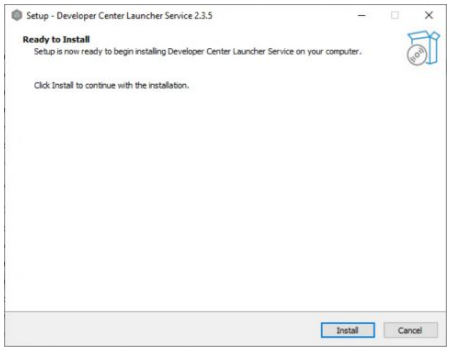
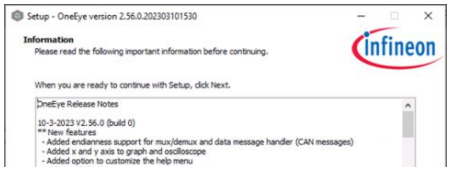
Starting the Windows 10 virtual machine

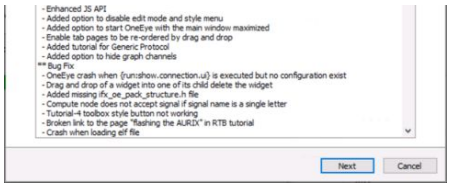
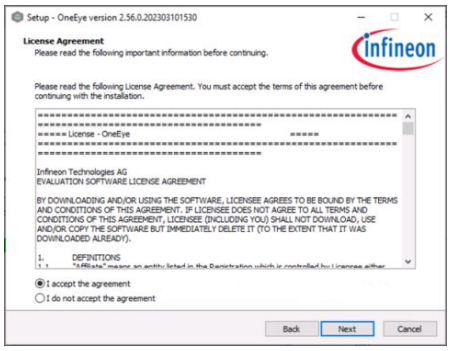
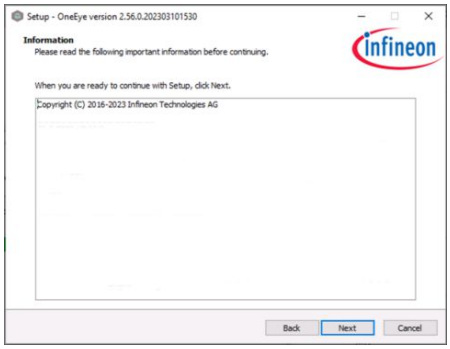
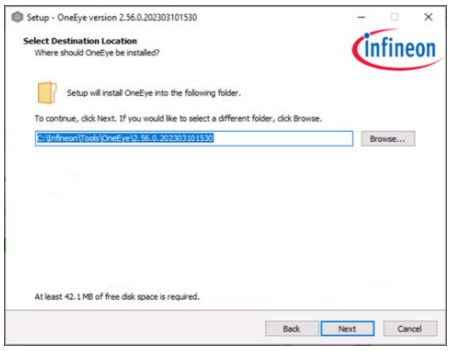
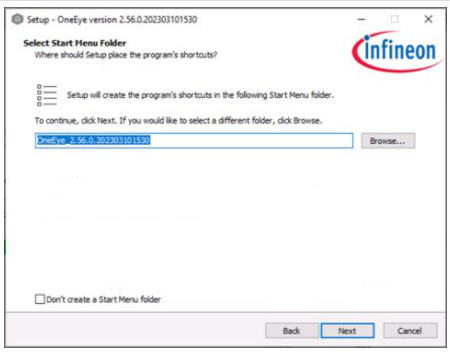
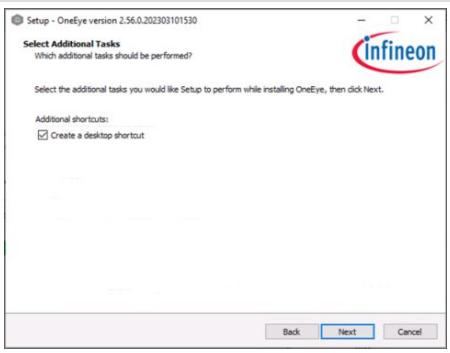
Step_1	From the VM desktop, click on the Virtual Machine Manager icon.	
Step_2	Select the VM to edit (in the image, only one VM is shown). Click on **Open** .	
Step_3	Click on the **VM display** icon to change the VM configuration.	
Step_4	Click on the **Play** icon to change the VM configuration.	
Step_5	Log in.	

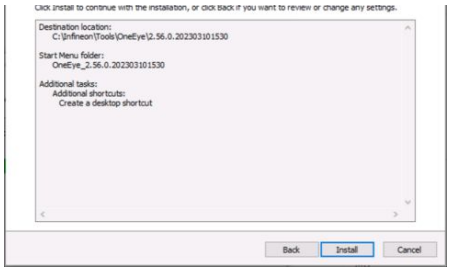
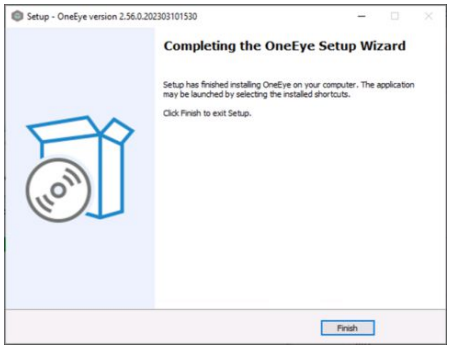
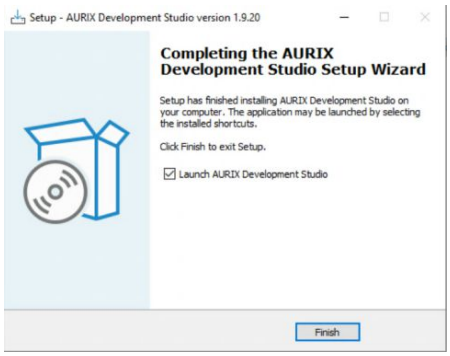
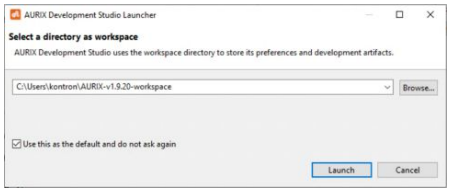
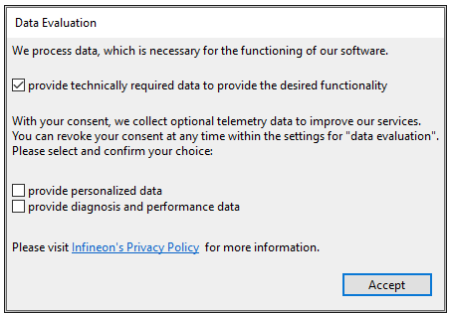
Installing AURIX Development Studio from Infineon

Step_1	Go to the following link (https://www.infineon.com/cms/en/product/promopages/aurix-development-studio/) to download **Development Studio** .	
Step_2	Log in to your Infineon account and click **Download here** .	
Step_3	Click on **Download** and select **Windows (x64) (exe)** .	
Step_4	Run the program downloaded (**Aurixde_1.9.20_Windows_x64**).	
Step_5	Click on **Yes** .	

		
Step_6	If you accept the agreement, click on **I accept the agreement** . Click on **Next** .	
Step_7	Click on **Next** .	
Step_8	Check the box **Create a desktop shortcut** . Click on **Next** .	
Step_9	Click on **Install** .	
Step_10	Click on **Next** .	
Step_11	Click on **Next** .	

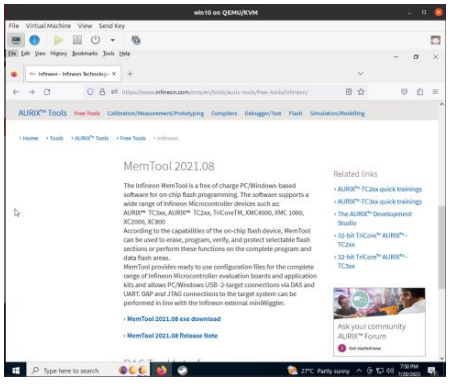
		
Step_12	If you accept the license, check the box **I accept the license** . Click on **Next** .	
Step_13	Click on **Next** .	
Step_14	Click on **Install** .	
Step_15	Click on **Install** .	
Step_16	Click on **Finish**	
Step_17	Click on **Install for me only (recommended)** .	
Step_18	Click on **Install** .	
Step_19	Click on **Next** .	

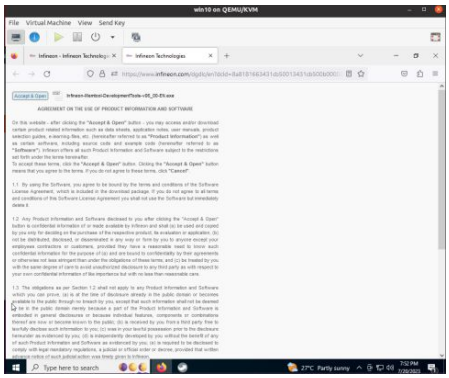
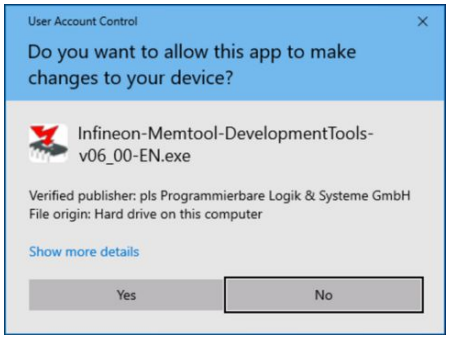
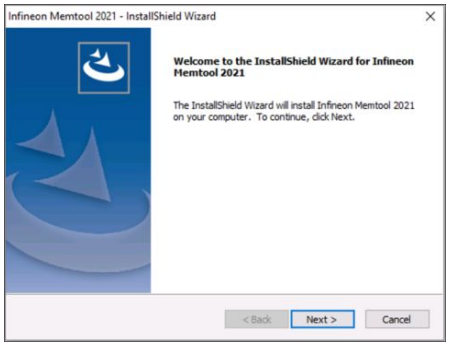
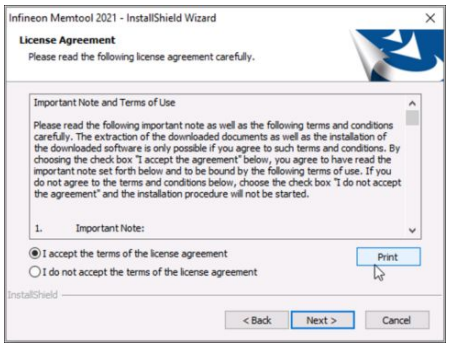
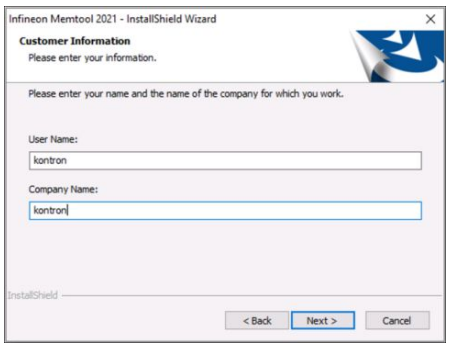
		
Step_20	If you accept the agreement, check the box **I accept the agreement** . Click on **Next** .	
Step_21	Click on **Next** .	
Step_22	Click on **Next** .	
Step_23	Click on **Next** .	
Step_24	Check the box **Create a desktop shortcut** . Click on **Next** .	
Step_25	Click on **Install** .	

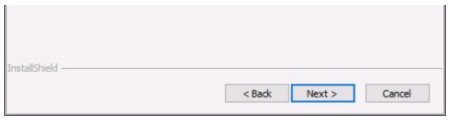
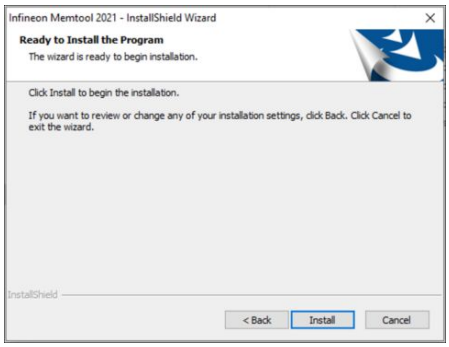
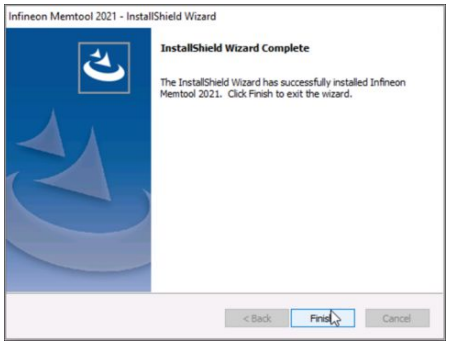
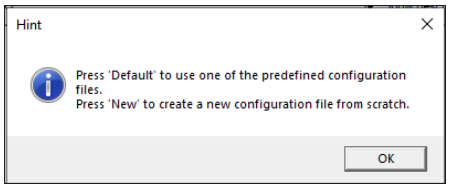
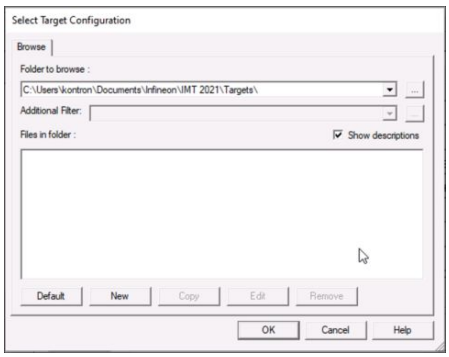
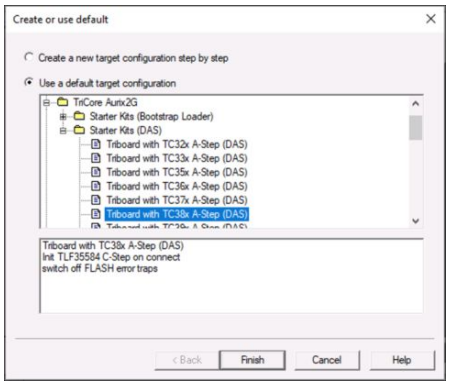
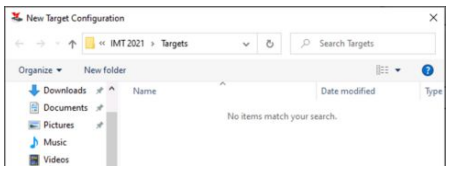
		
Step_26	Click on **Finish** .	
Step_27	Click on **Finish** .	
Step_28	If needed, check the box **Use this as the default and do not ask again** . Click on **Launch** .	
Step_29	If you accept, click on **Accept** .	

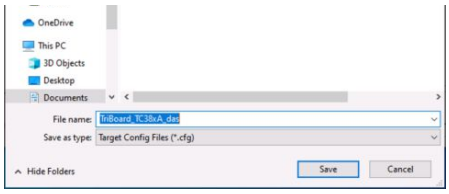
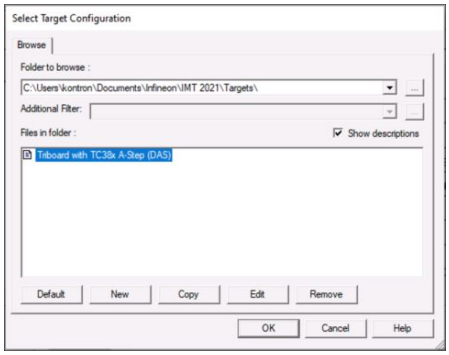
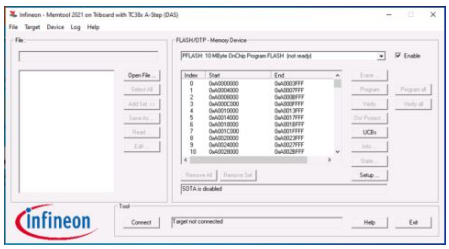
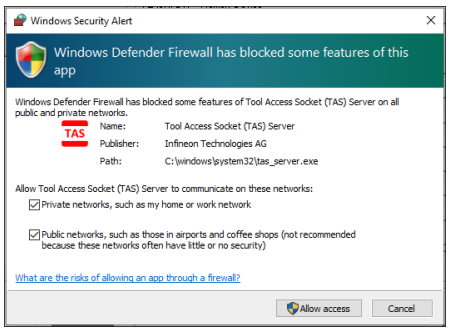
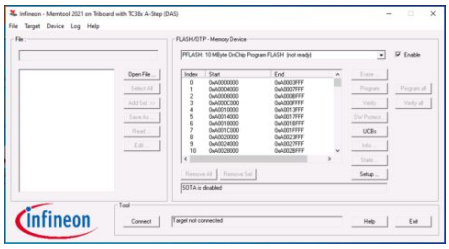
Installing MemTool from Infineon

To program the Aurix MCU, the Infineon MemTool programmer will be used with the DAS method using the UART port.

Step_1	Go to the following link (https://www.infineon.com/cms/en/tools/aurix-tools/free-tools/infineon/) to download **MemTool** .	
Step_2	Access the MemTool section and click on **MemTool 2021.08.exe download** .	

Step_3	If you accept the agreement, click on **Accept & Open** .	
Step_4	Run the **Infineon-MemTool-DevelopmentTools-v06_00-EN** program.	
Step_5	Click on **Yes** .	
Step_6	Click on **Next** .	
Step_7	If you accept the term of the license agreement, click on **Next** .	
Step_8	Enter your company name in the **Company Name** field. Click on **Next** .	
Step_9	Click on **Next** .	

		
Step_10	Click on **Install** .	
Step_11	Click on **Finish** .	
Step_12	From the VM desktop, double click on the **Infineon Memtool** icon.	
Step_13	Click on **OK** .	
Step_14	Click on **Default** .	
Step_15	Select the target that match you board and click on **Finish** . for TC387 use **Triboard with TC38x A-Step (DAS)** for TC397 use **Triboard with TC39x B-Step (DAS)**	
Step_16	Click on **Save** .	

		
Step_17	Click on **OK** .	
Step_18	Click on **Connect** .	
Step_19	Check the boxes that apply to your network policies. Typically, we check **Private networks, such as my home or work network** . Click on **Allow access** .	
Step_20	Click on **Exit** .	

Compiling the AURIX MCU demo code

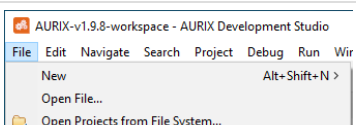
NOTE: This procedure applies for custom code programmed by clients.
The demo code provided by Kontron includes 2 versions:

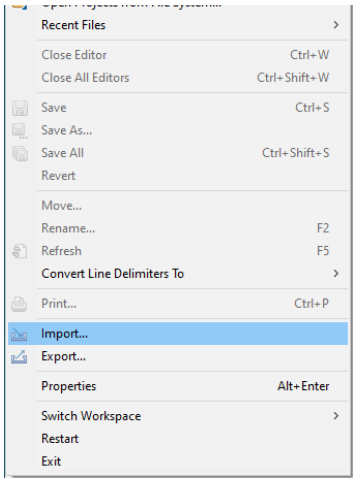
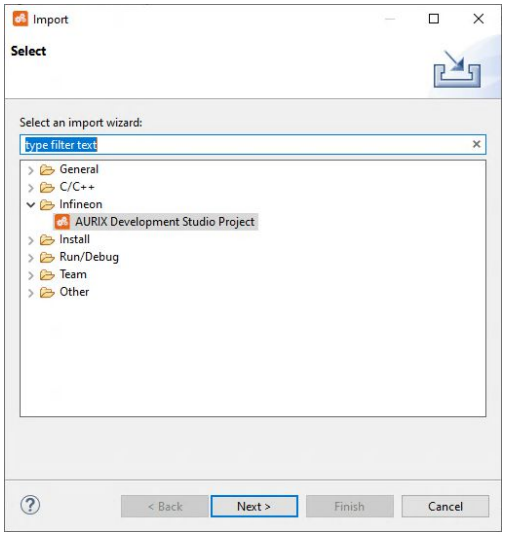
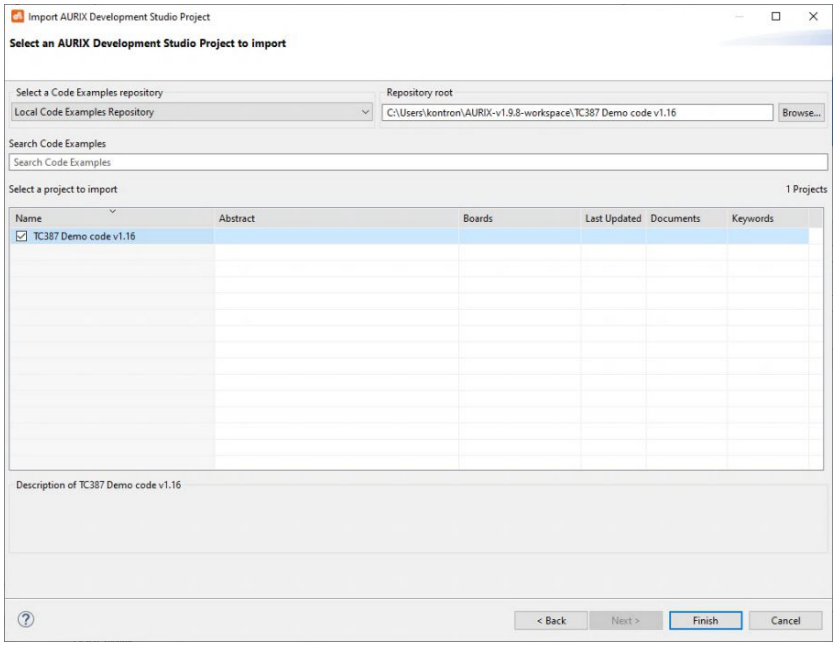
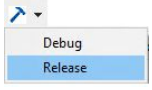
- Release – version that is tested and ready to use as described in the architecture
- Debug – version including experimental functionalities being tested by Kontron and not ready for rollout

Prerequisite

1	The demo code **.zip** archive provided by Kontron must be accessible from the AURIX Development Studio.

Procedure

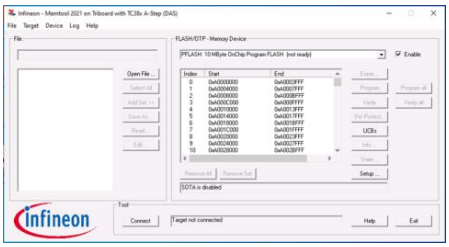
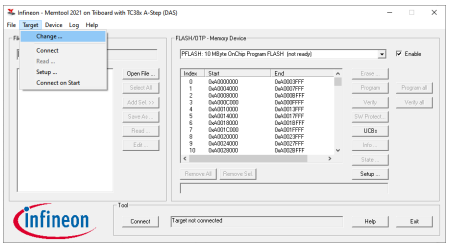
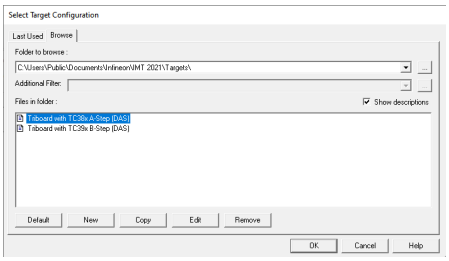
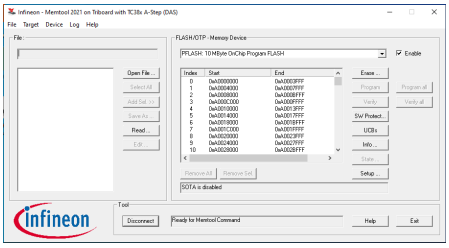
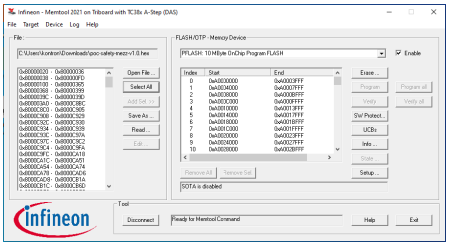
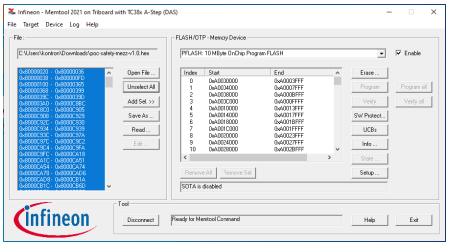
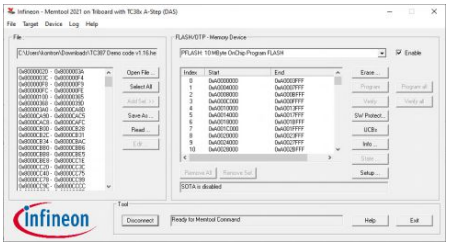
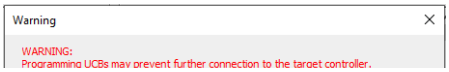
Step_1	Open AURIX Development Studio.	
Step_2	Select **File** and then **Import** .	

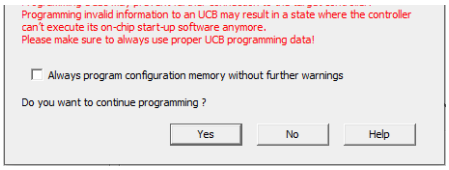
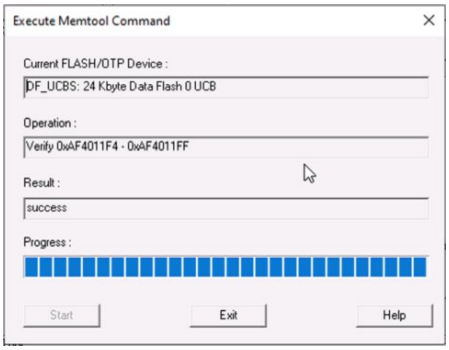
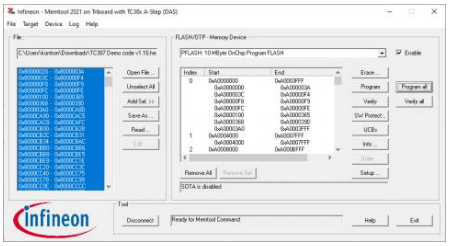
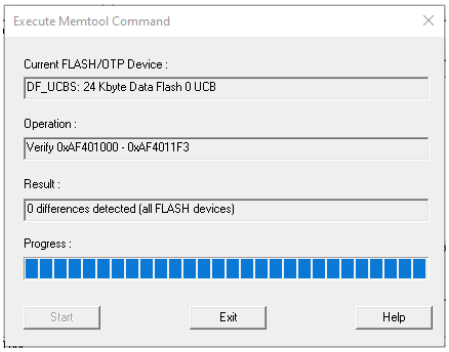
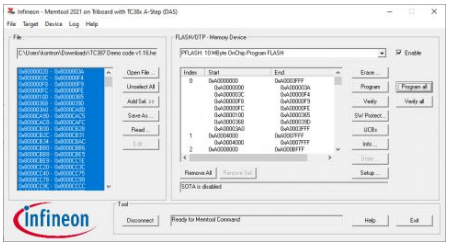
		
Step_3	Click on **AURIX Development Studio Project** and click on **Next**.	
Step_4	Click on **Browse** and select the folder in which the .zip archive was decompressed. Check the box beside the project name and click on **Finish** .	
Step_5	From the top menu bar, click on the arrow next to the icon **Build Active Project** and select **Release** to compile the code.	
Step_6	The .hex file generated will be available from the following folder: C:\Users\kontron\AURIX-v1.9.8-workspace\TC387 Demo code v1.16\Release\TC387 Demo code v1.16.hex. This is the file to download to the CAN bus mezzanine with the AURIX safety MCU.	

Executing the demo code from the AURIX MCU

Downloading the code compiled to the AURIX MCU

Step_1	From the Linux CLI, run the aurix-pre-prog.sh script. This will disable the watchdog timer of the Multi Voltage Safety Micro Processor Supply

	(TLF35584) installed on the CAN bus mezzanine with the AURIX safety MCU and perform a reset. LocalServer_OSPrompt:~# **sudo aurix-pre-prog.sh**	
Step_2	From the VM desktop, double click on the **Infineon Memtool** icon.	
Step_3	Click on **Connect** .	
Step_4	Confirm the AURIX target is the right one. From the **Target** tab, click on **Change...**	
Step_5	Select the proper AURIX target and click on **OK** . The possible targets are: • TC387: Select Triboard with TC38x A-Step (DAS) • TC397: Select Triboard with TC39x B-Step (DAS)	
Step_6	Click on **Connect** and confirm that **Ready for MemTool Command** is displayed in the **Tool** status text box.	
Step_7	Click on **Open File** . Select the appropriate .hex file and click on **Open** .	
Step_8	Click on **Select All** .	
Step_9	Click on **Add Sel >>** and click on **Program all** .	
Step_10	Click on **Yes** .	

		
Step_11	Once **success** is displayed in the **Result** text box, click on **Exit** .	
Step_12	Click on **Verify all** .	
Step_13	Message **0 differences detected (all FLASH devices)** should be displayed in the **Result** text box. Once it is, click on **Exit** .	
Step_14	Click on **Exit** .	
Step_15	From the Linux CLI, run the aurix-post-prog.sh script. This will re-enable the watchdog timer of the Multi Voltage Safety Micro Processor Supply (TLF35584) installed on the CAN bus mezzanine with the AURIX safety MCU and perform a reset. LocalServer_OSPrompt:~# **sudo aurix-post-prog.sh**	

Validating the demo code installation

Step_1	Open the OS CLI and connect via minicom. LocalServer_OSPrompt:~# **resize ; sudo minicom -w -D /dev/ttyS1**	[redacted]
Step_2	Open the shell to confirm installation. The version number will be displayed. Shell> **status** **NOTE:** Version number should be 1.16.	[redacted]

Configuring

Configuring the muxing between the GNSS and the AURIX MCU

Table of contents

Prerequisites

1	An OS is installed.
2	All relevant BSP components are installed.
3	The revision of the carrier board is B and greater.

Relevant sections:
Installing the board support package
Accessing the operating system of a server

The carrier board of the platform includes a MUX that enables a selection between the following components:

- USB port of the AURIX MCU (default)
- USB port of the GNSS

Redirection to the GNSS is not permanent. It will be lost upon reboot.

Muxing from the AURIX MCU to the GNSS

Step_1	Access the OS using the preferred method.	
Step_2	Confirm chip name for redirection of the USB port shared between the AURIX MCU and the GNSS. LocalServer_OSPrompt:~# **echo "Mux device name is $(sudo gpiodetect | grep "0077" |cut -d [-f 2 | cut -d] -f 1)"** **NOTE:** In this case, the device name is pca9539-1-0077.	
Step_3	Redirect communication from the AURIX MCU to the GNSS. LocalServer_OSPrompt:~# **sudo gpioset pca9539-1-0077 2=1**	
Step_4	Confirm the GNSS is detected. LocalServer_OSPrompt:~# **lsusb**	

Muxing from the GNSS to the AURIX MCU

Step_1	Access the OS using the preferred method.	
Step_2	Confirm chip name for redirection of the USB port shared between the AURIX MCU and the GNSS. LocalServer_OSPrompt:~# **echo "Mux device name is $(sudo gpiodetect | grep "0077" |cut -d [-f 2 | cut -d] -f 1)"** **NOTE:** In this case, the device name is pca9539-1-0077.	
Step_3	Redirect communication from the GNSS to the AURIX MCU. LocalServer_OSPrompt:~# **sudo gpioset pca9539-1-0077 2=0**	
Step_4	Confirm the AURIX MCU is detected. LocalServer_OSPrompt:~# **lsusb**	

Configuring the GNSS

Table of contents

The GNSS chip is a NEO-M9N.

Configuring the GNSS for network switch synchronization

Prerequisites

1	An OS is installed.
2	All relevant BSP components are installed.
3	Muxing is set to the GNSS.

Relevant sections:
Installing the board support package
Accessing the operating system of a server
Configuring the muxing between the GNSS and the AURIX MCU

Calculating the cable antenna delay

Configuring compensation of the antenna cable delay is recommended to get highly precise synchronization.

Item	Description	Default value	Value in this platform
CFG-TP-ANT_CABLEDELAY	Antenna cable delay	10 ns	User-defined

Example:
Here is an example with a good-quality 100-meter cable and a velocity factor of 0.86 (86% of the speed of light).

$$\frac{100m}{299792458\ \frac{m}{s} \times 0.86\ velocity\ factor} = 387.8\ ns$$

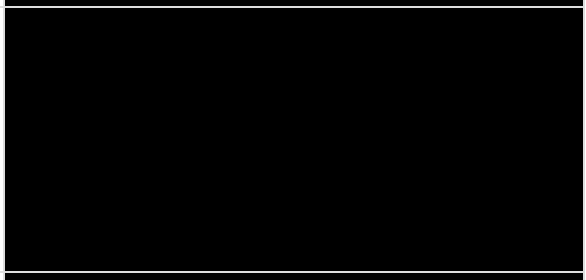
This 100-meter cable would add approximately 388 ns to the timing between it's ends.

Procedure

The GNSS receiver generates a PPS signal used by the switch NOS to determine when each second starts. The GNSS receiver provides the time of day through the NMEA data sent over the serial connection between the switch NOS and the GNSS receiver. The NMEA data is also available on the COMe through /dev/ttyACM0.

To change the GNSS receiver (NEO-M9N) settings, use **ubxtool** from the **gpsd** software package for Linux running on the integrated server.

> Version 3.20 (or more recent) of the **gpsd** software package is required. Please refer to https://gpsd.gitlab.io/gpsd/index.html for more information.

Step_1	Install the gpsd clients. LocalServer_OSPrompt:~# **sudo apt-get install -y gpsd-clients**	
Step_2	Configure the serial connection at a baud rate of 115200. LocalServer_OSPrompt:~# **sudo ubxtool -f /dev/ttyACM0 -P 32 -S 115200**	
Step_3	Configure the antenna cable delay. In this example, the value will be set to 10 ns. LocalServer_OSPrompt:~# **sudo ubxtool -f /dev/ttyACM0 -P32 -z CFG-TP-ANT_CABLEDELAY,[CABLE_DELAY]**	
Step_4	Save the configuratio n . LocalServer_OSPrompt:~# **sudo ubxtool -f /dev/ttyACM0 -P32 -p SAVE**	

Reading NMEA values

Prerequisites

1	An OS is installed.
2	All relevant BSP components are installed.
3	Muxing is set to the GNSS.

Relevant sections:

Installing the board support package
Accessing the operating system of a server
Configuring the muxing between the GNSS and the AURIX MCU

Procedure

Step_1	Access the OS using the preferred method.	
Step_2	Read the NMEA value directly from the USB device. LocalServer_OSPrompt:~# **sudo cat /dev/ttyACM0**	

Configuring date and time

Table of contents

Date and time can be configured in three of the platform's components: the COMe, the NOS and the AURIX MCU.
For the COMe, use the standard Linux procedure.
For the NOS, use the instruction below while considering the following:

- On the network switch, date and time needs to be configured on 2 distinct components:
  - Network operating system (NOS)
  - PTP
- Time zones need to be configured. Configuration will be reflected in both components (NOS and PTP).

For the AURIX MCU, use the instructions below.

Configuring the OS date and time

Access the OS. Refer to Accessing the operating system of a server for access instructions.

Step_1	Proceed with configuration as recommended in the OS documentation.

Typical commands in Linux

Setting the time	**date +%T -s "11:14:00"**
Setting the date	**date +%Y%m%d -s "20120418"**
Setting the date and time with NTP	See information on Chrony
Setting the date and time with PTP	See information on linuxptp

Configuring the switch NOS date and time based on the NTP

NOS date and time is used exclusively to obtain exact information for logs. Configuration can be done using:

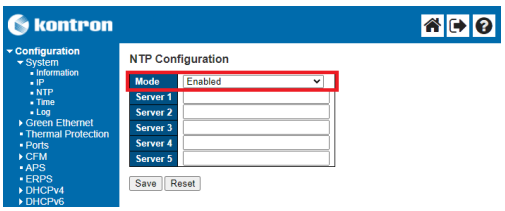
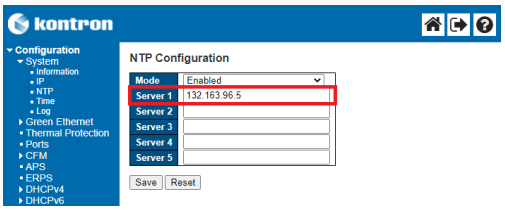
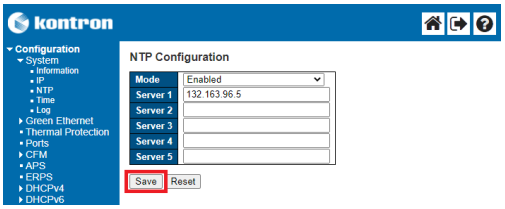
- The Web UI
- The CLI

	It is not possible to manually set the date and time in the switch NOS. NTP must be used as a time source. If no NTP server is present on the network, Chrony can be installed on the COMe.

	**Changes to the switch NOS configuration are not persistent** after rebooting the switch NOS. To preserve configurations, the current configuration needs to be saved to startup-config. From the switch NOS Web UI: • Select **Maintenance** , **Configuration** and then **Save startup-config** . Click on **Save Configuration** to confirm the change. From the switch NOS CLI: • LocalSwitchNOS_OSPrompt:~(config-if)# **end** • LocalSwitchNOS_OSPrompt:~# **copy running-config startup-config**

Configuring the switch NOS date and time based on the NTP using the Web UI

Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu, select **Configuration, System,** and then **NTP** .	
Step_2	Enable the NTP service by changing the value from the **Mode** dropdown menu to **Enabled** .	
Step_3	Enter the NTP server's address or hostname. **NOTE:** To enter a server hostname, a DNS service must be configured.	
Step_4	Repeat the previous step to add multiple NTP servers if needed.	
Step_5	Click on the **Save** button.	
Step_6	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring the switch NOS date and time based on the NTP using the CLI

Access the switch NOS CLI using one of the SSH methods described in section Accessing the switch network operating system .

Step_1	Enter configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	`# configure terminal`
Step_2	Enable the NTP. LocalSwitchNOS_OSPrompt:~(config)# **ntp** **NOTE:** To disable NTP, use **no ntp** .	`(config)# ntp`

Step_3	Configure the NTP server.
	LocalSwitchNOS_OSPrompt:~(config) # **ntp server [SERVER_ID] ip-address [IP_ADDRESS_OR_HOSTNAME]**
	NOTE: To enter a server hostname, a DNS service must be configured.
		`(config)# ntp server 1 ip-address 132.163.96.5`
		OR
		`(config)# ntp server 1 ip-address pool.ntp.org`

Step_4	Exit configuration mode. LocalSwitchNOS_OSPrompt:~(config)# **exit**	`(config)# exit`

Step_5	Verify the NTP configuration by displaying the list of NTP servers.
	LocalSwitchNOS_OSPrompt:~ # **show ntp status**
		`# show ntp status`
		`NTP Mode : enabled`
		`Idx  Server IP host address (a.b.c.d) or a host name string`
		`---  ----------------------------------------------------------`
		`1  132.163.96.5`
		`2`
		`3`
		`4`
		`5`

Step_6	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring the PTP date and time

The procedure to configure the PTP date and time is different if a GNSS is operational or not:

- If the switch NOS is configured as a grandmaster clock, use the GNSS a time source.
- If the switch NOS uses an external grandmaster clock, the local GNSS is not needed .

Relevant sections:
Configuring the GNSS
Configuring synchronization

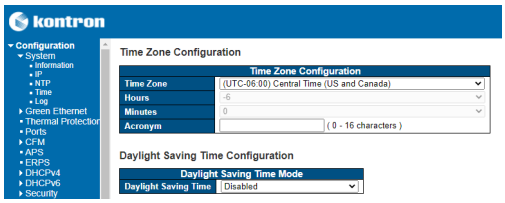
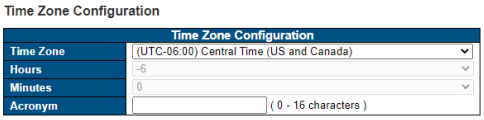
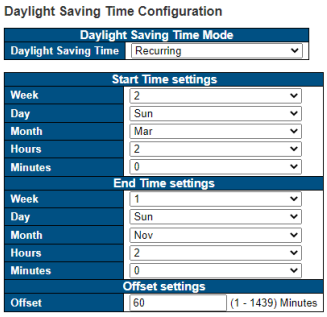
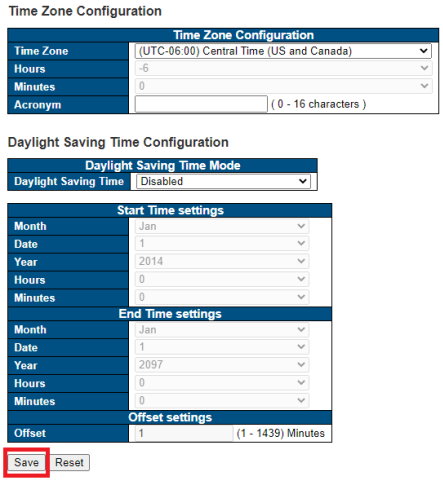
Configuring the time zone

Configuration can be done using:

- The Web UI
- The CLI

Configuring the time zone using the Web UI

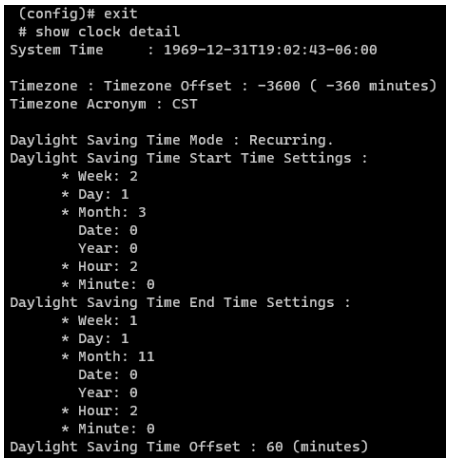
Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu, select **Configuration, System** and then **Time** .	
Step_2	Configure the time zone by selecting it from the **Time Zone** dropdown menu.	
Step_3	Configure the **Daylight Saving Time** .	
Step_4	Click on **Save** .	
Step_5	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring the time zone using the CLI

Access the switch NOS CLI using one of the SSH methods described in section Accessing the switch network operating system .

Step_1	Enter configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**
	`# configure terminal`
Step_2	Manually set the hour and minute offsets. LocalSwitchNOS_OSPrompt:~(config)# **clock timezone** [TIME_ZONE_ACRONYM] [HOUR_OFFSET] [MINUTE_OFFSET]
	`(config)# clock timezone CST -6 0`
Step_3	Configure the daylight saving time. LocalSwitchNOS_OSPrompt:~(config)# **clock summer-time** [TIME_ZONE_ACRONYM] **date** [STARTING_MONTH] [STARTING_DAY] [STARTING_YEAR] [STARTING_HH:MM] [ENDING_MONTH] [ENDING_DAY] [ENDING_YEAR] [ENDING_HH:MM] [OFFSET] **NOTE:** This command sets the parameters for one year only. They will have to be reprogrammed the following year. or LocalSwitchNOS_OSPrompt:~(config)# **clock summer-time** [TIME_ZONE_ACRONYM] **recurring** [STARTING_WEEK] [STARTING_MONTH] [STARTING_DAY 1=Sunday] [STARTING_HH:MM] [ENDING_WEEK] [ENDING_MONTH] [ENDING_DAY] [ENDING_HH:MM] [MINUTE_OFFSET] **NOTE:** This command sets the parameters for every year. No reprogramming needed.
	`clock summer-time CDT recurring 2 1 3 2:00 1 1 11 2:00 60`

Step_4	Verify the time zone configuration. LocalSwitchNOS_OSPrompt:~(config)# **exit** LocalSwitchNOS_OSPrompt:~# **show clock detail**
	
Step_5	(Optional) To make the change persistent, save running-config to startup-config.

Configuring the AURIX MCU date and time

This section provides the information required to program the time source of the AURIX MCU and also the repository to store it.

The AURIX MCU is connected to the carrier board GNSS (NEO-M9N-00B) with an I2C bus. This bus is available through the following pins: P11_13 (SDA) and P11_14 (SCL). (For future use.)
The AURIX MCU is equipped with a battery-backed I2C real-time clock/calendar (MCP79411). The I2C bus is available through the following pins: P15_5 (SDA) and P15_4 (SCL).

Configuring management access protocols

Table of contents

	**Changes to the switch NOS configuration are not persistent** after rebooting the switch NOS. To preserve configurations, the current configuration needs to be saved to startup-config. From the switch NOS Web UI: • Select **Maintenance** , **Configuration** and then **Save startup-config** . Click on **Save Configuration** to confirm the change. From the switch NOS CLI: • LocalSwitchNOS_OSPrompt:~(config-if)# **end** • LocalSwitchNOS_OSPrompt:~# **copy running-config startup-config**

Configuring the NOS Web UI

The Web server can be accessed using two protocols: HTTP and HTTPS. They are independent and both can be used simultaneously. The network switch can therefore operate in any of the following 3 modes:

- **HTTP only** – All information is transferred in clear text (even passwords). **Not secure!** Communications are on Port 80.
- **HTTPS only** – All information is transferred in encrypted packets. **Communication is secure** . HTTP requests are automatically translated as HTTPS requests. Communications are on Port 443. **A certificate is required for HTTPS.**
- **HTTP and HTTPS** – Users can use any of the 2 protocols. **This is the default state, but a certificate is required for HTTPS.**

For the secure HTTPS protocol to work, a certificate needs to be installed . See the Certificates sections below (Web UI and CLI).

The http and/or https services that enable communication with the switch NOS Web UI can be disabled and enabled and various parameters can be configured:

- Using the CLI
- Using the switch NOS Web UI

Configuring HTTPS support using the CLI

Displaying HTTPS states

Refer to Accessing the switch network operating system for access instructions.
To know the states of the various secure HTTP variables, two command can be used: **show ip http** (in normal mode) or **do show ip http** (in configuration mode).

Step_1	LocalSwitchNOS_OSPrompt:~# **show ip http**	NOS00A0A5E01CF4# show ip http Switch secure HTTP web server is disabled Switch secure HTTP web redirection is disabled Switch secure HTTP certificate is not presented

Field	Description	Value
Switch secure HTTP web server is	Shows the state of the **Switch secure HTTP web server** . When the state is **Enabled** , secure HTTPS communications trough port 443 are available. **NOTE** : For the state to be **Enabled** , a certificate **must** be present.	Enabled Disabled
Switch secure HTTP web redirection is	When the state is **Enabled** , HTTP communications are redirected to the **Switch secure HTTP web server** . This means the HTTP web server is no longer used. **NOTE** : For the state to be **Enabled** , the **Switch secure HTTP web server must** be set to **Enabled** beforehand.	Enabled Disabled
Switch secure HTTP certificate is	Shows if a certificate is installed in the system. Presented means that a certificate is installed and can be used for HTTPS encryption.	Presented Not presented

Disabling the http service of the switch NOS Web UI

Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

Step_1	Access the switch NOS CLI using the serial console from the integrated server.	
Step_2	Enter configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	[redacted]
Step_3	Disable the http service that allows communication with the switch NOS Web UI. LocalSwitchNOS_OSPrompt:~(config)# **no aaa authentication login http**	[redacted]
Step_4	Exit configuration mode. LocalSwitchNOS_OSPrompt:~(config)# **exit**	[redacted]
Step_5	(Optional) To make the change persistent, save running-config to startup-config.	

Enabling the http service of the switch NOS Web UI

Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

Step_1	Access the switch NOS CLI using the serial console from the integrated server.	
Step_2	Enter configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	[redacted]
Step_3	Enable the http service that allows communication with the switch NOS Web UI. LocalSwitchNOS_OSPrompt:~(config)# **aaa authentication login http local**	[redacted]
Step_4	Exit configuration mode. LocalSwitchNOS_OSPrompt:~(config)# **exit**	[redacted]
Step_5	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring certificates using the CLI

Refer to Accessing the switch network operating system for access instructions.
Any certificate will allow the web server to encrypt the information transferred.
Only certificates obtained from a trusted Certificate Authority (CA) can guarantee authenticity trough a chain of thrust. CA User Certificate Platform certificate.
There are 3 ways to insert a certificate:

- **Generate a self-signed certificate** – this should only be a temporary solution. It is secure, but not safe. Data will be encrypted, but cannot be trusted.
- **Upload a certificate from a URL**
- **Upload a certificate from a user file system**

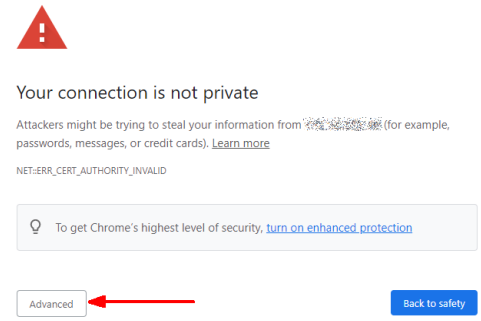
Displaying available commands

Step_1	Go in configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	``` NOS00A0A5E01CF4# configure terminal NOS00A0A5E01CF4(config)# ip http secure-certificate ? delete Delete the current certificate generate Generate a new self-signed RSA certificate upload Upload a certificate PEM file ```
Step_2	Show available commands. LocalSwitchNOS_OSPrompt:~(config)# **ip http secure-certificate ?**	

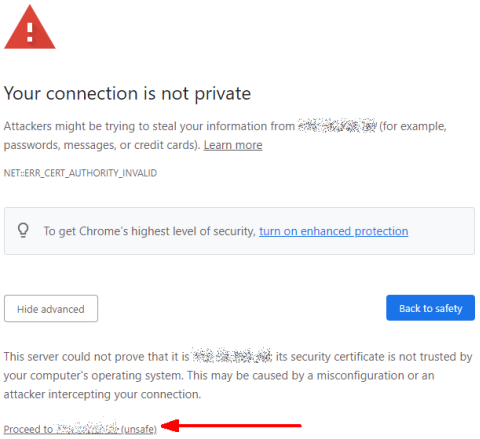
Generating a self-signed certificate

A self-signed certificate, which should only be used as a temporary solution, allows communication to be encrypted, but cannot certify that the server is

really what it claims to be.
NOTE : The self-signed certificate will be valid for a fixed time period (e.g. November 30th 2021 at 00:00:01 up to November 30th 2031 at 23:59:59).
If a self-signed certificate is used, the Web browser will display a warning message before you can access the page. If this is the case, click on **Advanced**.



Then click on the **Proceed to [IP_ADDRESS] (unsafe)** link.



From the network switch CLI:

Step_1	Go in configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	`NOS00A0A5E01CF4# configure terminal` `NOS00A0A5E01CF4(config)# ip http secure-certificate generate`
Step_2	Generate a certificate. LocalSwitchNOS_OSPrompt:~(config)# **ip http secure-certificate generate**	

Step_3	Ensure the certificate and HTTP web server are correctly configured.
	LocalSwitchNOS_OSPrompt:~# **do show ip http**
	NOTE : Certificate generation can take a few seconds. If it is still generating when checking the status, the CLI will indicate that it is generating...
		`NOS00A0A5E01CF4(config)# do show ip http`
		`Switch secure HTTP web server is disabled`
		`Switch secure HTTP web redirection is disabled`
		`Switch secure HTTP certificate is generating ...`

Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Uploading a certificate from a URL

Step_1	Enter the configuration terminal. LocalSwitchNOS_OSPrompt:~# **configure terminal**
Step_2	Upload the certificate. LocalSwitchNOS_OSPrompt:~(config)# **ip http secure-certificate upload [PROTOCOL] ://[USERNAME]: [PASSWORD]@ [HOST_IP_ADDRESS]: [PORT][FILE_PATH]**
	`NOS00A0A5E01CF4# configure terminal` `NOS00A0A5E01CF4(config)# ip http secure-certificate upload tftp://10.10.10.10/certificate.pem`
Step_3	Ensure the certificate and HTTP web server are correctly configured. LocalSwitchNOS_OSPrompt:~# **do show ip http** **NOTE** : Certificate generation can take a few seconds. If it is still generating when checking the status, the CLI will indicate that it is generating.

	`NOS00A0A5E01CF4(config)# do show ip http`
	`Switch secure HTTP web server is disabled`
	`Switch secure HTTP web redirection is disabled`
	`Switch secure HTTP certificate is generating ...`

Step_4	(Optional) To make the change persistent, save running-config to startup-config.

Deleting an installed certificate

Step_1	Go in configuration mode.
	LocalSwitchNOS_OSPrompt:~# **configure terminal**
		`NOS00A0A5E01CF4# configure terminal`
		`NOS00A0A5E01CF4(config)# ip http secure-certificate delete`
		`NOS00A0A5E01CF4(config)#`

Step_2	LocalSwitchNOS_OSPrompt:~(config)# **ip http secure-certificate delete**	
Step_3	Ensure the certificate and HTTP web server are correctly configured. LocalSwitchNOS_OSPrompt:~# **do show ip http**	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring the interface protocol using the CLI

The interface protocol can be configured as:

- HTTPS (recommended)
- HTTP and HTTPS
- HTTP

Configuring the interface for HTTPS only using the CLI

To configure the interface for HTTPS only, the HTTPS server must be enabled and the redirection must also be enabled. This will disable the HTTP server.

Step_1	Enter the configuration terminal.
	LocalSwitchNOS_OSPrompt:~# **configure terminal**
		`NOS00A0A5E01CF4(config)# ip http secure-server`
		`NOS00A0A5E01CF4(config)# ip http secure-redirect`
		`NOS00A0A5E01CF4(config)# do show ip http`
		`Switch secure HTTP web server is enabled`
		`Switch secure HTTP web redirection is enabled`
		`Switch secure HTTP certificate is presented`

Step_2	Configure the interface for HTTPS. LocalSwitchNOS_OSPrompt:~(config)# **ip http secure-server**	
Step_3	Enable redirection. LocalSwitchNOS_OSPrompt:~(config)# **ip http secure-redirect**	
Step_4	Ensure the certificate and HTTP web server are correctly configured. LocalSwitchNOS_OSPrompt: ~(config) # **do show ip http**	
Step_5	(Optional) To make the change persistent, save running-config to startup-config.	

Configure the interface for HTTP and HTTPS using the CLI

Step_1	Enter the configuration terminal.
	LocalSwitchNOS_OSPrompt:~# **configure terminal**
		`NOS00A0A5E01CF4(config)# ip http secure-server`
		`NOS00A0A5E01CF4(config)# do show ip http`
		`Switch secure HTTP web server is enabled`
		`Switch secure HTTP web redirection is disabled`
		`Switch secure HTTP certificate is presented`

Step_2	LocalSwitchNOS_OSPrompt:~(config)# **ip http secure-server**	
Step_3	Ensure the certificate and HTTP web server are correctly configured. LocalSwitchNOS_OSPrompt: ~(config) # **do show ip http**	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring the interface for HTTP only using the CLI

If the interface is configured for HTTP only, the HTTPS Switch secure HTTP web server will be disabled and so will the Switch secure HTTP web redirection.

Step_1	Enter the configuration terminal.
	LocalSwitchNOS_OSPrompt:~# **configure terminal**
		`NOS00A0A5E01CF4(config)# no ip http secure-server`
		`NOS00A0A5E01CF4(config)# do show ip http`
		`Switch secure HTTP web server is disabled`
		`Switch secure HTTP web redirection is disabled`
		`Switch secure HTTP certificate is presented`

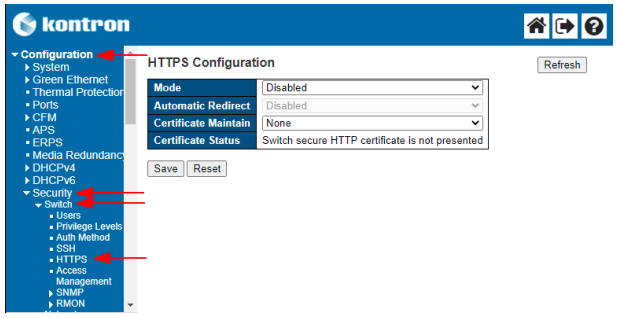
Step_2	LocalSwitchNOS_OSPrompt:~(config)# **no ip http secure-server**	
Step_3	Ensure the certificate and HTTP web server are correctly configured. LocalSwitchNOS_OSPrompt: ~(config) # **do show ip http**	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring HTTPS support using the Web UI

HTTPS configuration page

Refer to Accessing the switch NOS using the switch NOS Web UI for access instructions.
This page is used to configure the HTTPS settings and maintain the current certificate on the switch.

For the secure HTTPS protocol to work, a certificate needs to be installed. As a temporary measure, the switch can create a self-signed certificate, which is secure but cannot be trusted as a long term solution. Users will need to provide their own certificate, delivered from a valid certificate authority.

Step_1	From the left-side menu, select **Configuration** , **Security** , **Switch** and then **HTTPS** .	
Step_2	Select the desired settings for **Mode** , **Automatic Redirect** , **Certificate Maintain** (based on the value chosen, additional fields will be available) and **Certificate Status** . See the table below for an explanation of the values available for each field.	
Step_3	Press on the **Save** button to confirm.	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Values available for fields used for HTTPS configuration

Field	Description	Values
Mode	Sets the HTTPS operation mode.	**Enabled** : HTTPS operation mode is enabled. **Disabled** : HTTPS operation mode is disabled.
Automatic Redirect	Sets the HTTPS redirect operation mode. This setting is required only when **Mode** is set to **Enabled** . When redirection is enabled, the HTTP connection will be redirected to the HTTPS connection automatically. Note that the browser may not allow redirection due to security considerations, unless the switch certificate is trusted by the browser. An HTTPS connection needs to be manually initialized in this case. When the value of this field is set to **Enabled** , the HTTP protocol is effectively disabled.	**Enabled** : HTTPS redirect operation mode is enabled. **Disabled** : HTTPS redirect operation mode is disabled.
Certificate Maintain	Performs certificate maintenance. This setting is operational only when **Mode** is set to **Disabled** .	**None** : Nothing happens. **Delete** : Deletes the current certificate. **Upload** : Uploads a certificate PEM file. **Generate** : Generates a new self-signed RSA certificate.
Certificate Pass Phrase (Available when the **Certificate Maintain** field is set to **Upload** .)	Holds the passphrase protecting the certificate to upload.	
Certificate	Uploads a	**Web Browser** : Upload a certificate via a Web browser.

	Upload (Available when the **Certificate Maintain** field is set to **Upload** .)		certificate PEM file into the switch. The file should contain both the certificate and private key. If the certificate and private key are in two separate files, use the Linux cat command to combine them into a single PEM file: `cat my.cert my.key > my.pem` Note that an RSA certificate is recommended since most newer browser versions have removed support for DSA in certificates (e.g. Firefox v37 and Chrome v39).	**URL** : Upload a certificate via an URL.
		File Upload (Available when the **Certificate Upload** field is set to **Web Browser** .)	Lets users select the file to upload.	
		URL (Available when the **Certificate Upload** field is set to **URL** .)	Holds the URL.	URL format: [PROTOCOL]://[USERNAME]:[PASSWORD]@[HOST_IP_ADDRESS]:[PORT] [FILE_PATH] . The protocols supported are HTTP, HTTPS, TFTP and FTP. For example: • **tftp://10.10.10.10/new_image_path/new_image.dat** • http://username:password@10.10.10.10:80/new_image_path/new_image.dat A valid file name is a text string drawn from alphabet letters (A-Za-z), digits (0-9), dots (.), hyphens (-) and under scores (_). The maximum length is 63 and a hyphen must not be the first character. A file name that only contains '.' is not allowed.
Certificate Status			Displays the current status of the switch certificate.	**Switch secure HTTP certificate is presented :** When a valid certificate is present. **Switch secure HTTP certificate is not presented :** When no valid certificate is present or the certificate has been deleted. **Switch secure HTTP certificate is generating ....:** When the self-signed certificate is being generated (wait 1 minute and then refresh the page for results).

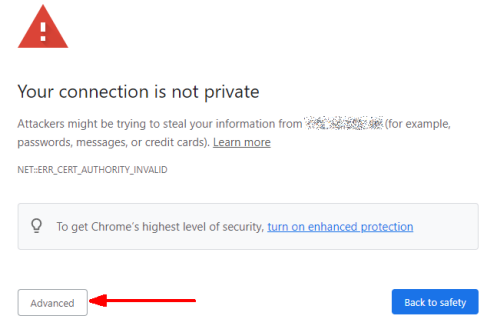
Configuring certificates using the Web UI

Refer to Accessing the switch NOS using the switch NOS Web UI for access instructions.
Any certificate will allow the web server to encrypt the information transferred.
Only certificates obtained from a trusted Certificate Authority (CA) can guarantee authenticity trough a chain of thrust. CA User Certificate Platform certificate.
There are 3 ways to insert a certificate:

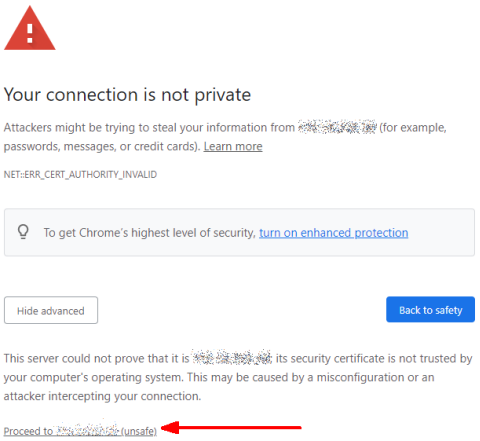
- **Generate a self-signed certificate** – this should only be a temporary solution. It is secure, but not safe. Data will be encrypted, but cannot be trusted.
- **Upload a certificate from a URL**
- **Upload a certificate from a user file system**

Generating a self-signed certificate

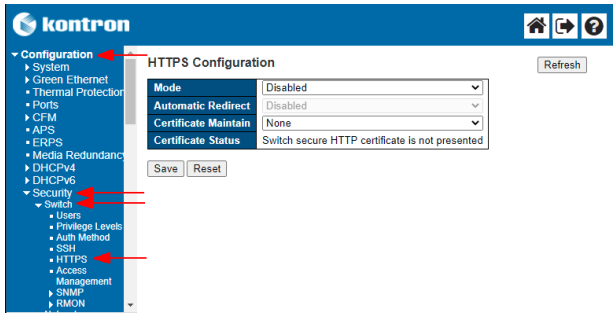
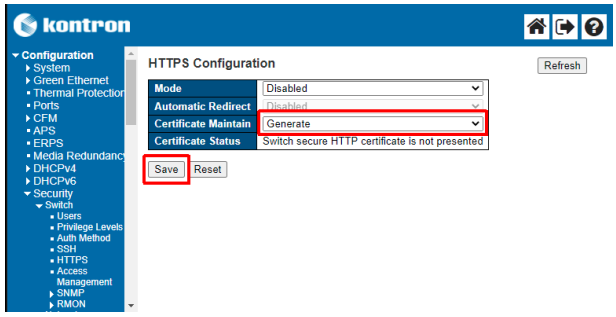
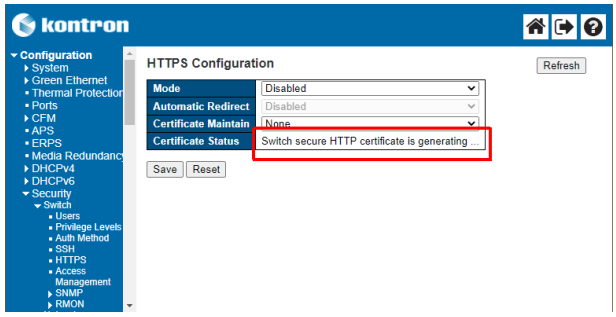
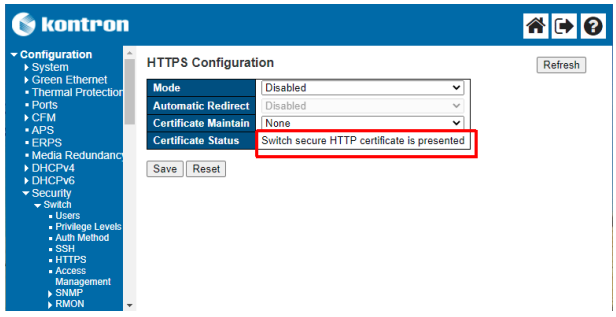
A self-signed certificate, which should only be used as a temporary solution, allows communication to be encrypted, but cannot certify that the server is really what it claims to be.
NOTE : The self-signed certificate will be valid for a fixed time period (e.g. November 30th 2021 at 00:00:01 up to November 30th 2031 at 23:59:59).
If a self-signed certificate is used, the Web browser will display a warning message before you can access the page. If this is the case, click on **Advanced.**



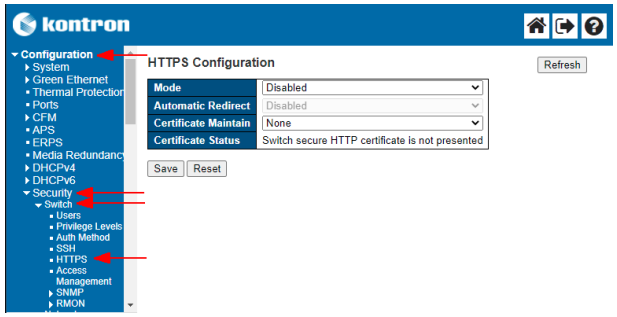
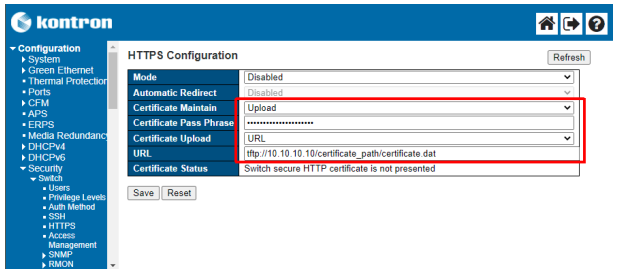
Then click on the **Proceed to [IP_ADDRESS] (unsafe)** link.



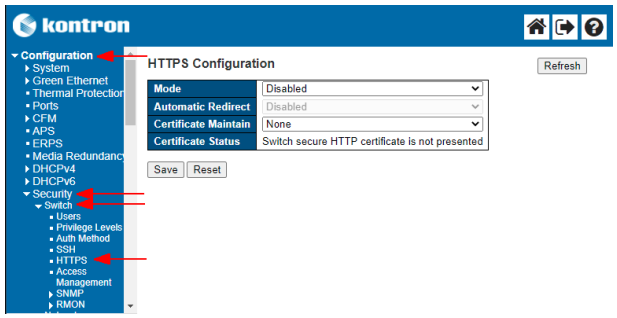
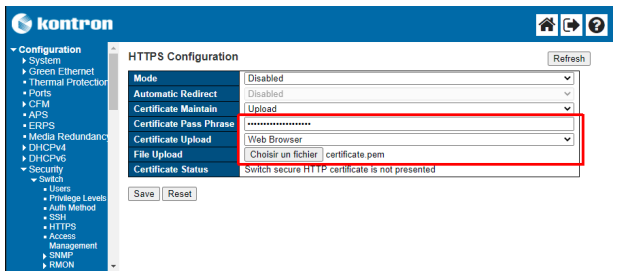
From the switch Web UI, perform the following steps.

Step_1	From the left-side menu, select **Configuration** , **Security** , **Switch** and then **HTTPS** .	
Step_2	Set the **Certificate Maintain** field to **Generate** .	
Step_3	Press **Save** to confirm.	
Step_4	The **Certificate Status** field will indicate that the switch is generating the certificate and will self-refresh.	
Step_5	The **Certificate Status** field will indicate that the certificate is present.	
Step_6	(Optional) To make the change persistent, save running-config to startup-config.	

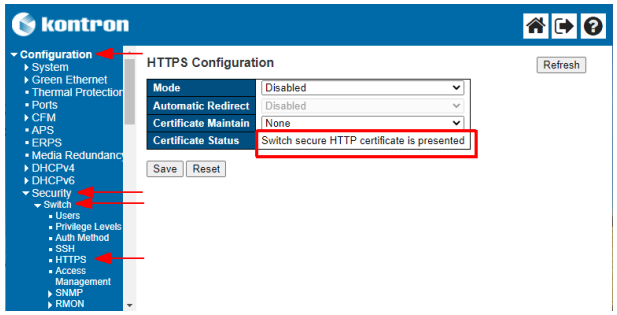
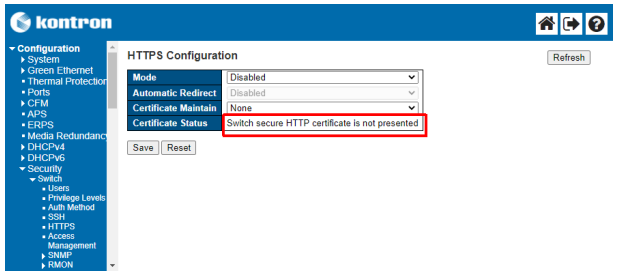
Uploading a certificate from a URL

Step_1	From the left-side menu, select **Configuration** , **Security** , **Switch** and then **HTTPS** .	
Step_2	Set the **Certificate Maintain** field to **Upload** .	
Step_3	Enter the pass phrase in the **Certificat Pass Phrase** field.	
Step_4	Set the **Certificate Upload** field to **URL** .	
Step_5	Enter the URL of the certificate in field **URL**.	
Step_6	Press **Save** to confirm.	
Step_7	The **Certificate Status** field will indicate that the certificate is present.	
Step_8	(Optional) To make the change persistent, save running-config to startup-config.	

Uploading a certificate from a user file system

Step_1	From the left-side menu, select **Configuration** , **Security** , **Switch** and then **HTTPS** .	
Step_2	Set the **Certificate Maintain** field to **Upload** .	
Step_3	Enter the pass phrase in the **Certificat Pass Phrase** field.	
Step_4	Set the **Certificate Upload** field to **Web Browser** .	
Step_5	In the **File Upload** field, click **Choose a file** and browse for the desired file.	
Step_6	Press **Save** to confirm.	
Step_7	The **Certificate Status** field will indicate that the certificate is in present.	
Step_8	(Optional) To make the change persistent, save running-config to startup-config.	

Deleting an installed certificate

Step_1	From the left-side menu, select **Configuration** , **Security** , **Switch** and then **HTTPS** . Ensure the **Certificate Status** is set to **Switch secure HTTP certificate is presented** .	
Step_2	Set the **Certificate Maintain** field to **Delete** .	
Step_3	The **Certificate Status** field will indicate that the **Switch secure HTTP certificate is not presented** .	
Step_4	Press **Save** to confirm.	
Step_5	(Optional) To make the change persistent, save running-config to startup-config.	

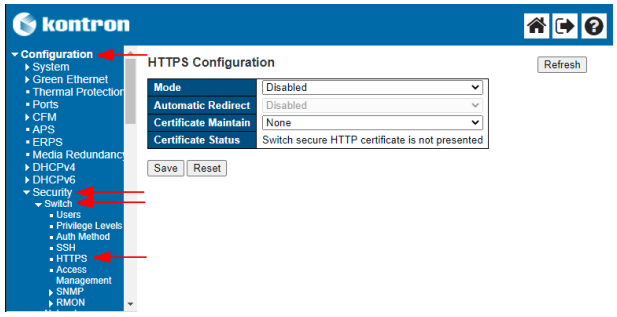
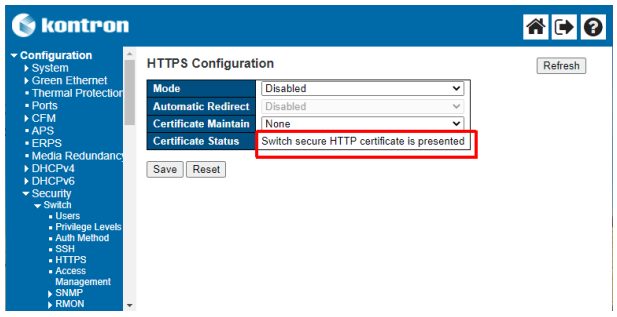
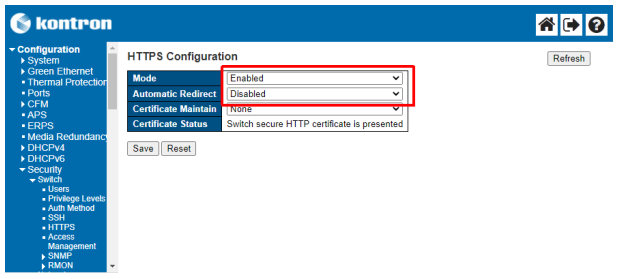
Configuring the interface protocol using the Web UI

Refer to Accessing the switch network operating system for access instructions.

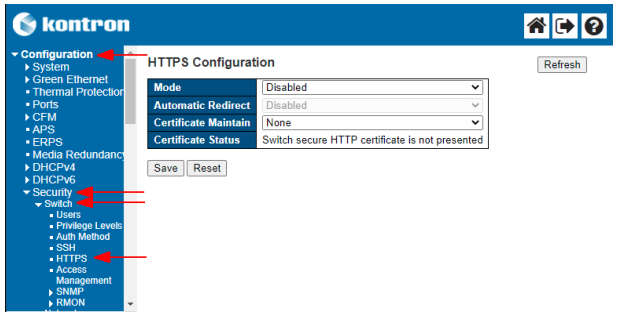
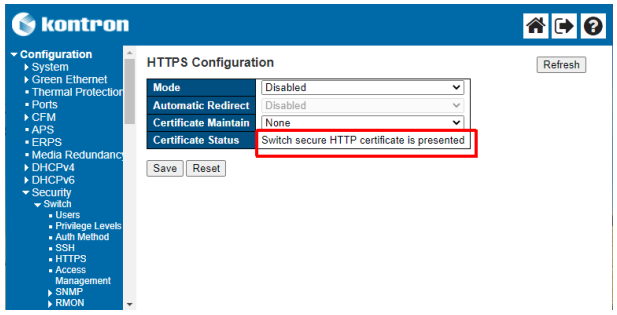
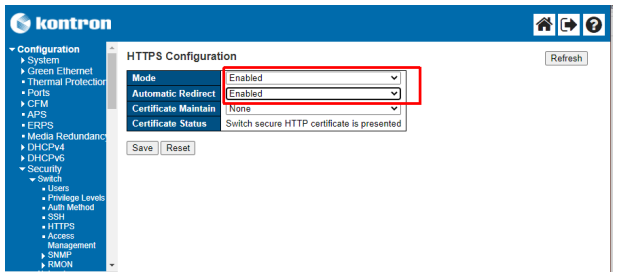
The interface protocol can be configured as:

- HTTPS (recommended)
- HTTP and HTTPS
- HTTP

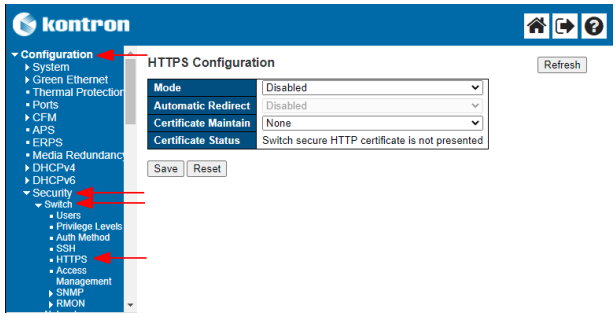
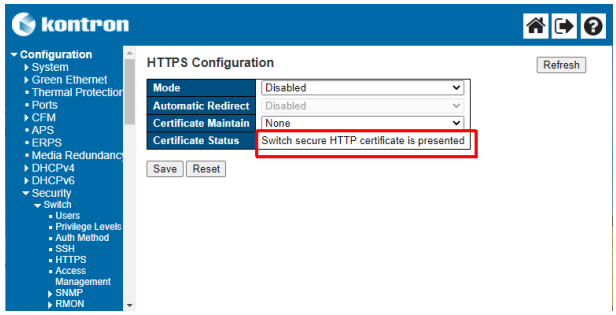
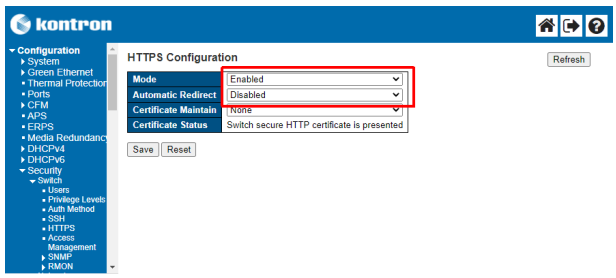
Configuring the interface for HTTP and HTTPS using the Web UI

Step_1	From the left-side menu, select **Configuration** , **Security** , **Switch** and then **HTTPS** .	
Step_2	Ensure the **Certificate Status** field is set to **Switch secure HTTP certificate is presented** .	
Step_3	Set the **Mode** field to **Enabled** .	
Step_4	Set the **Automatic Redirect** field to **Disabled** .	
Step_5	Press **Save** to confirm.	
Step_6	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring the interface for HTTPS only using the Web UI

Step_1	From the left-side menu, select **Configuration** , **Security** , **Switch** and then **HTTPS** .	
Step_2	Ensure the **Certificate Status** field is set to **Switch secure HTTP certificate is presented** .	
Step_3	Set the **Mode** field to **Enabled** .	
Step_4	Set the **Automatic Redirect** field to **Enabled** .	
Step_5	Press **Save** to confirm.	
Step_6	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring the interface for HTTP only using the Web UI

Step_1	From the left-side menu, select **Configuration** , **Security** , **Switch** and then **HTTPS** .	
Step_2	Ensure the **Certificate Status** field is set to **Switch secure HTTP certificate is presented** .	
Step_3	Set the **Mode** field to **Enabled** .	
Step_4	Set the **Automatic Redirect** field to **Disabled** .	
Step_5	Press **Save** to confirm.	
Step_6	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring SSH

SSH can be disabled and enabled.

Disabling SSH

Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

Step_1	Access the switch NOS CLI using the serial console from the integrated server.	
Step_2	Enter configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	[redacted]
Step_3	Enable or disable SSH. LocalSwitchNOS_OSPrompt:~(config)# **no ip ssh**	[redacted]
Step_4	Exit configuration mode. LocalSwitchNOS_OSPrompt:~(config)# **exit**	[redacted]
Step_5	(Optional) To make the change persistent, save running-config to startup-config.	

Enabling SSH

Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

Step_1	Access the switch NOS CLI using the serial console from the integrated server.	
Step_2	Enter configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	[redacted]
Step_3	Enable or disable SSH. LocalSwitchNOS_OSPrompt:~(config)# **ip ssh**	[redacted]
Step_4	Exit configuration mode. LocalSwitchNOS_OSPrompt:~(config)# **exit**	[redacted]
Step_5	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring switch NOS networking

Table of contents

Changes to the switch NOS configuration are not persistent after rebooting the switch NOS.
To preserve configurations, the current configuration needs to be saved to startup-config.
From the switch NOS Web UI:
- Select **Maintenance** , **Configuration** and then **Save startup-config** . Click on **Save Configuration** to confirm the change.

From the switch NOS CLI:
- LocalSwitchNOS_OSPrompt:~(config-if)# **end**
- LocalSwitchNOS_OSPrompt:~# **copy running-config startup-config**

Configuring IP addresses to access the switch NOS

This section is used to configure IP addresses allowing access to the configuration and management interfaces of the network operating system (NOS). This is the application responsible for implementing L2/L3 packet forwarding features.
One such feature is packet forwarding decisions based on VLAN tag. In that context, IP addresses to communicate with the NOS are attached to a VLAN defined in the NOS database. The switch always has at least VLAN1 that can be assigned an interface.
Refer to Configuring switch VLANs for procedures to add VLANs with the network operating system.

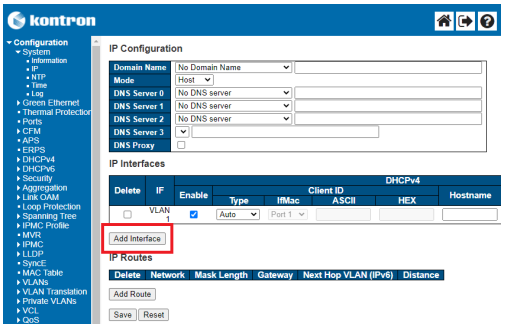
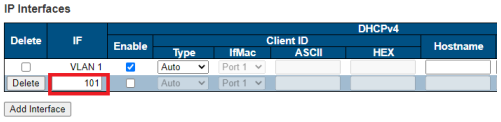
Adding a NOS VLAN interface IP address

This can be done using:
- The Web UI
- The CLI

Adding a NOS VLAN interface IP address using the Web UI

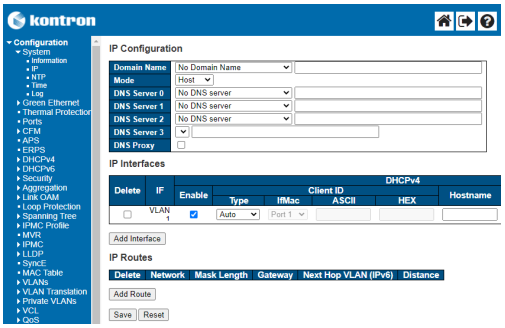
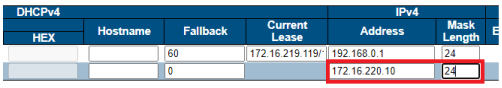
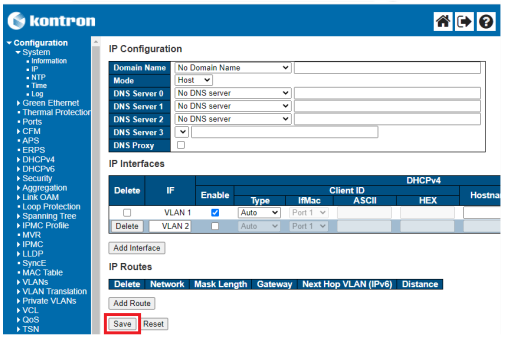
Refer to Accessing the switch NOS using the switch NOS Web UI for access instructions.

Adding a NOS VLAN interface

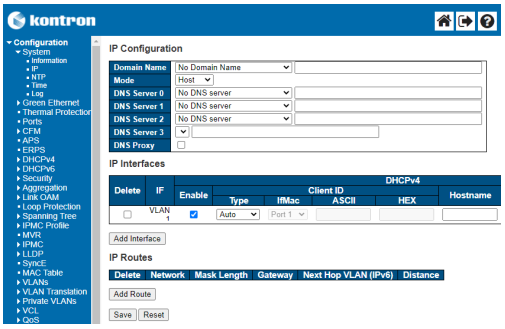
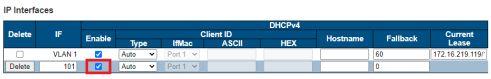
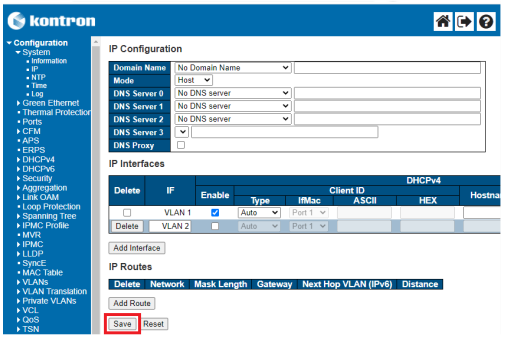
Step_1	From the left-side menu, select **Configuration** , **System** and then **IP** .	
Step_2	Click on the **Add Interface** button.	
Step_3	Enter the VLAN numerical ID. **NOTE:** As explained above, the VLAN must already exist to create the NOS IP address interface.	
Step_4	Proceed with IP address configuration as explained below.	

There are two options to configure IP addresses:
- Configuring a static IP address
- Configuring a dynamic IP address using DHCP

Configuring a static IP address

Step_1	From the left-side menu, select **Configuration** , **System** and then **IP** .	
Step_2	Manually configure the IP address and the network mask length of the VLAN interface.	
Step_3	Press on the **Save** button to confirm.	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring a dynamic IP address using DHCP

Step_1	From the left-side menu, select **Configuration** , **System** and then **IP** .	
Step_2	Enable the DHCP by checking the checkbox associated with the interface. The **Hostname** field allows the DHCP client to use a different hostname than the NOS for the DHCP option 12 field. The **Fallback** is a timeout in seconds after which the interface will be configured using the static IP address in the proper fields if an address cannot be obtained via DHCP.	
Step_3	Press on the **Save** button to confirm.	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Adding a NOS VLAN interface IP address using the CLI

Refer to Accessing the switch network operating system for access instructions.

Adding a NOS VLAN interface using a static IP address

Step_1	Enter the VLAN interface configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal** LocalSwitchNOS_OSPrompt:~(config)# **interface VLAN [VLAN_ID]**	`# configure terminal` `(config)# interface vlan 1`
Step_2	Set the static IP address source. LocalSwitchNOS_OSPrompt:~(config-if-vlan)# **ip address [IP_ADDRESS] [MASK]**	`(config-if-vlan)# ip address 192.168.0.1 255.255.255.0`
Step_3	(Optional) To make the change persistent, save running-config to startup-config.	

Adding a NOS VLAN interface using DHCP

Step_1	Enter the VLAN interface configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal** LocalSwitchNOS_OSPrompt:~(config)# **interface VLAN [VLAN_ID]**	`# configure terminal` `(config)# interface vlan 1`
Step_2	Set the IP address source to DHCP. LocalSwitchNOS_OSPrompt:~(config-if-vlan)# **ip address dhcp** **NOTE:** To view the IP address assigned, use command **do show ip interface** .	`(config-if-vlan)# ip address dhcp`
Step_3	(Optional) To make the change persistent, save running-config to startup-config.	

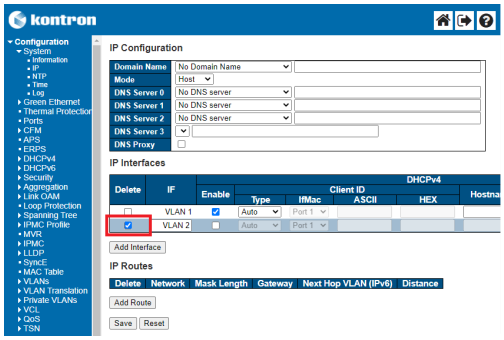
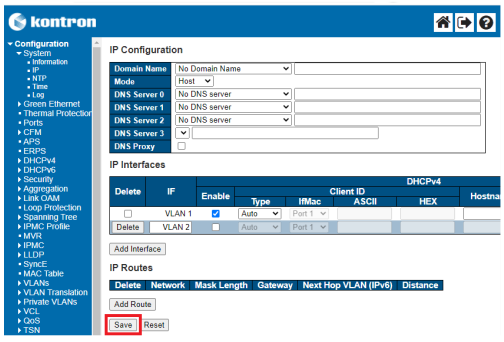
Removing a NOS VLAN interface IP address

This can be done using:

- The Web UI
- The CLI

Removing a NOS VLAN interface IP address using the Web UI

Refer to Accessing the switch NOS using the switch NOS Web UI for access instructions.

Step_1	From the left-side menu, select **Configuration** , **System** and then **IP** .	
Step_2	Select the VLAN interface to delete.	
Step_3	Press on the **Save** button to confirm.	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Removing a NOS VLAN interface IP address using the CLI

Refer to Accessing the switch network operating system for access instructions.

Step_1	Enter configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	`# configure terminal`
Step_2	Remove the VLAN. LocalSwitchNOS_OSPrompt:~(config)# **no interface vlan [VLAN_ID]**	`(config)# no interface vlan 101`
Step_3	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring the serial port linking the switch NOS to the GNSS

Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

Step_1	Enter configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	
Step_2	Configure the clock for auto select clock control, based on PTP profile and available hardware resources. LocalSwitchNOS_OSPrompt:~# **ptp ext output auto**	
Step_3	Configure the serial link between the NOS and the GNSS. LocalSwitchNOS_OSPrompt:~# **ptp rs422 baudrate 115200 parity none wordlength 8 stopbits 1 flowctrl none**	
Step_4	Configure the protocol to use over the serial link. LocalSwitchNOS_OSPrompt:~# **ptp rs422 sub ser proto rmc**	
Step_5	End the configuration. LocalSwitchNOS_OSPrompt:~# **end**	
Step_6	(Optional) To make the change persistent, save running-config to startup-config. LocalSwitchNOS_OSPrompt:~# **copy running-config startup-config**	

Configuring the network switch

Table of contents

Relevant sections:

- Accessing the switch network operating system
- Accessing the operating system of a server
- Configuring and managing users

Changes to the switch NOS configuration are not persistent after rebooting the switch NOS.
To preserve configurations, the current configuration needs to be saved to startup-config.
From the switch NOS Web UI:

- Select **Maintenance** , **Configuration** and then **Save startup-config** . Click on **Save Configuration** to confirm the change.

From the switch NOS CLI:

- LocalSwitchNOS_OSPrompt:~(config-if)# **end**
- LocalSwitchNOS_OSPrompt:~# **copy running-config startup-config**

Help tools

Switch Web user interface help

The Help menu of the switch Web user interface is comprehensive. It should be used to configure the system.

Switch CLI help

The switch CLI contains a context-sensitive help feature. Use the **?** symbol to display the next possible parameters or commands and their descriptions. Almost all configuration commands have a corresponding 'no' form. The 'no' form is syntactically similar (but not necessarily identical) to the configuration command; however, it either resets the parameters to default values for the configurable item or disables the item altogether.

```
NOS00A0A5E01CF4# show interface * ?
    <port_type_list>    Port list for all port types
    capabilities        Display capabilities.
    description         Description of interface
    statistics          Display statistics
    status              Display status.
    switchport          Show interface switchport information
    transceiver         Show SFP transceiver properties
    veriphy             Display the latest cable diagnostic results.
NOS00A0A5E01CF4# show interface *
```

Switch NOS port mapping

The following table lists the physical ports of the Ethernet switch of an S1901. Note that, in the switch NOS, physical ports are a category of interfaces. The port designation is used in CLI commands, denoted by [INTERFACE_ID] below, to monitor or configure the corresponding port.

Connector	Speed	[INTERFACE_ID]	Port number
GbE 0	1 Gbps		COMe eno2
GbE 1	1 Gbps	Eth 1/1	NOS "Ethernet 1/1" or "Eth 1/1"
GbE 2	1 Gbps	Eth 1/2	NOS "Ethernet 1/2" or "Eth 1/2"
GbE 3	1 Gbps	Eth 1/3	NOS "Ethernet 1/3" or "Eth 1/3"
GbE 4	1 Gbps	Eth 1/4	NOS "Ethernet 1/4" or "Eth 1/4"
GbE 5	1 Gbps	Eth 1/5	NOS "Ethernet 1/5" or "Eth 1/5"
GbE 6	1 Gbps	Eth 1/6	NOS "Ethernet 1/6" or "Eth 1/6"
2.5GbE	2.5 Gbps	Eth 1/7	NOS "Ethernet 1/7" or "Eth 1/7"
10GbE	10 Gbps	Eth 1/8	NOS "Ethernet 1/8" or "Eth 1/8"
Internal	10 Gbps	Eth 1/9	NOS "Ethernet 1/9" or "Eth 1/9" **NOTE:** This is the network connection between the network switch and COMe eno1.

Relevant section:
Linux devices (for information on automotive Ethernet ports)

Verifying link status

Link status can be verified using:

- The CLI
- The switch Web UI

Verifying link status using the CLI

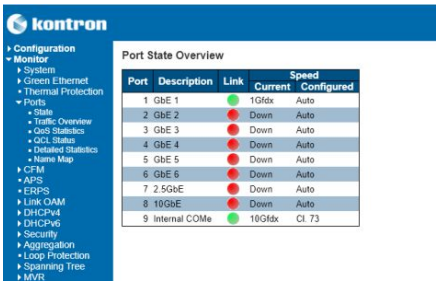
Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

Step_1	List link name or description.
	LocalSwitchNOS_OSPrompt:~# **show interface * description**
	`NOS00A0A5E01C4F# show interface * description`
	`Interface  Description`
	`---------  -----------`
	`Eth 1/1  GbE 1`
	`Eth 1/2  GbE 2`
	`Eth 1/3  GbE 3`
	`Eth 1/4  GbE 4`
	`Eth 1/5  GbE 5`
	`Eth 1/6  GbE 6`
	`Eth 1/7  2.5 GbE`
	`Eth 1/8  10 GbE`
	`Eth 1/9  Internal to COMe`

Step_2	Verify every link status.
	LocalSwitchNOS_OSPrompt:~# **show interface * status**
	NOTE : On the last line, under Media Type (**SFP**), column SFP Family (**10G CuBP**) must also be read. It means a **10GbE Copper Back Plane** link.
	`NOS00A0A5E01C4F# show interface * status`
	`Interface  Mode  Speed  Aneg  Media Type  SFP Family  Link  Operational Warnings`
	`Eth 1/1  Enabled  Auto  Yes  RJ45  N/A  1Gfdx`
	`Eth 1/2  Enabled  Auto  Yes  RJ45  N/A  1Gfdx`
	`Eth 1/3  Enabled  Auto  Yes  RJ45  N/A  Down`
	`Eth 1/4  Enabled  Auto  Yes  RJ45  N/A  Down`
	`Eth 1/5  Enabled  Auto  Yes  RJ45  N/A  Down`
	`Eth 1/6  Enabled  Auto  Yes  RJ45  N/A  Down`
	`Eth 1/7  Enabled  Auto  Yes  RJ45  N/A  Down`
	`Eth 1/8  Enabled  Auto  Yes  RJ45  N/A  Down`
	`Eth 1/9  Enabled  Cl. 73  Yes (Cl73) SFP  10G CuBP  10Gfdx`

Verifying link status using the Web UI

Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu, select **Monitor** , **Ports** and then **State** . The **Port State Overview** will display the status of the links.	

Enabling a switch port

Switch ports can be enabled using:

- The CLI
- The switch Web UI

Enabling a switch port using the CLI

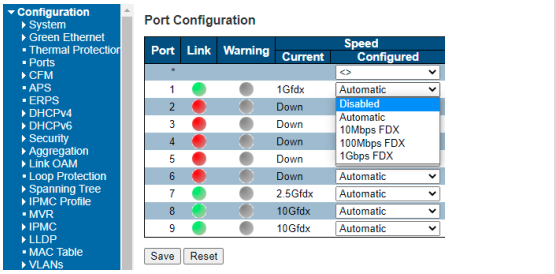
Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

Step_1	Access the interface setup menu.
	LocalSwitchNOS_OSPrompt:~# **configure terminal**
	LocalSwitchNOS_OSPrompt:~(config)# **interface [INTERFACE_ID]**
		`# configure terminal`
		`(config)# interface Ethernet 1/2`
		`(config-if)#`

Step_2	Enable the interface. LocalSwitchNOS_OSPrompt:~(config-if)# **no shutdown**	`(config-if)# no shutdown`
Step_3	(Optional) To make the change persistent, save running-config to startup-config.	

Enabling a switch port using the Web UI

Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu, select **Configuration** and then **Ports** .	
Step_2	Enable a switch port by selecting its speed configuration. **NOTE:** Recommended setting is **Automatic** .	
Step_3	Press on the **Save** button to confirm. **NOTE:** Click on **Refresh** to refresh the Link column status LEDs.	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Disabling a switch port

Switch ports can be disabled using:

- The CLI
- The switch Web UI

Disabling a switch port using the CLI

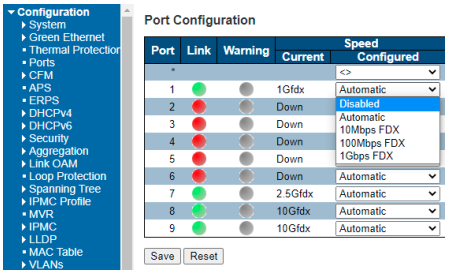
Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

Step_1	Access the interface setup menu.
	LocalSwitchNOS_OSPrompt:~# **configure terminal**
	LocalSwitchNOS_OSPrompt:~(config)# **interface [INTERFACE_ID]**
		`# configure terminal`
		`(config)# interface Ethernet 1/2`
		`(config-if)#`

Step_2	Disable the interface. LocalSwitchNOS_OSPrompt:~(config-if)# **shutdown**	`(config-if)# shutdown`
Step_3	(Optional) To make the change persistent, save running-config to startup-config.	

Disabling a switch port using the Web UI

Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu, select **Configuration** and then **Ports** .	
Step_2	Disable a switch port by changing its speed configuration to **Disabled** .	
Step_3	Press on the **Save** button to confirm.	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Changing link speed

Link speed can be changed using:

- The CLI
- The switch Web UI

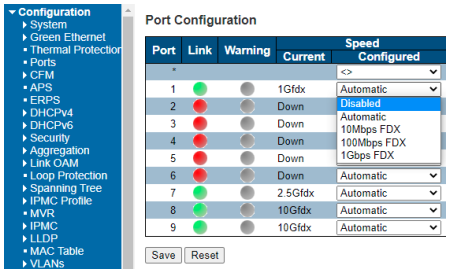
Changing link speed using the CLI

Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.
NOTE: For interfaces 2.5GbE (Eth 1/7) and 10GbE (Eth 1/8), the speeds currently supported are **"auto 10g"** (10GbE only), **"auto 2.5g"** , **"auto 1000"** or **"auto"** .

Step_1	Enter the configuration terminal. LocalSwitchNOS_OSPrompt:~# **configure terminal**	`# configure terminal`
Step_2	Enter the interface configuration menu. LocalSwitchNOS_OSPrompt:~(config)# **interface [INTERFACE]**	`(config)# interface Eth 1/8`
Step_3	Change the speed. LocalSwitchNOS_OSPrompt:~(config-if)# **speed [SPEED]**	`(config-if)# speed auto 1000`
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Changing link speed using the Web UI

Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu, select **Configuration** , and then **Ports** .	
Step_2	Select a value from the **Speed** dropdown menu.	
Step_3	Press on the **Save** button to confirm.	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring switch VLANs

Several VLAN configurations can be performed using the CLI or the switch Web UI:

- Displaying a VLAN
- Creating a VLAN
- Removing a VLAN
- Configuring the port membership

Displaying VLANs

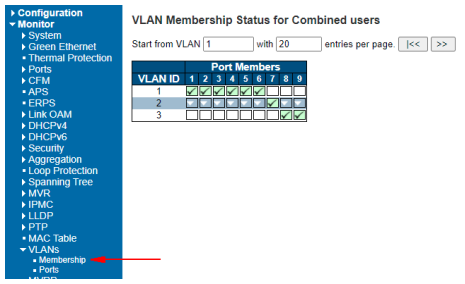
Displaying VLANs using the CLI

Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

Step_1	Display the VLAN status for every switch port.
	LocalSwitchNOS_OSPrompt:~# **show vlan**
		`NOS00A0A5E01C4F# show vlan`
		`VLAN  Name                            Interfaces`
		`----  --------------------------------  ----------`
		`1     default                         Eth 1/1-6`
		`2     VLAN0002                        Eth 1/7`
		`3     VLAN0003                        Eth 1/8-9`

Displaying VLANs using the Web UI

Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu, select **Monitor** , **VLANs** and then **Membership** . The VLAN port membership should be displayed.	

Creating a VLAN

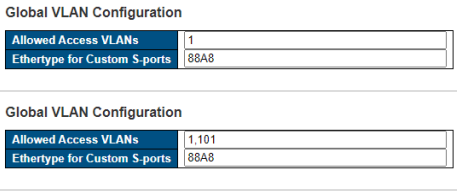
Creating a VLAN using the CLI

Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

Step_1	Enter configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	`# configure terminal`
Step_2	Create a new VLAN. LocalSwitchNOS_OSPrompt:~(config)# **vlan [VLAN_ID]**	`(config)# vlan 9` `(config-vlan)#`
Step_3	(Optional) To make the change persistent, save running-config to startup-config.	

Creating a VLAN using the Web UI

Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu, select **Configuration** , **VLANs** and then **Configuration** .	
Step_2	From the **Global VLAN Configuration** , add the desired VLAN(s) to the **Allowed Access VLANs** list. **NOTE:** The list of VLANs needs to be delimited by commas between each interface ID, or by a dash to indicate a serie (1-5 meaning the same as 1,2,3,4,5).	
Step_3	Click on the **Save** button.	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Removing a VLAN

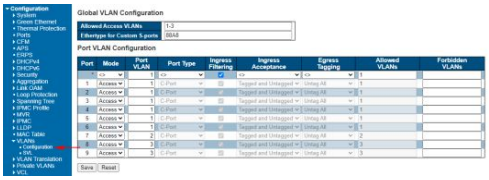
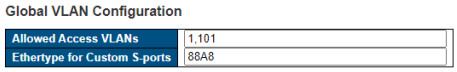
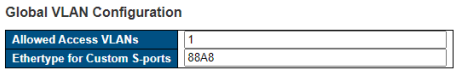
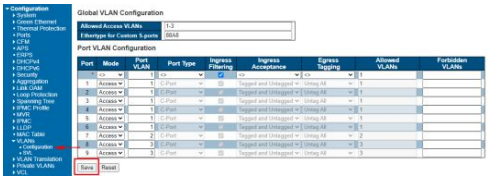
Removing a VLAN using the CLI

Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

Step_1	Enter configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	`# configure terminal`
Step_2	Remove a VLAN using the following command. LocalSwitchNOS_OSPrompt:~(config)# **no vlan [VLAN_ID]**	`(config)# no vlan 9` `(config)#`
Step_3	(Optional) To make the change persistent, save running-config to startup-config.	

Removing a VLAN using the Web UI

Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu, navigate to **Configuration** , **VLANs** , and then **Configuration** .	
Step_2	From the **Global VLAN Configuration** , remove the desired VLANs from the **Allowed Access VLANs** list.	 
Step_3	Click on the **Save** button.	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring VLAN port membership

Configuring port membership using the CLI

Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

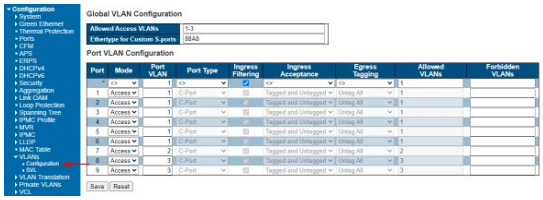
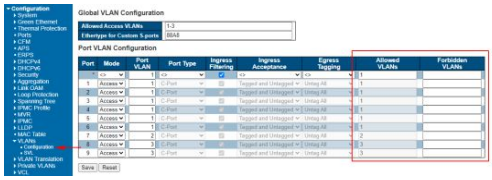
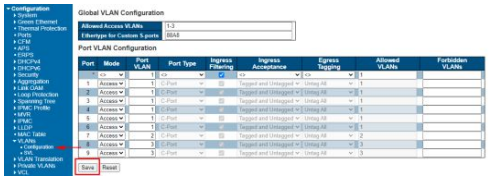
Step_1	Access the desired interface configuration menu. LocalSwitchNOS_OSPrompt:~# **configure terminal** LocalSwitchNOS_OSPrompt:~(config)# **interface [INTERFACE_ID]**	`# configure terminal` `(config)# interface Ethernet 1/3`

Step_2	Proceed with port membership configuration. Use the built-in help feature using " **?** " to see the possible configurations.
	VLAN membership configuration command descriptions:
	• Adding one or multiple VLANs using the **add** command.
	• Adding all currently defined VLANs using the **all** command.
	• Excluding one or multiple VLANs using the **except** command.
	• Excluding all currently defined VLANs using the **none** command.
	• Removing one or multiple VLANs using the **remove** command.
	LocalSwitchNOS_OSPrompt:~(config-if)# **switchport hybrid allowed vlan add [VLAN_ID]**
		`(config-if)# switchport hybrid allowed vlan`
		`<vlan_list>  add  all  except  none  remove`
		`(config-if)# switchport hybrid allowed vlan  add 1`

Step_3	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring port membership using the Web UI

Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu, navigate to **Configuration** , **VLANs** and then **Configuration** .	
Step_2	Proceed with port membership configuration using the last two columns. The list of VLANs is constructed using a comma to separate elements or a hyphen to describe a range. Example: 1,101-103,4093 Which is equivalent to: 1,101,102,103,4093	
Step_3	Press on the **Save** button to confirm.	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Setting an IP address for a VLAN

If users wish to access the switch CLI on a VLAN (VLAN 1 is there by default), an IP address can be assigned. **NOTE:** If no IP address is set for a VLAN, the CLI will not be accessible from this VLAN.
Two methods can be used to set an IP address:

- Using a DHCP server
- Manually

Relevant section:
MAC addresses

Setting an IP address for a VLAN using a DHCP server

Step_1	LocalSwitchNOS_OSPrompt:~# **configure terminal**
	`NOS00A0A5E02722# configure terminal`
Step_2	LocalSwitchNOS_OSPrompt:~# **interface vlan [VLAN_ID]**
	`NOS00A0A5E02722(config)# interface vlan 2`
Step_3	LocalSwitchNOS_OSPrompt:~# **ip address dhcp fallback [DHCP_SERVER_IP_ADDRESS] [MASK_ADDRESS] timeout [SECONDS]** **NOTE:** The timeout command portion is optional.
	`NOS00A0A5E02722(config-if-vlan)# ip address dhcp fallback 192.168.0.1 255.255.255.0 timeout 60`
Step_4	(Optional) To make the change persistent, save running-config to startup-config.

Setting an IP address for a VLAN manually

Step_1	LocalSwitchNOS_OSPrompt:~# **configure terminal**
	`NOS00A0A5E02722# configure terminal`
Step_2	LocalSwitchNOS_OSPrompt:~# **interface vlan [VLAN_ID]**
	`NOS00A0A5E02722(config)# interface vlan 2`
Step_3	LocalSwitchNOS_OSPrompt:~# **ip address [PLANNED_VLAN_IP_ADDRESS] [MASK_ADDRESS]**
	`NOS00A0A5E02722(config-if-vlan)# ip address 192.168.100.2 255.255.255.0`
Step_4	(Optional) To make the change persistent, save running-config to startup-config.

Configuring static routing

Static routing can be configured using:

- The CLI
- The switch Web UI

Configuring static routing using the CLI

Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

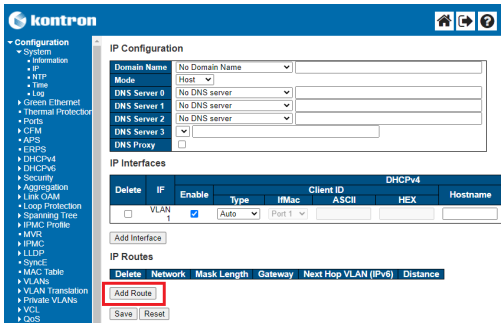
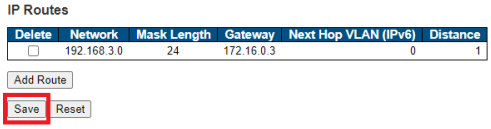
Step_1	Enter configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	`# configure terminal`
Step_2	Configure static routing. LocalSwitchNOS_OSPrompt:~(config)# **ip route [HOST_ADDRESS] [NETWORK_MASK] [GATEWAY_ADDRESS]**	`(config)# ip route 192.168.3.0 255.255.255.0 172.16.0.3`
Step_3	Exit the configuration menu. LocalSwitchNOS_OSPrompt:~(config)# **exit**	

Step_4	Display the list of routes to confirm the static route was added.
	LocalSwitchNOS_OSPrompt:~# **show ip route**
		`# show ip route`
		`Codes: C - connected, S - static`
		`* - FIB route, D - DHCP installed route`
		`D* 0.0.0.0/0 [253/0] via 172.16.0.1, VLAN 1, 18:33:31`
		`C* 172.16.0.0/16 is directly connected, VLAN 1, 18:33:31`
		`S* 192.168.3.0/24 [1/0] via 172.16.0.3, VLAN 1, 18:33:31`

Step_5	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring static routing using the Web UI

Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu, select **Configuration** , **System** and then **IP** .	
Step_2	Click on the **Add Route** button.	
Step_3	Proceed with configuration: • Enter host address in the **Network** column. • Enter network mask in number of bits in the **Mask Length** column. • Enter the gateway address in the **Gateway** column. • Configure the **Next Hop VLAN (IPv6)** and **Distance** parameters, if needed.	
Step_4	Press on the **Save** button to confirm.	
Step_5	(Optional) To make the change persistent, save running-config to startup-config.	

Managing the switch configuration

Managing the switch configuration using the CLI

Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

Displaying the running configuration using the CLI

NOTE: If you have display problems, refer to Minicom problems .

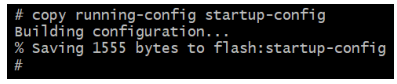
Step_1	Display the current configuration.
	LocalSwitchNOS_OSPrompt:~# **show running-config**
		`NOS00A0A5E10E54# show running-config`
		`Building configuration...`
		`username admin privilege 15 password encrypted 4114dc09c554cbc78c5d5916ca7d0267a`
		`66c020fb4abeac88b9085491dea74e127c29e0f5fdf14e100c82f46d2410c830045931f03770adda`
		`c2c9f1bf89d4227`
		`!`
		`vlan 1`
		`!`
		`!`
		`!`
		`!`
		`spanning-tree mst name 00-a0-a5-e1-0e-54 revision 0`
		`!`
		`!`
		`ptp ext output auto`
		`ptp rs422 main-auto ser proto rmc`
		`!`
		`-- more --, next page: Space, continue: g, quit: ^C`

Displaying versions

Step_1	Display versions. LocalSwitchNOS_OSPrompt:~# **show version**	

Saving the current configuration using the CLI

Changes to the switch configuration are not persistent after rebooting the switch. To preserve custom configurations, use the following command.

Step_1	Save the current configuration. LocalSwitchNOS_OSPrompt:~# **copy running-config startup-config**	

Restoring the default configuration using the CLI

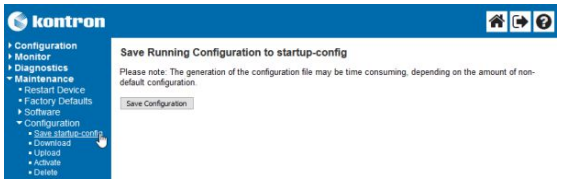
Step_1	Restore the default configuration. LocalSwitchNOS_OSPrompt:~# **reload defaults**	
Step_2	(Optional) To make the change persistent, save running-config to startup-config.	

Managing the switch configuration using the Web UI

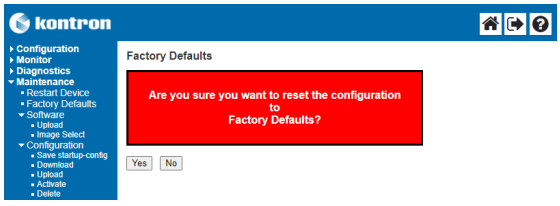
Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Saving the current configuration using the Web UI

Changes to the switch configuration are not persistent after rebooting the switch. To preserve custom configurations, use the following command.

Step_1	From the left-side menu, select **Maintenance** , **Configuration** , and then **Save startup-config** .	
Step_2	Press on the **Save Configuration** button.	

Restoring the default configuration using the Web UI

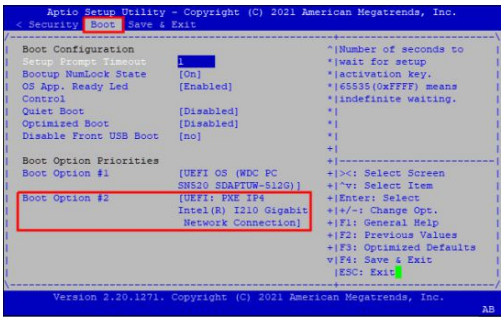
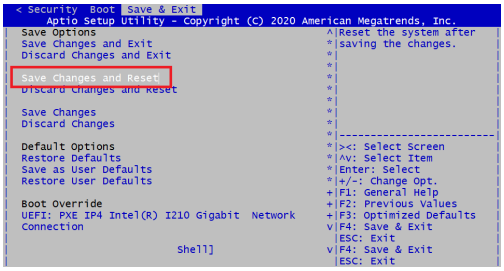
Step_1	From the left-side menu, select **Maintenance** and then **Factory Defaults** .	
Step_2	Press on the **Yes** button to confirm the choice.	

Configuring UEFI BIOS options

Table of contents

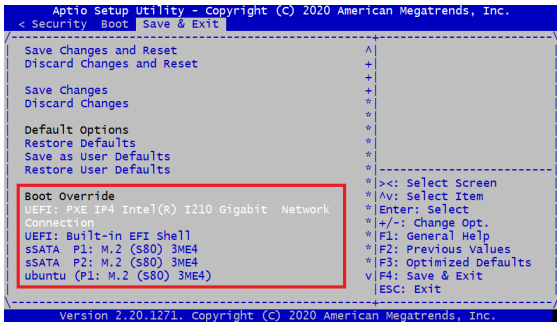
Access the UEFI/BIOS. Refer to Accessing the UEFI BIOS f or access instructions.

Changing the boot order

Step_1	From the UEFI/BIOS setup menu, navigate to the **Boot** menu. Configure the boot order as desired.	
Step_2	Select the **Save & Exit** menu, go to **Save Changes and Reset** and press **Enter** to confirm and save the new boot order.	

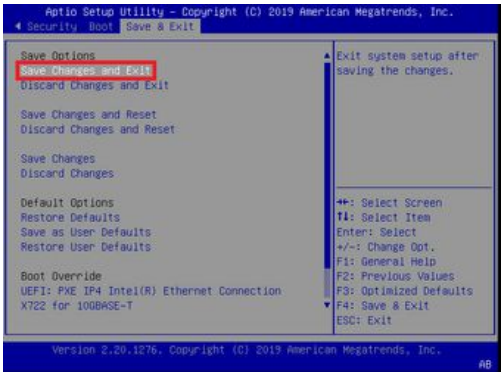
Overriding the boot order

This is a non-persistent option to allow booting to a specific device while maintaining the normal boot order.

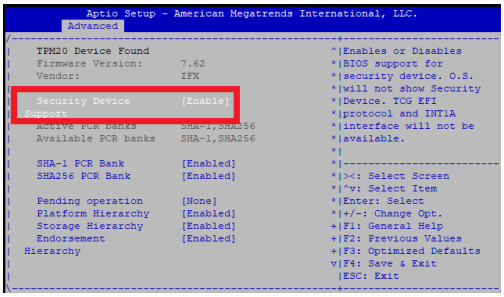
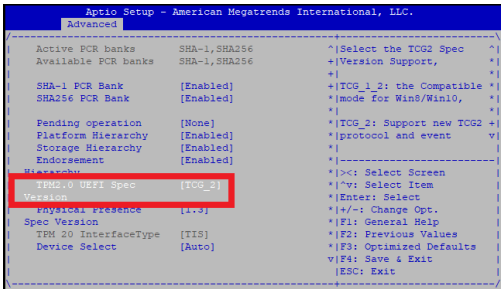
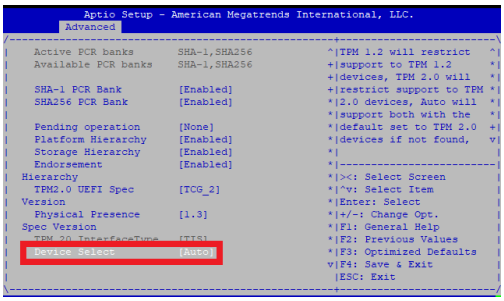
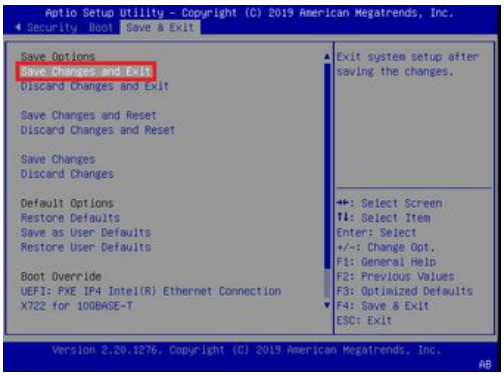
Step_1	Reboot the platform and access the UEFI/BIOS setup menu.	
Step_2	Navigate to the **Save & Exit** menu and then to the **Boot Override** section.	

Enabling secure boot

The following application notes are required to generate secure boot keys and configure them: Generating custom secure boot keys and Provisioning custom secure boot keys .

Step_1	Navigate to the **Security** tab and access the **Secure Boot** submenu.	
Step_2	Select the **Secure Boot** option and change it to **Enabled** .	
Step_3	Use the application notes mentioned above as reference to generate and configure secure boot keys.	
Step_4	Navigate to the **Save & Exit** menu, go to **Save Changes and Exit** and press **Enter** to confirm.	

Configuring the TPM

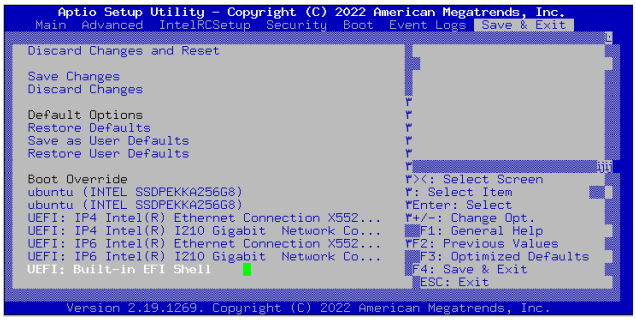
Step_1	Navigate to the **Advanced** menu, go to **Trusted Computing** and then **Security Device Support**. Verify that it is set to **Enable** . Possible values: [Enable / Disable] **NOTE:** The TPM has to be inserted to see the menu.	
Step_2	From the **Advanced** menu and the **Trusted Computing** section, select **TPM2.0 UEFI Spec Version** and set the applicable spec. Possible values: [TCG_1_2 / TCG_2] **NOTE:** The TPM has to be inserted to see the menu.	
Step_3	From the **Advanced** menu and the **Trusted Computing** section, select **Device Select** and set the applicable device. Possible values: [TPM 1.2 / TPM 2.0 / Auto] **NOTE:** The TPM has to be inserted to see the menu.	
Step_4	Navigate to the **Save & Exit** menu, go to **Save Changes and Exit** and press **Enter** to confirm.	

Configuring the PBIT

Relevant section:
User guide for PBIT

Disabling "Network Stack" doesn't prevent Intel Network(s) to be seen in the Advanced menu.
The PBIT feature utilizes a part of the UEFI network driver to test the network, but the driver menu will still be present.

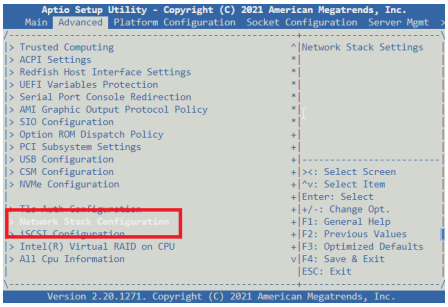
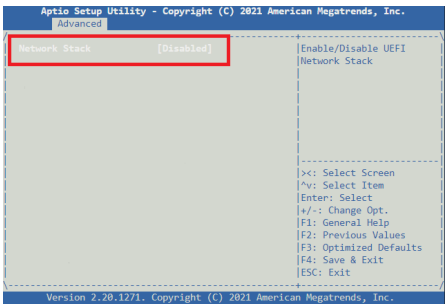
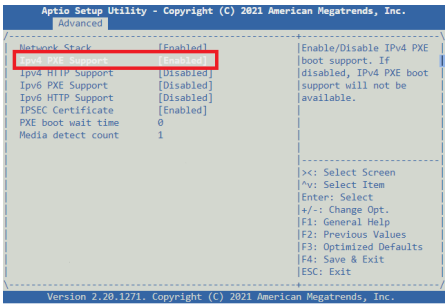
To use the PBIT tool and read its result, it must be enabled in the EFI Shell.

Step_1	From the UEFI/BIOS setup menu, navigate to the **Save & Exit** menu and then to the **Boot Override** section. Select **UEFI: Built-in EFI Shell** .	
Step_2	Run the command that will teach the PBIT the system configuration. Shell> **kdiag learn system**	kdiag learn system Seg Bus Dev Func --- --- --- ---- 00 00 00 00 ==> Bridge Device - Host/PCI bridge Vendor 8086 Device 6F00 Prog Interface 0 00 00 01 00 ==> Bridge Device - PCI/PCI bridge Vendor 8086 Device 6F02 Prog Interface 0 00 0E 00 00 ==> Network Controller - Ethernet controller Vendor 8086 Device 1531 Prog Interface 0 00 14 00 00 ==> Network Controller - Ethernet controller Vendor 8086 Device 1533 Prog Interface 0 FPGA Version 0x8010009 DRAM size 64 MB BIOS Setup Checksum : 131274 BIOS Version : S1901R110 3 Hubs USB Dev 1 connected USB Dev 3 connected USB Dev 5 connected USB Dev 10 connected USB Dev 11 connected Number of System Test Elements detected : 50 DRAM area [0x7316A0A0 0x7316B1A8] will be stored in EEPROM Storing system infos... Storing system configuration...
Step_3	Exit the EFI Shell. Shell> **exit**	

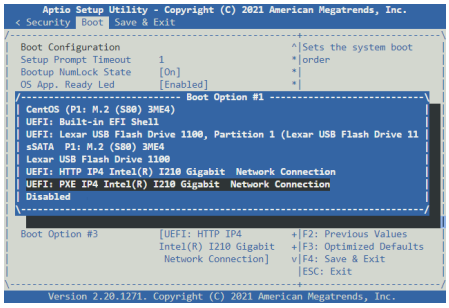
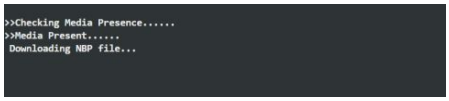
Configuring PXE network boot

Enabling PXE support

	Disabling "Network Stack" doesn't prevent Intel Network(s) to be seen in the Advanced menu. The PBIT feature utilizes a part of the UEFI network driver to test the network, but the driver menu will still be present.

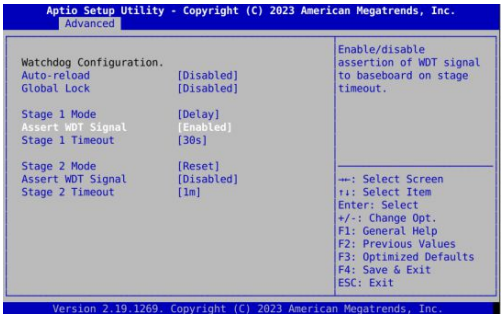
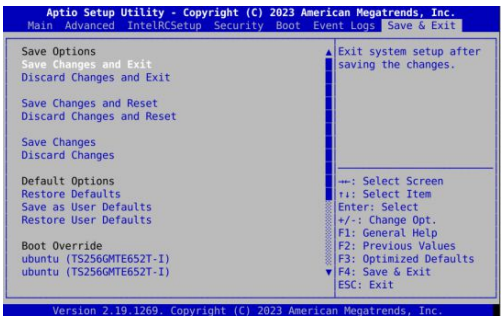
Step_1	From the UEFI/BIOS setup menu, navigate to the **Advanced** menu and enter the **Network Stack Configuration** sub-menu.	
Step_2	If needed, enable the **Network Stack**. **NOTE:** If the network stack is disabled, the UEFI network boot is consequently disabled.	
Step_3	Enable or disable the **IPv4 PXE Support** and/or **IPv6 PXE Support** .	
Step_4	Select the **Save & Exit** menu, go to **Save Changes and Reset** and press **Enter** .	

Performing PXE network boot

Step_1	From the UEFI/BIOS setup menu, navigate to the **Boot** menu. Configure the boot order to as desired. The PXE boot option should be first in order to have priority over the other boot options. **NOTE:** Boot override can also be used to choose manually for a one-time boot.	
Step_2	Select the **Save & Exit** menu, go to **Save Changes and Reset** and press **Enter** to confirm and save the new boot order. The platform should boot using PXE.	

Configuring the COMe w atchdog

The COMe watchdog behavior can be configured based on the application used.
Refer to the COMe user guide for specific information on the various configurations.

Step_1	Navigate to the **Advanced** menu, go to **Miscellaneous** and then **Watchdog Configuration** . S elect **Stage 1 Mode** and set it to the desired behavior [**Disabled, Reset, Delay, WDT Signal only**]. **NOTE:** WDT Signal is connected to the CAN bus mezzanine with the AURIX safety MCU.	
Step_2	This is an example of available options when **Reset** is selected at Step_1. The timeout can be set to various values [1s / 5s 10s / **30s** / 1m / 3m /10m / 30m].	
Step_3	This is an example of available options when **Delay** is selected at Step_1. The behavior will be as follows based on the selection shown: • When 30 seconds have elapsed since the watchdog was last serviced, the WDT Signal linked to the AURIX MCU will be triggered. • After another 60 seconds elapse, a reset will be done if the watchdog was not serviced.	
Step_4	Once configuration is done, navigate to the **Save & Exit** menu, go to **Save Changes and Exit** and press **Enter** to confirm.	

Configuring serial ports

Table of contents

Serial ports of S1901 platforms can be configured. It is possible to:

- Configure specific serial ports as either RS-232 or RS-485/422
- Enable or disable the internal chip receiver termination on RS-485/422 ports

Relevant sections:

Linux devices
Connector pinouts for building custom cables
Kontron test cables

Default serial port configurations

Serial ports	Connector	Default configuration	Console parameters
RS-232 #1, #2, #3, #4	J1	RS-232 (Rx, Tx) Not configurable	No hardware flow control
RS-232 #5 or RS-485/422 #1	J1	RS-485 full-duplex	Hardware flow control (RTS/CTS, DSR/DTR) in RS-232
RS-232 #6 or RS-485/422 #2	J1	RS-485 full-duplex	Hardware flow control (RTS/CTS, DSR/DTR) in RS-232
RS-485/422 #3, #4	J2	RS-485 full-duplex Configuration not documented. For detailed information, refer to the documentation on chip LTM2881.	

Voltage driven by RS-485/422 ports

The platform can drive voltages of -7 V to +12 V. Users must ensure all receivers can handle these voltages.

Configuring specific serial ports as either RS-232 or RS-485/422

In an S1901 platform, the following ports can be configured as either RS-232 or RS-485/422:

- RS-232 #5 or RS-485/422 #1
- RS-232 #6 or RS-485/422 #2

Both configurable serial ports are on the same chip. The pins required for configuration are listed in the table below.

LTC2872 pin	Signal name	Serial port controlled	Connected to chip, pin
485/232#_1# (default 1, RS-485)	XR_COM_MODE0	RS-232 #5 or RS-485/422 #1	exar_gpio 0, 0
TE485_1 (default 0, termination OFF)	XR_COM_TE0	RS-232 #5 or RS-485/422 #1	exar_gpio 0, 1
H/F# (default 0, full-duplex)	XR_COM_FULL#	RS-232 #5 or RS-485/422 #1 RS-232 #6 or RS-485/422 #2 **NOTE:** When both RS-485/422 ports are enabled, both ports must be in the same mode (half-duplex or full-duplex).	pca9539-[SMBUS_NO]-0074 , 10 **NOTE:** To define the [SMBUS_NO] variable, refer to Defining the I2C bus number .
DXEN1 (default 1, transmitter enabled)	XR_COM_DXEN0	RS-232 #5 or RS-485/422 #1	exar_gpio 0, 2
RXEN1# (default 0, receiver enabled)	XR_COM_RXEN0#	RS-232 #5 or RS-485/422 #1	exar_gpio 0, 3
485/232#_2# (default 1, RS-485)	XR_COM_MODE1	RS-232 #6 or RS-485/422 #2	exar_gpio 0, 4
TE485_2 (default 0, termination ON)	XR_COM_TE1	RS-232 #6 or RS-485/422 #2	exar_gpio 0, 5
DXEN2 (default 1, driver enabled)	XR_COM_DXEN1	RS-232 #6 or RS-485/422 #2	exar_gpio 0, 6
RXEN2# (default 0, receiver enabled)	XR_COM_RXEN1#	RS-232 #6 or RS-485/422 #2	exar_gpio 0, 7

For detailed information on the process, refer to the documentation on chip LTC2872.

Defining the I2C bus number

All devices linked to one of the I2C buses of the platform will be assigned a number when the platform boots. For configuration and operation purposes, the number assigned to the SMBus and the kempld must be known.

Step_1	From the OS CLI, run the following command to determine the SMBus number. LocalServer_OSPrompt:~# **echo "SMBUS is on $(i2cdetect -l | grep "SMBus I801" | cut -f 1 | cut -d - -f 2)"** **NOTE:** The answer will be SMBUS is on [SMBUS_NO].
Step_2	Run the following command to determine the kempld number. LocalServer_OSPrompt:~# **echo "KEMPLD is on $(i2cdetect -l | grep "i2c-kempld" | cut -f 1 | cut -d - -f 2)"** **NOTE:** The answer will be KEMPLD is on [KEMPLD_NO].

Configuring the specific ports as RS- 232

Access the OS and open a CLI. Refer to Accessing the operating system of a server for access instructions.

Enabling RS-232 #5

Use the following command:
LocalServer_OSPrompt:~# **sudo g pioset exar_gpio0 0=0 2=1 3=0**

As an option, flow control can be enabled:
LocalServer_OSPrompt:~# **stty -F /dev/ttyS4 crtscts**

Enabling RS-232 #6

Use the following command:
LocalServer_OSPrompt:~# **sudo g pioset exar_gpio0 4=0 6=1 7=0**

As an option, flow control can be enabled:
LocalServer_OSPrompt:~# **stty -F /dev/ttyS5 crtscts**

Configuring the specific ports as RS-485 half-duplex

NOTE: Support of RS-485 half-duplex with auto RS-485 output buffer enabled requires a specific driver. This driver is included in the BSP (Installing the board support package). It must be installed only if you intend to use this functionality.
Access the OS and open a CLI. Refer to Accessing the operating system of a server for access instructions.

Configuring RS-485 #1 and RS-485 #2 as half-duplex

To use this command, RS-485 #1 and RS-485 #2 must be enabled.
To find the value of the [SMBUS_NO] variable, refer to Defining the I2C bus number .
Use the following command:
LocalServer_OSPrompt:~# **sudo gpioset pca9539-[SMBUS_NO]-0074 10=1**

Configuring RS-485 #1 as half-duplex

Use the following commands:
LocalServer_OSPrompt:~# **sudo** gpioset exar_gpio0 0=1
LocalServer_OSPrompt:~# **stty -F /dev/ttyS5 crtscts**
Followed by the following BSP command:
LocalServer_OSPrompt:~# **sudo s1901-rs485-half-duplex.sh ttyS4**

Configuring RS-485 #2 as half-duplex

Use the following commands:
LocalServer_OSPrompt:~# **sudo** gpioset exar_gpio0 4=1
LocalServer_OSPrompt:~# **stty -F /dev/ttyS5 crtscts**
Followed by the following BSP command:
LocalServer_OSPrompt:~# **sudo s1901-rs485-half-duplex.sh ttyS5**

Configuring the specific ports as RS-485 full-duplex

NOTE: RS-485 full-duplex is used as an RS-422.
Access the OS and open a CLI. Refer to Accessing the operating system of a server for access instructions.

Configuring RS-485 #1 and RS-485 #2 as full-duplex

To use this command, RS-485 #1 and RS-485 #2 must be enabled.
To find the value of the [SMBUS_NO] variable, refer to Defining the I2C bus number .
Use the following command:
LocalServer_OSPrompt:~# **sudo gpioset pca9539-[SMBUS_NO]-0074 10=0**

Enabling RS-485 #1

Use the following command:
LocalServer_OSPrompt:~# **sudo** gpioset exar_gpio0 0=1 2=1 3=1
Followed by the following BSP command:
LocalServer_OSPrompt:~# **sudo s1901-rs485-full-duplex.sh ttyS4**

Enabling RS-485 #2

Use the following command:
LocalServer_OSPrompt:~# **sudo** gpioset exar_gpio0 4=1 6=1 7=1
Followed by the following BSP command:
LocalServer_OSPrompt:~# **sudo s1901-rs485-full-duplex.sh ttyS5**

Enabling or disabling the internal chip receiver termination on RS-485/422 ports

In an S1901 platform, the internal chip receiver termination of the following ports can be enabled or disabled:

- RS-485/422 #1
- RS-485/422 #2
- RS-485/422 #3
- RS-485/422 #4

Access the OS and open a CLI. Refer to Accessing the operating system of a server for access instructions.
In the commands, [VALUE] is 1 to enable and 0 to disable.

Enabling or disabling the receiver termination on RS-485 #1

Use the following command:
LocalServer_OSPrompt:~# **sudo** gpioset exar_gpio0 1=[VALUE]

Enabling or disabling the receiver termination on RS-485 #2

Use the following command:
LocalServer_OSPrompt:~# **sudo** gpioset exar_gpio0 5= [VALUE]

Enabling or disabling the receiver termination on RS-485 #3

Use the following command:
LocalServer_OSPrompt:~# **sudo** gpioset exar_gpio0 8= [VALUE]

Enabling or disabling the receiver termination on RS-485 #4

Use the following command:
LocalServer_OSPrompt:~# **sudo gpioset exar_gpio0 10= [VALUE]**

Configuring the AURIX MCU

Refer to AURIX MCU demo code for information on how to configure the CAN bus mezzanine with the AURIX safety MCU .

Configuring synchronization

Table of contents

Relevant sections:

Linux devices
Connector pinouts for building custom cables (for information on the SMA labeled PPS)
Installing the board support package
Accessing the switch network operating system
Accessing the operating system of a server

> **Changes to the switch NOS configuration are not persistent** after rebooting the switch NOS.
> To preserve configurations, the current configuration needs to be saved to startup-config.
> From the switch NOS Web UI:
> - Select **Maintenance** , **Configuration** and then **Save startup-config** . Click on **Save Configuration** to confirm the change.
>
> From the switch NOS CLI:
> - LocalSwitchNOS_OSPrompt:~(config-if)# **end**
> - LocalSwitchNOS_OSPrompt:~# **copy running-config startup-config**

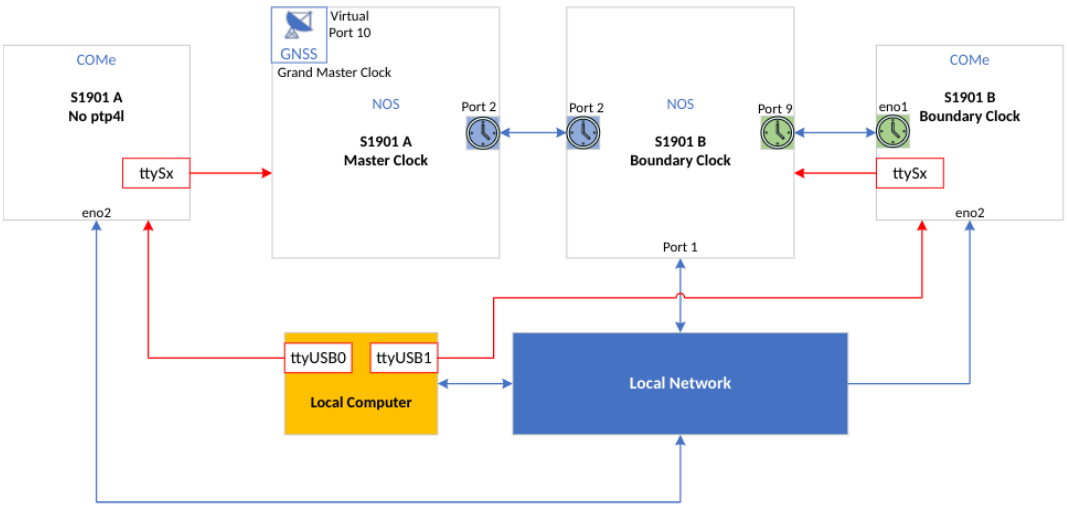
Platform synchronization must be configured for all components to communicate effectively. There are 2 possible configurations for the NOS PTP:

- When the NOS PTP is configured as a grandmaster clock using the GNSS, the GNSS is used to obtain phase synchronization and time of day (ToD) and the SMA labeled PPS will be automatically configured as a 1PPS output.
- When the NOS PTP is not configured as a grandmaster clock, the source for ToD and phase synchronization will be external to the NOS.

This section will describe how to configure synchronization for the NOS PTP as a grandmaster or an ordinary boundary clock. A validation step is provided and requires 2 platforms, a local computer and a local network:

- Platform A – NOS configured as a grandmaster
- Platform B – NOS configured as a boundary clock and COMe configured as a boundary clock with Linux PTP

The image below illustrates the test setup.



The last step of both grandmaster configuration procedures is a validation step to confirm proper configuration. These steps do not confirm the intended service will be provided by the grandmaster.

- Last step of the grandmaster procedure using the switch Web UI
- Last step of the grandmaster procedure using the switch CLI

To validate the grandmaster service is provided as intended, perform the validation step (last step) of the selected boundary clock configuration procedure. Make sure this validation step is performed using the test setup shown above.

- Last step of the boundary clock procedure using the switch Web UI
- Last step of the boundary clock procedure using the switch CLI

Also, to validate the boundary clock of NOS B provides the service as intended to the boundary clock with Linux PTP of the COMe, perform the procedure described in section Validating the service provided by the NOS B boundary clock below.

Configuring the switch NOS PTP as a grandmaster clock using the GNSS

When a GNSS is used, the PPS connector must be configured as an output. This is the default platform configuration.

Prerequisites when using a GNSS

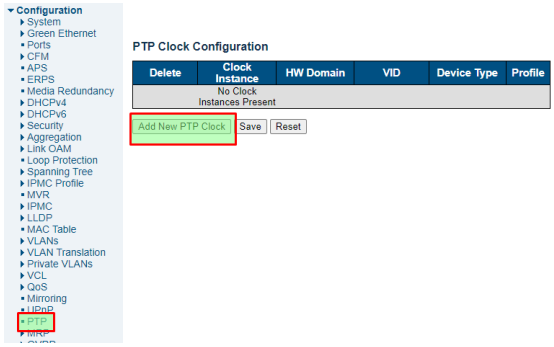
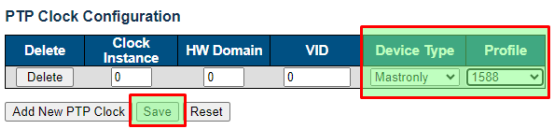
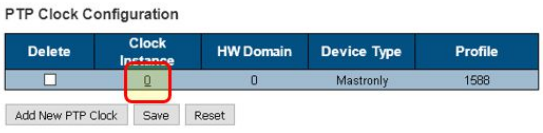
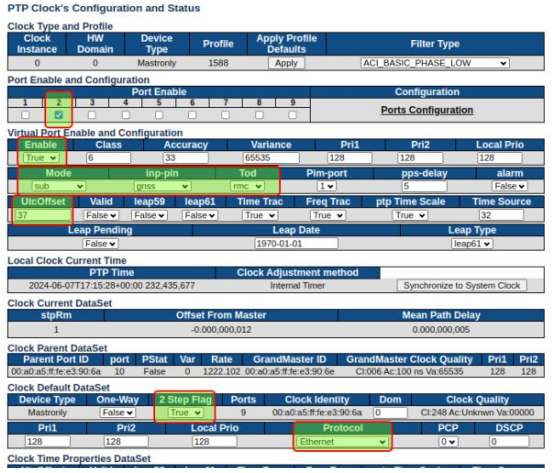
1	The GNSS serial port must be configured at a baud rate of 115200.
2	The switch NOS serial port must be configured at a baud rate of 115200.
3	The cable antenna delay should be configured if highly accurate timing (nanoseconds) is required.

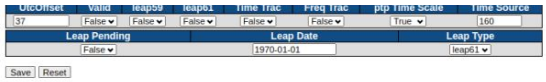
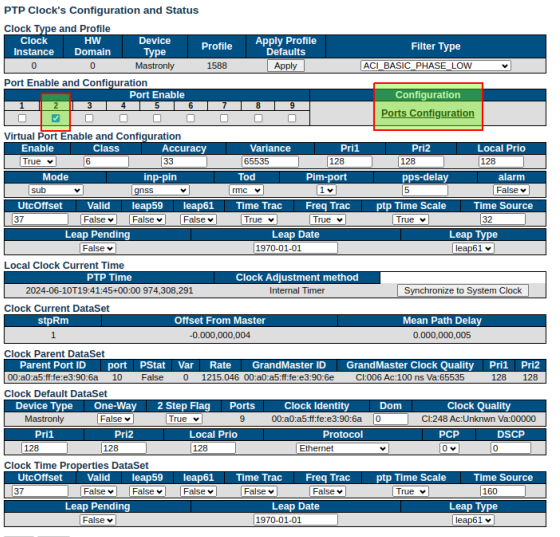
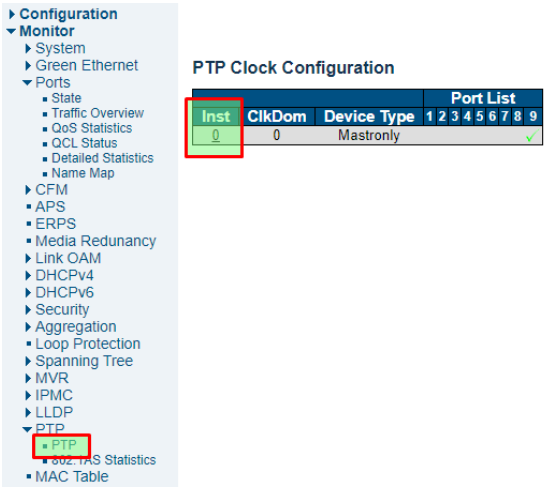
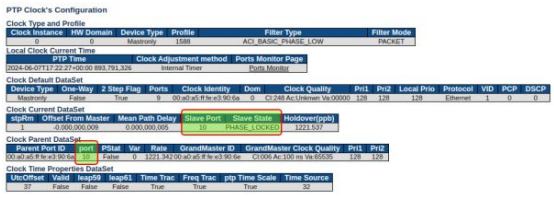
Relevant sections:
Configuring the GNSS (to configure the GNSS serial port and the cable antenna delay)
Configuring switch NOS networking (to configure the switch NOS serial port)

The switch NOS can be configured as a grandmaster clock using the switch NOS:

- Web UI
- CLI

Configuring the switch NOS PTP as a grandmaster clock using the switch NOS Web UI

Step_1	From the left-side menu, select **Configuration** and then **PTP** . Click on **Add New PTP Clock** .	
Step_2	From the **Device Type** drop-down list, select **Mastronly** . From the **Profile** drop-down list, select **1588** . Click on **Save** .	
Step_3	Under **Clock Instance** , click on the **0** . This will open the configuration page.	
Step_4	Proceed with configuration as follows (they are shown in the green boxes): 1. Under **Port Enable and Configuration** : 1. Check the box under the ports on which the PTP service must be enabled. In this examp le, it will be enabled on port 2 of NOS A. 2. Under **Clock Default DataSet** : 1. From the **2 Step Flag** drop-down list, select **True** . 2. From the **Protocol** drop-down list, select **Ethernet** . 3. Click on **Save** . The configuration shown in the green boxes under **Virtual Port Enable and Configuration** should already be done. The virtual port used is the one configured to enable serial and PPS communication between the GNSS and the switch NOS: • Field **Enable** is set to **True** – virtual ports are used, they will have numbers above the actual maximum port number (e.g. 9+1=10) • Field **Mode** is set to **sub** – the SMA pin is a PPS output • Field **inp-pin** is set to **gnss** – PPS from the GNSS is used as a reference for phase synchronization	

	• Field **Tod** is set to **rmc** – RMC format is used for the NMEA data of the GNSS • Field **UtcOffset** is set to **37** seconds	
Step_5	(Optional) Open the port configuration page. Under **Port Enable and Configuration/Configuration** , click on **Ports Configuration** to open the PTP service configuration page (for port 2 in this example).	
Step_6	(Optional) Configure PTP service per port interface. Click on **Save.** **NOTE:** For information on the parameters available, click on the question mark icon in the switch NOS Web UI to access the network switch help feature.	
Step_7	(Optional) To make the change persistent, save running-config to startup-config.	
Step_8	Access the PTP page to confirm PTP 0 configuration. From the left-side menu, select **Monitor** , **PTP** and **PTP** again. Select the PTP 0 instance by clicking on the **0** under **Inst** .	
Step_9	Check the configuration of PTP 0. **NOTE:** The final desired **Slave State** status is **PHASE_LOCKED** . Interim steps can be displayed: **FREQ_LOCKING** , **FREQ_LOCKED** and **HOLDOVER** . The time to reach **PHASE_LOCKED** varies depending on many factors. As a reference, less than 5 minutes is typical. **NOTE:** If the status is **FREERUN** , check the GNSS configuration. **NOTE:** The **Slave Port** and **port** fields highlighted in green are indicating port 10, which is the virtual port. To confirm the switch NOS PTP is the grandmaster clock: • Ensure the **Parent Port ID** (**Clock Parent DataSet** section) and the **Clock Identity** (**Clock Default DataSet** section) are the same. AND • Check the Class in the **GrandMaster Clock Quality** parameters (**Clock Parent DataSet** section): Cl:006 Class 006 (Locked with Primary Reference Clock, i.e. the GNSS in this configuration).	

Configuring the switch NOS PTP as a grandmaster clock using the switch NOS CLI

Step_1	Enter configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	
Step_2	Create the PTP clock instance **0** . Then add the desired interface(s) to **ptp 0** , the clock instance created. LocalSwitchNOS_Prompt(config)# **ptp 0 mode master twostep ethernet twoway id [CLOCK ID] vid [VLAN ID] 0 profile [PROFILE ID] mep 1** LocalSwitchNOS_Prompt(config)# **ptp 0 filter-type aci-basic-phase-low** LocalSwitchNOS_Prompt(config)# **ptp 0 time-property utc-offset 37 ptptimescale time-source 160** LocalSwitchNOS_Prompt(config)# **ptp 0 virtual-port** LocalSwitchNOS_Prompt(config)# **ptp 0 virtual-port time-property utc-offset 37 time-traceable freq-traceable ptptimescale time-source 32** Where: **[CLOCK ID]** is created according to the following rule: [First 6 digits of the NOS MAC address]:FF:FE:[Last 6 digits of the NOS MAC address]. **[PROFILE ID]** can be 802.1as or ieee1588.	
Step_3	Add the PTP service to the appropriate interfaces. In this example, it will be added to port 2 of NOS A. LocalSwitchNOS_OSPrompt:~# **interface Ethernet [PORT ID]** LocalSwitchNOS_OSPrompt:~# **ptp 0** LocalSwitchNOS_OSPrompt:~# **ptp 0 announce interval 1 timeout 3** LocalSwitchNOS_OSPrompt:~# **ptp 0 sync-interval 0** LocalSwitchNOS_OSPrompt:~# **ptp 0 delay-mechanism [DLM]** LocalSwitchNOS_OSPrompt:~# **ptp 0 delay-req interval 0** LocalSwitchNOS_OSPrompt:~# **ptp 0 delay-asymmetry 0** LocalSwitchNOS_OSPrompt:~# **ptp 0 ingress-latency 0** LocalSwitchNOS_OSPrompt:~# **ptp 0 egress-latency 0** **NOTE:** The **[PORT ID]** parameter can be used to configure all ports simultaneously: • **1/1-9** for all ports from 1 to 9 • **1/1-4,9** for ports 1, 2, 3, 4 and 9 **NOTE:** The **[DLM]** parameter is the port delay mechanism and is optional: • **e2e** (default) – e nd-to-end delay measurement • **p2p** – peer-to-peer delay measurement • **common-p2p** – common peer-to-peer delay measurement used in 802.1AS	
Step_4	End configuration. LocalSwitchNOS_OSPrompt:~# **end**	
Step_5	(Optional) To make the change persistent, save running-config to startup-config.	
Step_6	Verify the current PTP 0 status. LocalSwitchNOS_OSPrompt:~# **show ptp 0 parent** The desired clock class (under GrandmasterClockQuality) is 6 (Cl:006): • 6 – the local clock is using a time reference from a GNSS receiver • 7 – the local clock is in holdover after losing its time reference from the local GNSS receiver, but for no more than 10 minutes • 165 – the local clock is configured as a boundary clock in holdover; the boundary clock was previously locked to a grandmaster clock with a clock class of 6 • 248 – the local clock is configured as boundary clock	

Configuring the switch NOS PTP as an o rdinary boundary c lock

A boundary clock is a multiport network device with an essential role in hierarchical synchronization architectures. It synchronizes to the reference time on one port (usually from a grandmaster clock) and serves time on one or more other ports. Boundary clocks can be both a destination and a source of synchronization information.

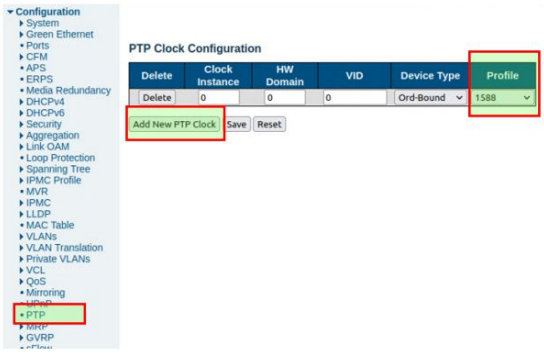
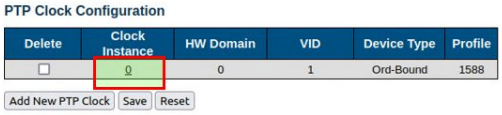
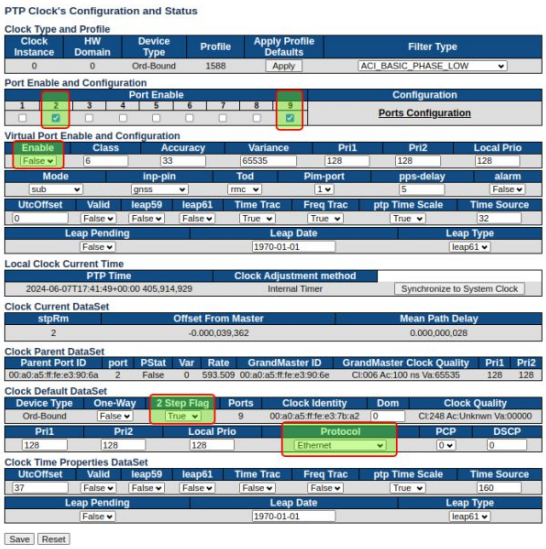
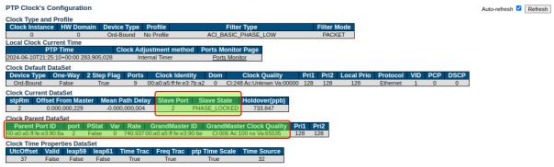
Prerequisites

1	An grandmaster must be connected to the platform.

The switch NOS can be configured as an ordinary boundary clock using the switch NOS:

- Web UI
- CLI

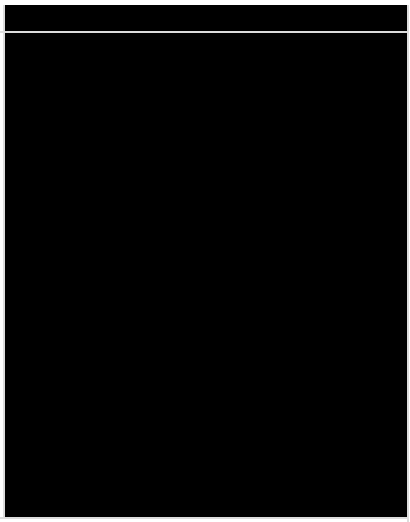
Configuring the switch NOS PTP as an o rdinary boundary c lock using the switch NOS Web UI

Step_1	Create the clock. 1. From the left-side menu, select **Configuration** and then **PTP** . 2. Click on **Add New PTP Clock** and fill the following fields: 1. **Clock Instance:** 0 2. **HW Domain :** 0 3. **VID:** 0 4. **Device Type:** Ord-Bound 5. **Profile:** 1588 3. Click on **Save** .	
Step_2	Click on the **Clock Instance** number to access the **PTP Clock's Configuration and Status** page.	
Step_3	This includes interfaces connected to the potential network grandmaster as well as interfaces connected to downstream slave clocks (boundary clock or slave clock). Ports will assume master or slave mode automatically. Proceed with configuration as follows (they are shown in the green boxes): 1. Under **Port Enable and Configuration** : 1. Check all the boxes for ports where you want to have PTP. In this example, port 2 is connected to the grandmaster (platform A in the block diagram above), and port 9 is connected to the COMe, which will also be configured as an ordinary boundary clock. 2. Under **Virtual Port Enable and Configuration** 1. From the **Enable** drop-down list, select **False** so the platform does not use the internal GNSS. 3. Under **Clock Default DataSet** : 1. From the **2 Step Flag** drop-down list, select **True** . 2. From the **Protocol** drop-down list, select **Ethernet** . 4. Click on **Save** .	
Step_4	Verify the configuration is working properly. From the left-side menu, select **Monitor, PTP** and **PTP** again . Under **Inst** , click on the **0** . This will open the configuration page.	
Step_5	Check the configuration. To follow the configuration evolution, check the Auto-refresh b ox in the upper right corner. Under **Clock Current DataSet** , ensure that the **Slave State** is in the desired state (i.e. **PHASE_LOCKED**). **NOTE:** The final desired **Slave State** status is **PHASE_LOCKED** . Interim steps can be displayed: **FREERUN** , **FREQ_LOCKING** , **FREQ_LOCKED** and **HOLDOVER** . The time to reach **PHASE_LOCKED** varies depending on many factors. As a reference, less than 5 minutes is typical. Interesting fields: • **Offset From Master** is the time difference between this clock and the	

	grandmaster clock. Last digit is nanoseconds. In this example, the platform is 5 ns in advance. The lower this number, the better. • **Mean Path Delay** is an average time to travel from the grandmaster clock to this clock. It is also displayed in nanoseconds • **Slave Port** field indicates from which port of the platform the PTP packets are coming. In the **Clock Parent DataSet** section, the **Parent Port ID** , **GrandMaster ID** , and **port** values indicate where the timing is coming from. In this example, it is coming out of port 2 of the remote platform. In the **Clock Parent DataSet** section, check the **GrandMaster Clock Quality** parameters: • Cl:006 Class 006 (Synchronized to a Primary Reference Clock - Atomic or GPS clock). The lower this number, the better. • Ac:100 ns Accuracy within 100 ns • Va:65536 Variance (0xFFFF = Value not computed)	
Step_6	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring the switch NOS as an o rdinary boundary c lock with the IEEE 1588 profile using the switch NOS CLI

Step_1	Enter configuration mode. LocalSwitchNOS_OSPrompt:~# **configure terminal**	[redacted]
Step_2	Create the PTP clock instance **0** . Then add the desired interface(s) to **ptp 0** , the clock instance created. LocalSwitchNOS_Prompt(config)# **ptp 0 mode boundary twostep ethernet twoway id [CLOCK ID] vid [VLAN ID] 0 profile [PROFILE ID] mep 1** LocalSwitchNOS_Prompt(config)# **ptp 0 filter-type aci-basic-phase-low** LocalSwitchNOS_Prompt(config)# **ptp 0 time-property utc-offset 37 ptptimescale time-source 160** Where: **[CLOCK ID]** is created according to the following rule: [First 6 digits of the NOS MAC address]:FF:FE:[Last 6 digits of the NOS MAC address]. **[VLAN ID]** value is 1 to 65535 or 1 if no VLAN is used **[PROFILE ID]** can be 802.1as or ieee1588. **NOTE:** The **utc-offset** value changes periodically and should correspond to the current value.	[redacted]
Step_3	Add the PTP service to the appropriate interfaces. In this example, it will be added to port 2 of the NOS (slave port to platform B) and to port 9 (COMe). For information on the parameters available, log in the switch NOS Web UI and click on the question mark icon to access the network switch help feature. LocalSwitchNOS_OSPrompt:~# **interface Ethernet [PORT ID]** LocalSwitchNOS_OSPrompt:~# **ptp 0** LocalSwitchNOS_OSPrompt:~# **ptp 0 announce interval 1 timeout 3** LocalSwitchNOS_OSPrompt:~# **ptp 0 sync-interval 0** LocalSwitchNOS_OSPrompt:~# **ptp 0 delay-mechanism [DLM]** LocalSwitchNOS_OSPrompt:~# **ptp 0 delay-req interval 0** LocalSwitchNOS_OSPrompt:~# **ptp 0 delay-asymmetry 0** LocalSwitchNOS_OSPrompt:~# **ptp 0 ingress-latency 0** LocalSwitchNOS_OSPrompt:~# **ptp 0 egress-latency 0** **NOTE:** The **[PORT ID]** parameter can be used to configure all ports simultaneously: • **1/1-9** for all ports from 1 to 9 • **1/1-4,9** for ports 1, 2, 3, 4 and 9 **NOTE:** The **[DLM]** parameter is the port delay mechanism and is optional: • **e2e** (default) – e nd-to-end delay measurement • **p2p** – peer-to-peer delay measurement • **cp2p** – common peer-to-peer delay measurement used in 802.1AS (useful over many PTP domains)	[redacted]
Step_4	End configuration.	[redacted]

	LocalSwitchNOS_OSPrompt:~# **end**	
Step_5	Verify the current PTP 0 status. LocalSwitchNOS_OSPrompt:~# **show ptp 0 default**	
Step_6	(Optional) To make the change persistent, save running-config to startup-config.	

Validating the service provided by the NOS B boundary clock

Interpreting the ptp4l log

The following list explains some of the messages generated in the log of the next section.

Message	Definition
port 0: INITIALIZING to LISTENING on INIT_COMPLETE	ptp4l is listening to PTP messages on eno2 (GbE0).
port 1: INITIALIZING to LISTENING on INIT_COMPLETE	ptp4l is listening to PTP messages on eno1 (10G port between the switch NOS and the COMe).
port 1: new foreign master **[CLOCK ID]** -9 **NOTE: [CLOCK ID]** is created according to the following rule: [First 6 digits of the NOS MAC address]:FF:FE:[Last 6 digits of the NOS MAC address].	NOS clock has been detected. The **-9** parameter at the end of the message means port 9 of the switch NOS.
Selected best master clock **[CLOCK ID]** **NOTE: [CLOCK ID]** is created according to the following rule: [First 6 digits of the GNSS MAC address]:FF:FE:[Last 6 digits of the GNSS port MAC address].	Defines which clock has been selected. **NOTE:** The clock ID starts and ends with the MAC address displayed in the NOS CLI +4 (GNSS port MAC address).
Running in a temporal vortex	The NOS grandmaster clock is reporting a UTC offset that is not accurate. **NOTE:** The **Utcoffset** parameter needs to be changed to a valid value. As of December 31, 2023, the UTC offset is 37 seconds.
port 1: LISTENING to UNCALIBRATED on RS_SLAVE	The local PTP is not yet synchronized with the grandmaster.
port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED	ptp4l on the COMe has successfully synchronized with a NOS PTP grandmaster clock.
Master offset [COLUMN1] [COLUMN2] [COLUMN3] [COLUMN4]	**COLUMN1** : Offset from the master (in nanoseconds). **COLUMN2** : Clock Servo State • s0 = unlocked • s1 = step (clock change by step) • s2 = locked – **this is the desired state** –the clock will be adjusted slowly based on the pi_offset_const option, with no step changes **COLUMN3** : Clock frequency adjustment (in parts per billion, ppb). **COLUMN4** : Estimated delay for the synchronization messages sent from the master (in nanoseconds).

Validation procedure

Prerequisite

1	The ptp4linux component included in the BSP provided by Kontron must be installed.

Procedure

Step_1	Confirm that the state of the network interface with the NOS is **UP** . LocalServer_OSPrompt:~# **ip a show eno1**	[redacted]
Step_2	Look at the PTP traffic between the switch NOS and the COMe (eno1) to confirm communication is established. The status should be **s2** for successful connectivity. LocalServer_OSPrompt:~# **sudo ptp4l -H -i eno1 -s -m -l 6 -E -2** Where: **-H** is the hardware time stamp **-i eno1** is the 10G interface (port 1 is eno1 and port 0 is eno2) **-s** is used to force the ptp4l in slave mode only **-m** is used to output messages on screen instead of in a log in /var/log/syslog **-l 6** sets the log level at 6 **-E** assumes the 1588 is using the delay request-response (end-to-end) mechanism **-2** assumes the 1588 is set to use Ethernet for PTP messages	[redacted]

Troubleshooting

If you want to see what is happening on the network, use tshark as described below.

Step_1	Open a second console to run tshark.	
Step_2	Install tshark to sniff the network. LocalServer_OSPrompt:~# **sudo apt install -y tshark**	
Step_3	Confirm connectivity is functioning as expected. LocalServer_OSPrompt:~# **sudo tshark -n -i eno1 -Y 'ptp'** The following message sequence indicates the two-step message exchange is working as expected: **[IP of NOS configured as grandmaster to multicast IP]** Sync Message **[IP of NOS configured as grandmaster to multicast IP]** Follow_Up Message **[IP of COMe configured as slave to multicast IP]** Delay_Req Message **[IP of NOS configured as grandmaster to multicast IP]** Delay_Resp Message	

Configuring and managing users

Table of contents

Configuring switch NOS users

Changes to the switch NOS configuration are not persistent after rebooting the switch NOS.
To preserve configurations, the current configuration needs to be saved to startup-config.
From the switch NOS Web UI:

- Select **Maintenance** , **Configuration** and then **Save startup-config** . Click on **Save Configuration** to confirm the change.

From the switch NOS CLI:

- LocalSwitchNOS_OSPrompt:~(config-if)# **end**
- LocalSwitchNOS_OSPrompt:~# **copy running-config startup-config**

Configuring switch NOS users using the switch NOS command-line interface

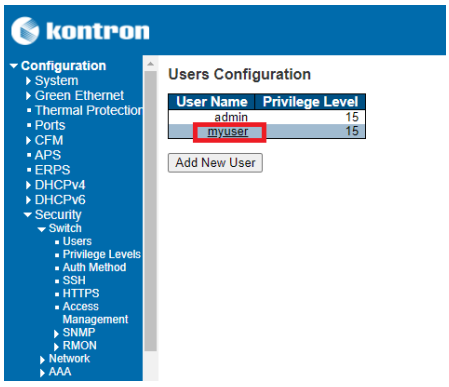
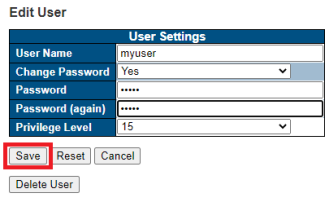
Refer to Accessing the switch network operating system for access instructions.
NOTE: If the switch NOS configuration is restored to default, the administrator password will be reset.

Step_1	Access the configuration setup menu. LocalSwitchNOS_OSPrompt:~# **configure terminal**	`# configure terminal`
Step_2	Configure the user. LocalSwitchNOS_OSPrompt:~(config)# **username [USERNAME] privilege [PRIVILEGE_LEVEL] password unencrypted [PASSWORD]** **NOTE:** The username is only used to identify the user and therefore can't be changed.	`(config)# username user privilege 15 password unencrypted newPassword`
Step_3	(Optional) To make the change persistent, save running-config to startup-config.	

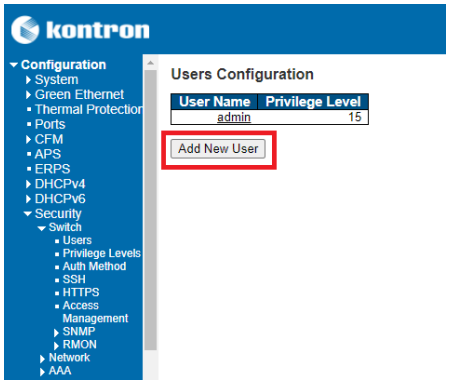
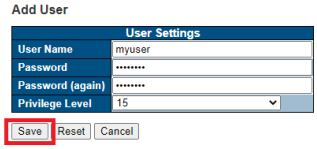
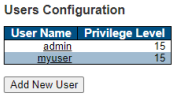
Configuring switch NOS users using the switch NOS Web UI

Refer to Accessing the switch NOS using the Web UI for access instructions.

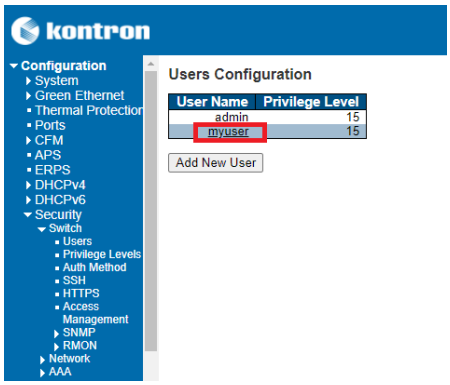
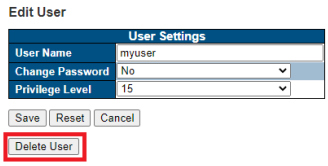
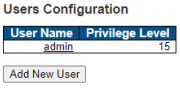
Changing the password of a user

Step_1	From the left-side menu, select **Configuration** , **Security** , **Switch** and then **Users** .	
Step_2	Click on the desired user.	
Step_3	Change the value of the **Change Password** dropdown menu to **Yes** .	
Step_4	Enter the password in fields **Password** and **Password (again)** .	
Step_5	Click on **Save** to confirm.	
Step_6	(Optional) To make the change persistent, save running-config to startup-config.	

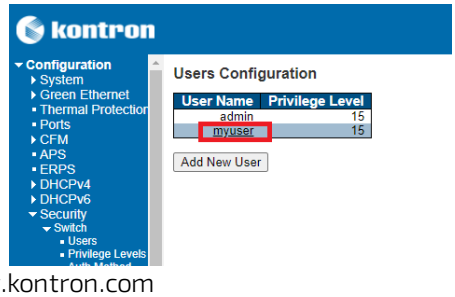
Adding a user

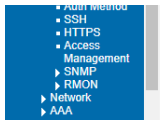
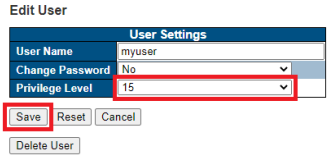
Step_1	From the left-side menu, select **Configuration** , **Security** , **Switch** and then **Users** .	
Step_2	Click on the **Add New User** button.	
Step_3	Fill the required fields: **User Name** , **Password** , **Password (again)** and **Privilege Level** . **NOTE:** For more information on the different privilege levels, click on the help button located at the top-right corner of the switch NOS Web UI page.	
Step_4	Click on the **Save** button to add the user.	
Step_5	A new user should be displayed in the user list.	
Step_6	(Optional) To make the change persistent, save running-config to startup-config.	

Deleting a user

Step_1	From the left-side menu, select **Configuration** , **Security** , **Switch** and then **Users** .	
Step_2	Click on the desired user.	
Step_3	Click on the **Delete User** button.	
Step_4	The user should be removed from the user list.	
Step_5	(Optional) To make the change persistent, save running-config to startup-config.	

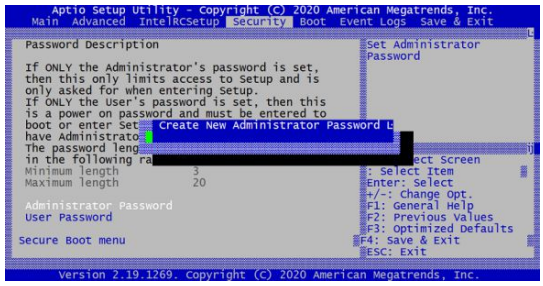
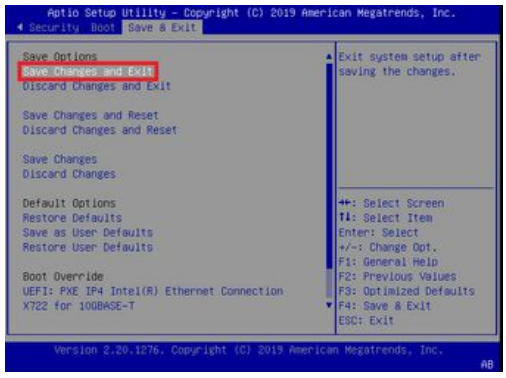
Configuring privilege level

Step_1	From the left-side menu, select **Configuration** , **Security** , **Switch** and then **Users** .	
Step_2	Click on the desired user.	

		
Step_3	Change the privilege level using the dedicated dropdown menu. **NOTE:** For more information on the different privilege levels, click on the help button located at the top-right corner of the switch NOS Web UI page.	
Step_4	Click on the **Save** button to confirm.	
Step_5	(Optional) To make the change persistent, save running-config to startup-config.	

Configuring UEFI/BIOS users

Refer to Accessing the UEFI BIOS for access instructions.
The UEFI/BIOS can only have two users: Administrator and User, which can both be protected with a password.
If a password is required, a password has to be set for Administrator before one can be set for User. This means there can be a password for Administrator alone or one for both Administrator and User.
WARNING: If the Administrator password is lost, contact customer support for assistance.

Step_1	From the UEFI/BIOS setup menu, navigate to the **Security** menu and select **Administrator Password** or **User Password** .	
Step_2	Set a password and press **Enter** .	
Step_3	Confirm the password by entering it again and press **Enter** .	
Step_4	Navigate to the **Save & Exit** menu, go to **Save Changes and Exit** and press **Enter** to confirm.	

Configuring OS users

Access the OS. Refer to Accessing the operating system of a server for access instructions.

Step_1	Proceed with configuration as recommended in the OS documentation.

Typical commands in Linux

Adding a user	useradd [OPTIONS] USERNAME
Removing a user	userdel [OPTIONS] USERNAME

For more information, search for "User and Group Management in Linux".

Configuring software handling ignition key switch

Table of contents

Ignition key switch behavior

The behavior of the Ignition Key Switch (IKS) can be configured based on application and usage:

- Full hardware management using an FPGA to control IKS behavior
  - The FPGA intercepts the IKS state change and generates a **Power Button event** in the COMe. This event initiates the shutdown mechanism in the OS. When the OS shutdown is completed, the COMe chipset sends a signal back to the FPGA (**payload power supply** signal (S5=OFF) and maintains only the **suspend power supply** signal (S3=ON). When this state is reached (S5=OFF and S3=ON), the FPGA powers off the PSU. The IKS hardware management includes a configurable timeout that can be used when the OS is either not loaded or not able to properly shutdown.
- Full software management of the behavior using a customer polling application associated with the IKS state
  - The IKS software management is activated by setting the **KeySoft** bit in the FPGA register. An OS software must then be used to read the state of the IKS and to use the OS command to perform the shutdown.

The AURIX MCU is aware of the IKS state. When the IKS is set in software mode on Linux, the demo code of the AURIX MCU continues to be aware of its status and includes a function (called reboot) to execute a short press of the power button (same action as the IKS).

	If the platform shuts down or reboots, all configurations made in the FPGA (e.g. software or hardware management, timeout delay) return to their default values.

Typical IKS actions are as follows:

1. Setting the IKS to the ON position
2. Setting the IKS to the OFF position

There are also exceptional actions that could be performed:

1. Power off before the OS is ready and able to receive the shutdown signal. When this happens, the PSU will be turned OFF after the timeout set in the FPGA has elapsed.
2. Power off while the OS is ready and a program prevents shutdown. When this happens, the PSU will be turned OFF after the timeout set in the FPGA has elapsed.
3. From a state when the OS is ready, rapid toggle of the IKS from OFF position to ON position. When this happens, the platform will reboot. If you want to shut it down, turn the IKS to the OFF position and wait for the LED to turn off.

Relevant section:
AURIX MCU demo code (for information on the reboot and keyswitch commands)

Prerequisites for IKS configuration

1	The following software tool is installed: • i2c-tools
2	The FPGA version must be higher than 1.02.08002246.
3	Access to the OS CLI.
4	Users must be connected as root. LocalServer_OSPrompt:~# **sudo su**

Relevant section:
Accessing the operating system of a server using a physical connection

Defining the I2C bus number

All devices linked to one of the I2C buses of the platform will be assigned a number when the platform boots. For configuration and operation purposes, the number assigned to the SMBus and the kempld must be known.

Step_1	From the OS CLI, run the following command to determine the SMBus number. LocalServer_OSPrompt:~# **echo "SMBUS is on $(i2cdetect -l | grep "SMBus I801" | cut -f 1 | cut -d - -f 2)"** **NOTE:** The answer will be SMBUS is on [SMBUS_NO].
Step_2	Run the following command to determine the kempld number. LocalServer_OSPrompt:~# **echo "KEMPLD is on $(i2cdetect -l | grep "i2c-kempld" | cut -f 1 | cut -d - -f 2)"** **NOTE:** The answer will be KEMPLD is on [KEMPLD_NO].

Checking the FPGA version

Step_1	U se the following commands to determine the FPGA version . LocalServer_OSPrompt:~# **i2cdump -y [SMBUS_NO] 0x55 i | grep 00\: | awk '{print $17 "." $16 "." $15 $14 $13 $12 }'**	[redacted]

Selecting the ignition key switch management method

	If the platform shuts down or reboots, all configurations made in the FPGA (e.g. software or hardware management, timeout delay) return to their default values.

There are two management methods:

- Hardware management
- Software management

FPGA registers used for management method selection

The KeyOvrST bit is used to set the management method.

FPGA Registers		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0x02	Read	Reserved	Reserved	Reserved	Reserved	KeySt	KeyOvrSt	KeySoft	Reserved
	Write						KeyOvrSt	KeySoft	
	Init						0	0	

Managing using hardware

This is the default state configured on all platforms delivered with an OS.

Step_1	Use the following commands to use hardware management. LocalServer_OSPrompt:~# **i2cset -y [SMBUS_NO] 0x55 2 "$(($(i2cget -y [SMBUS_NO] 0x55 2) & 0xFD))"**

Refer to Configuring the full hardware IKS behavior to proceed with configuration.

Managing using software

Step_1	Use the following commands to use software management. LocalServer_OSPrompt:~# **i2cset -y [SMBUS_NO] 0x55 2 "$(($(i2cget -y [SMBUS_NO] 0x55 2) | 0x02))"**

Refer to Configuring the full software IKS behavior to proceed with configuration.

Confirming the management method configured

Step_1	Use the following commands to confirm the configuration. LocalServer_OSPrompt:~# **i2cget -y [SMBUS_NO] 0x55 2 | awk '{print "Key handled by " (and(strtonum($1),0x02) ? "Software": "Hardware")}'**

Configuring the full hardware IKS behavior

This is the default state configured on all platforms delivered with an OS. If the OS is reinstalled or an OS was not provided, a script must be added to the OS for the IKS to work properly (refer to Adding a script in the OS to program the IKS behavior). The hardware management is associated with a timeout configurable in the FPGA.

The default behavior when using hardware management is as follows:

- When the platform is not powered:
  - The IKS is set to the ON position
  - The PSU is powered
  - The power LED turns on
  - The platform is powered
  - The OS booting process occurs
- When the platform is powered:
  - The IKS is set to the OFF position
  - The OS shuts down

- The PSU and power LED turn off

Adding a script in the OS to program the IKS behavior

For the typical behavior to work properly, a script must be included in the OS. This script is typically included by Kontron, but it is good practice to check for its presence. Should you required to include it, here is the procedure.

Step_1	Send a command to create a file. LocalServer_OSPrompt:~# **vi /etc/acpi/events/power**
Step_2	Edit the file created with the following content. **event=button/power** **action=/usr/bin/logger "ACPI_POWER_BTTN: rebooting"** **action=/sbin/reboot**
Step_3	Exit the file by pressing the **Esc** key followed by **:q** .
Step_4	Activate the new script for the power event. LocalServer_OSPrompt:~# **service acpid restart**

Setting the ignition key switch timeout in the FPGA

If the platform shuts down or reboots, all configurations made in the FPGA (e.g. software or hardware management, timeout delay) return to their default values.

As shown in the table below, the default timeout value is 16 seconds. Note that it is recommended that the timeout be set to a value lower than the expected booting time.

FPGA Registers		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0x07	Read	Shutdown timeout: 0 to 255 sec							
	Write	Shutdown timeout: 0 to 255 sec							
	Init	16 sec							

Step_1	Set the timeout. LocalServer_OSPrompt:~# **i2cset -y [SMBUS_NO] 0x55 7 60** **NOTE:** The command above sets the timeout to 60 seconds. If the timeout is set to 0, the function will generate an immediate power off.

Configuring the full software IKS behavior

An OS software must be used:

- To read the **KeySoft** bit to determine the IKS state
- To write to the **KeyOvrSt** to turn off the PSU

The final step for software management is to activate the **KeyOvrSt** bit. This will turn off the PSU provided the IKS is physically set to the OFF position.

Reading the KeySoft state

Step_1	Use the following commands to read the KeySoft state. LocalServer_OSPrompt:~# **i2cget -y [SMBUS_NO] 0x55 2 | awk '{print "Ignition Key Switch pin = " (and(strtonum($1),0x08) ? "ON (Vbat+)": "OFF (Vbat-)")}'**

Reading the KeyOvrSt state

Step_1	Use the following commands to read the KeyOvrSt state. LocalServer_OSPrompt:~# **i2cget -y [SMBUS_NO] 0x55 2 | awk '{print "Key override state = " (and(strtonum($1),0x04) ? "ON": "OFF")}'**

Setting the KeyOvrSt bit to 1 to turn off the PSU

Step_1	Use the following commands to set the KeyOvrSt to 1. LocalServer_OSPrompt:~# **i2cset -y [SMBUS_NO] 0x55 2 "$(($(i2cget -y [SMBUS_NO] 0x55 2) & 0xFB))"**

Setting the KeyOvrSt bit to 0 to ignore the IKS request until the software finishes the shutdown process

Step_1	Use the following commands to set the KeyOvrSt to 0 . LocalServer_OSPrompt:~# **i2cset -y [SMBUS_NO] 0x55 2 "$(($(i2cget -y [SMBUS_NO] 0x55 2) | 0x04))"**

Operating

Platform power management

Table of contents

For the ignition key switch to operate as designed, the behavior of the power button must be properly configured. If the OS provided by Kontron is not used, ensure the configuration described in section Enabling the ignition key switch has been done before operating the platform.

General considerations

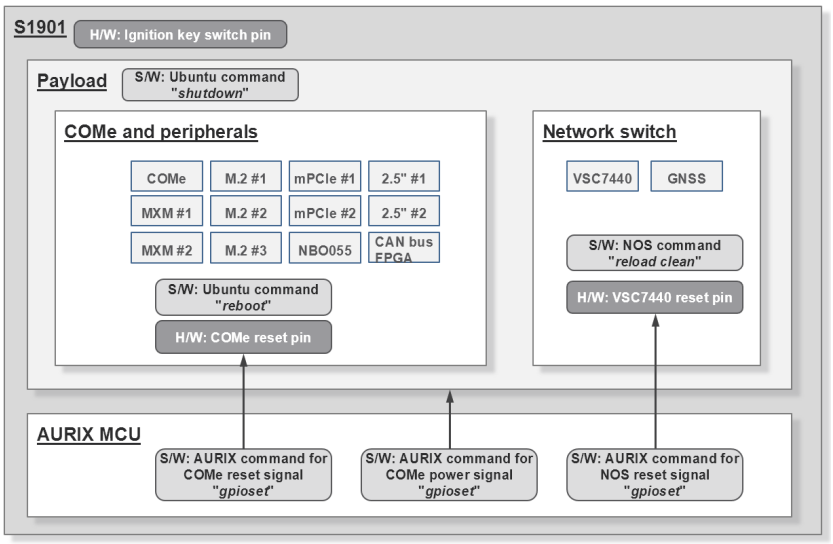
The platform has 3 power states:

- Completely powered OFF (power supply cable is unplugged)
- 10 mA (less than 10 mA are consumed; only the ignition key switch is powered)
- Powered ON (all devices are powered)

The platform is designed to be powered using the ignition key switch (IKS) signal. The IKS simulates the "Power ON signal" provided by the power button. The power button and the reset button are available on the J1 connector for lab use only. In a deployment, the platform is controlled by the IKS only.

T he power block diagram of the platform is shown below.

- S1091 – power for the platform
- Payload – power for everything in the platform, except for t he AURIX MCU (when installed)
- COMe and peripherals – power for the COMe and all its related peripherals
- Network switch – power for the NOS and the GNSS
- AURIX MCU – power for the AURIX MCU



Information on the platform power block diagram

S1901

Payload

When a shutdown command is sent through the LocalServer_OSPrompt , the COMe and its peripherals as well as the network switch are powered OFF. This command does not affect the AURIX MCU (when installed).
NOTE: To prevent users getting locked out of the platform, it is designed to restart when a software shutdown command is sent. If a proper power OFF of the platform is desired, the IKS must be used.

COMe and peripherals

When a reboot command is sent through the LocalServer_OSPrompt, the COMe and its peripherals are reset . This command does not affect the network switch and the AURIX MCU (when installed).

Network switch

When a reload clean command is sent through the LocalSwitchNOS_OSPrompt, the network switch will reboot. This command does not affect the COMe and its peripherals and the AURIX MCU (when installed).

AURIX MCU

A custom software program loaded into the AURIX MCU can use an I/O to change the state of the platform COMe RESET . Two actions can be triggered:

- A pulse to high – a regular reset will occur on the COMe and its peripherals
- Maintain to high – the COMe and its peripherals will be held in reset, meaning they will be disabled

A custom software program loaded into the AURIX MCU can use an I/O to change the state of the platform COMe PWRBTN . Two actions can be triggered:

- A short press to high – the power sequence triggered by a short press on a power button will occur, causing the components within the payload power block to power OFF accordingly
- A long press to high – the power sequence triggered by a long press on a power button will occur, causing the components within the payload power block to power OFF accordingly

A custom software program loaded into the AURIX MCU can use an I/O to change the state of the network switch NOS RESET. Two actions can be triggered:

- A pulse to low – a regular reset will occur on the network switch
- Maintain to low – the network switch will be held in reset, meaning it will be disabled

Relevant sections:
Power consumption and power budget
Connector pinouts for building custom cables
Kontron test cables
Accessing the operating system of a server (to access the OS CLI)
Accessing the switch network operating system (to access the network switch CLI or Web UI)
Configuring software handling ignition key switch
Disabling the network switch via the AURIX MCU (to reset or deactivate the NOS)
Disabling the platform COMe via the AURIX MCU (to reset or deactivate the COMe)

Summary of available power commands

The following options are available:

- Powering ON
  - From OFF mode
- Going to OFF mode
  - From powered ON state
- Rebooting the COMe
  - Using the OS CLI
  - Using software programmed in the AURIX MCU
- Resetting the network switch
  - Using the network switch CLI
  - Using the network switch Web UI
  - Using software programmed in the AURIX MCU
- Removing power

Powering ON

The platform can be powered ON:

- From OFF mode

NOTE: The Power LED is OFF until the IKS is set to Vbat+.

Powering ON from OFF mode

This action is performed using the ignition key switch.

Electrical prerequisites

1	The power supply cable must be plugged in.
2	The ignition key switch input pin is not connected.

Procedure

Step_1	Connect the ignition key switch input pin to Vbat+. Resulting actions: • PSU: Powers ON • COMe: Powers ON • Network switch: Powers ON • Power LED: Turns ON • COMe serial port (console): Becomes available • AURIX MCU: Powers ON (if installed)

Going to OFF mode

When the platform is in OFF mode, it consumes less than 10 mA. This power is used only to read the ignition key switch state.

The platform can go to OFF mode:

- From powered ON state

Going to OFF mode from powered ON state

This action is performed using the ignition key switch.

Electrical prerequisites

1	The power supply cable must be plugged in.
2	The platform is in powered ON state.
3	The ignition key switch input pin is connected to Vbat+.

Procedure

Step_1	Connect the ignition key switch input pin to Vbat- or disconnect it. The carrier board FPGA instructs the OS to shutdown. If the shutdown was not completed within one minute, the platform will still go to OFF mode. Resulting actions: • PSU: Goes to OFF mode • COMe: Powers OFF completely • Network switch: Powers OFF completely • Power LED: Turns OFF • COMe serial port (console): Becomes unavailable • AURIX MCU: Powers OFF completely (if installed) NOTE: Command **linux /etc/systemd/logind.conf HandlePowerKey= ignore** has no effect, the platform will go to OFF mode after one minute. NOTE: A software ignition key switch bypass exists, refer to the appropriate configuration page listed in the Relevant sections above for more information.

Rebooting the COMe

The COMe can be rebooted using:
- The OS CLI
- Software programmed in the AURIX MCU

Rebooting the COMe using the OS CLI

Electrical prerequisites

1	The power supply cable must be plugged in.
2	The ignition key switch input pin is connected to Vbat+.

Procedure

Step_1	Access the OS CLI.
Step_2	Use the following command to reboot the COMe through a clean shutdown of the OS. LocalServer_OSPrompt:~# **sudo reboot** Resulting actions: • PSU: Remains ON • COMe: Powers OFF and then powers ON • Network switch: Remains ON • Power LED: Remains ON • COMe serial port (console): Remains available (shows the shutdown sequence, displays nothing for one minute and then shows the UEFI/BIOS information) • AURIX MCU: Remains ON (if installed)

Rebooting the COMe using software programmed in the AURIX MCU

Electrical prerequisites

1	The power supply cable must be plugged in.
2	The ignition key switch input pin is connected to Vbat+.

Procedure

Refer to the Relevant section above and go to Disabling the platform COMe via the AURIX MCU to implement software.

Resetting the network switch

NOTE: Make sure all changes to the configuration are saved prior to rebooting the switch NOS.

The network switch can be reset using:
- The network switch CLI
- The network switch Web UI
- Software programmed in the AURIX MCU

The only resulting action will be a switch reset.

Resetting the network switch using the network switch CLI

Electrical prerequisites

1	The power supply cable must be plugged in.
2	The ignition key switch input pin is connected to Vbat+.

Procedure

Access the network switch CLI.

Step_1	LocalSwitchNOS_OSPrompt:~# **reload cold** **NOTE:** Rebooting the switch NOS may take several seconds.

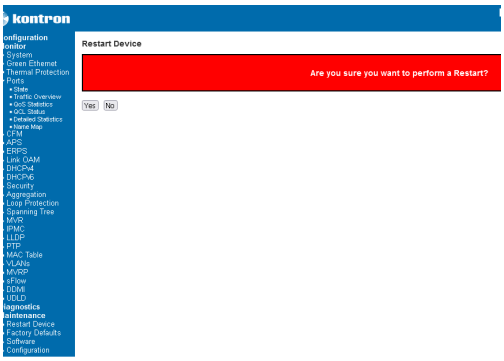
Resetting the network switch using the network switch Web UI

Electrical prerequisites

1	The power supply cable must be plugged in.
2	The ignition key switch input pin is connected to Vbat+.

Procedure

Access the network switch Web UI.

Step_1	From the left-side menu, select **Maintenance** and then **Restart Device** .	

Resetting the network switch using software programmed in the AURIX MCU

Electrical prerequisites

1	The power supply cable must be plugged in.
2	The ignition key switch input pin is connected to Vbat+.

Procedure

Refer to the Relevant section above and go to Disabling the network switch via the AURIX MCU to implement software.

Removing power

Power can be completely removed from the platform by unplugging the power supply cable.

Controlling LEDs

Table of contents

Prereq uisites

1	Access to the OS is required.
2	All relevant BSP components are installed.

Relevant sections:
Connector pinouts for building custom cables
Installing the board support package
Accessing the operating system of a server

Device used to control LEDs

A **PCA9539** I/O chip provides most of the GPIOs.
There are 10 GPOs (pins) on the PCA9539 chip for controlling the 4 LEDs on the front panel (L1, L2, STATUS and POWER):

- L1 and L2 are bi-color LEDs (red/green) with infrared
- STATUS is a blue LED with infrared
- POWER is a green LED with infrared

Defining the I2C bus number

All devices linked to one of the I2C buses of the platform will be assigned a number when the platform boots. For configuration and operation purposes, the number assigned to the SMBus and the kempld must be known.

Step_1	From the OS CLI, run the following command to determine the SMBus number. LocalServer_OSPrompt:~# **echo "SMBUS is on $(i2cdetect -l | grep "SMBus I801" | cut -f 1 | cut -d - -f 2)"** **NOTE:** The answer will be SMBUS is on [SMBUS_NO].
Step_2	Run the following command to determine the kempld number. LocalServer_OSPrompt:~# **echo "KEMPLD is on $(i2cdetect -l | grep "i2c-kempld" | cut -f 1 | cut -d - -f 2)"** **NOTE:** The answer will be KEMPLD is on [KEMPLD_NO].

Writing a value to a LED-related GPO

Step_1	From the platform's operating system, use the following command to write a value to a GPO. LocalServer_OSPrompt:~# **gpioset --mode=[MODE] [GPIO_CHIP_NAME] [GPIO_PIN]=[VALUE]** Where: [MODE] = wait [GPIO_CHIP_NAME] = pca9539-[SMBUS_NO]-0074 [GPIO_PIN] = see table below [VALUE] = see table below

Commands for controlling the LEDs

The power LED indicates that power is available at the power supply output to supply platform components. It will not be ON when power is supplied and the platform is in 10 mA state . On this platform, the power LED will turn ON by default . To use the night vision mode, it needs to be set to OFF.

LED ID	PCA9539 pin number [GPIO_PIN]	Description [VALUE]	Command
L1	0	0 = Turn ON red	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 0=0`
		1 = Turn OFF red	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 0=1`
	1	0 = Turn ON green	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 1=0`
		1 = Turn OFF green	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 1=1`
	11	0 = Turn ON IR	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 11=0`
		1 = Turn OFF IR	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 11=1`
L2	2	0 = Turn ON red	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 2=0`
		1 = Turn OFF red	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 2=1`
	3	0 = Turn ON green	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 3=0`
		1 = Turn OFF green	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 3=1`
	12	0 = Turn ON IR	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 12=0`
		1 = Turn OFF IR	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 12=1`
STATUS	4	0 = Turn ON blue	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 4=0`
		1 = Turn OFF blue	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 4=1`
	14	0 = Turn ON IR	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 14=1`
		1 = Turn OFF IR	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 14=0`
POWER	5	1 = Turn ON green	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 5=1`
		0 = Turn OFF green	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 5=0`
	13	0 = Turn ON IR	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 13=0`
		1 = Turn OFF IR	`gpioset --mode=wait pca9539-[SMBUS_NO]-0074 13=1`

Controlling GPIOs

Table of contents

Prerequisites

1	Access to the OS is required.
2	All relevant BSP components are installed.

Relevant sections:

Accessing the operating system of a server
Connector pinouts for building custom cables (for information on GPIO electrical characteristics)
Kontron test cables
Grounding (for information on electrical connections)
Installing the board support package
Configuring the GPIOs of the AURIX MCU (for information on the internal GPIOs of the AURIX MCU)

Description of platform GPIOs

I/O type	Can be configured as	Note
GPI #2, #3, #4, #5, #6, #7, #8, #9	GPI only	It is not possible to write to these GPIs.
GPI #1 (direct to COMe)	GPI only	It is not possible to write to this GPI.
GPO #1 (direct to COMe)	GPO only	It is not possible to read this GPO.
GPO #2, #3, #4, #5, #6, #7, #8	GPO only	It is not possible to read these GPOs.
GPIO #1, #2, #3, #4, #5, #6, #7	GPO or GPI	If you read a GPIO, it will become a GPI. If you write to a GPIO, it will become a GPO.

Detecting GPIO devices

There are 6 GPIO controller devices:

- tic12400, mcp23s17-0, mcp23s17-1 and gpio-kempld that provide user-configurable GPIOs on the J1 and J2 connectors
- pca9539-[SMBUS_NO]-0074 that provides LED control
- pca9539-[KEMPLD_NO]-0077 that provides internal GPIO control

Defining the I2C bus number

All devices linked to one of the I2C buses of the platform will be assigned a number when the platform boots. For configuration and operation purposes, the number assigned to the SMBus and the kempld must be known.

Step_1	From the OS CLI, run the following command to determine the SMBus number. LocalServer_OSPrompt:~# **echo "SMBUS is on $(i2cdetect -l | grep "SMBus I801" | cut -f 1 | cut -d - -f 2)"** **NOTE:** The answer will be SMBUS is on [SMBUS_NO].
Step_2	Run the following command to determine the kempld number. LocalServer_OSPrompt:~# **echo "KEMPLD is on $(i2cdetect -l | grep "i2c-kempld" | cut -f 1 | cut -d - -f 2)"** **NOTE:** The answer will be KEMPLD is on [KEMPLD_NO].

Detecting GPIOs

Administrative privileges (sudo) are required.

Step_1	From the platform's operating system, use the following command to list the GPIO controller devices present on the platform. LocalServer_OSPrompt:~# **gpiodetect** NOTE: The controller's name is in the second column and is the one to use because the controller ID mapping (name in the first column) could vary. It is good practice to use the controller's name instead of its ID.

Devices

Device name [GPIO_CHIP_NAME]	Description	GPIOs provided

pca9539-[KEMPLD_NO]-0077	A **pca9539** I/O chip provides internal GPIO	• Control of various devices, including:
		  ◦ Ethernet switch reset
		  ◦ M.2 WWAN disable

pca9539-[SMBUS_NO]-0074	A **pca9539** I/O chip provides LED control	• 10 GPOs for Controlling LEDs
gpio-kempld	The **gpio-kempld** I/O chip is directly connected to the COMe carrier board	• 1 GPI going to J1 • 1 GPO going to J1
tic12400	0-36 VDC inputs	• 8 GPIs going to J1 and J2
mcp23s17-0	Open drain outputs up to 100 mA	• 7 GPOs going to J1 and J2
mcp23s17-1	0-5 VDC when set as a GPI Push-pull when set as a GPO • Maximum of 25 mA per output and maximum of 125 mA for all outputs	• 7 GPIOs going to J2

Reading the value of a GPI

Step_1	From the platform's operating system, use the following command to read the value of a GPI. LocalServer_OSPrompt:~# **s1901-gpioget tic12400 [GPIO_PIN]** **or** LocalServer_OSPrompt:~# **gpioget gpio-kempld [GPIO_PIN]** Where: [GPIO_PIN] = see tables below

Writing a value to a GPO

Step_1	From the platform's operating system, use the following command to write a value to a GPO. LocalServer_OSPrompt:~# **s1901-gpioset mcp23s17-0 [GPIO_PIN]** **or** LocalServer_OSPrompt:~# **gpioset --mode=[MODE] [GPIO_CHIP_NAME] [GPIO_PIN]=[VALUE]** Where: [MODE] = wait [GPIO_CHIP_NAME] = gpio-kempld or pca9539-[SMBUS_NO]-0074 [GPIO_PIN] = see tables below [VALUE] = 0 (low) or 1 (high)

Configuring a GPIO

Configuring a GPIO as a GPI

Step_1	From the platform's operating system, use the following command to read the value of a GPIO. This will configure it as a GPI. LocalServer_OSPrompt:~# **s1901-gpioget mcp23s17-1 [GPIO_PIN]** **or** LocalServer_OSPrompt:~# **gpioget [GPIO_CHIP_NAME] [GPIO_PIN]** Where: [GPIO_CHIP_NAME] = gpio-kempld or pca9539-[KEMPLD_NO]-0077 [GPIO_PIN] = see tables below

Configuring a GPIO as a GPO

Step_1	From the platform's operating system, use the following command to write a value to a GPIO. This will configure it as a GPO. LocalServer_OSPrompt:~# **s1901-gpioset mcp23s17-1 [GPIO_PIN]=[VALUE]** **or** LocalServer_OSPrompt:~# **gpioset --mode=[MODE] [GPIO_CHIP_NAME] [GPIO_PIN]=[VALUE]** Where: [MODE] = wait [GPIO_CHIP_NAME] = gpio-kempld or pca9539-[KEMPLD_NO]-0077 [GPIO_PIN] = see tables below [VALUE] = 0 (low) or 1 (high)

GPIO mapping

J1 GPIO mapping

There are 6 GPIs and 5 GPOs in the J1 connector:

- 1 GPI and 1 GPO coming from the CPU
- 5 GPIs coming from tic12400
- 4 GPOs coming from mcp23s17-0

GPIs

Name used in the connector pinout section	J1 pin	P6 pin of the DB25 connector of the Kontron test cable for J1	tic12400 pin number [GPIO_PIN]	gpio-kempld pin number [GPIO_PIN]
Input 1 Isolated (COMe GPIO)	16	1		0
Input 2 Isolated	15	3	2	
Input 3 Isolated	8	5	3	
Input 4 Isolated	7	7	4	
Input 5 Isolated	2	9	5	
Input 6 Isolated	21	11	6	

GPOs

Name used in the connector pinout section	J1 pin	P6 pin of the DB25 connector of the Kontron test cable for J1	mcp23s17-0 pin number [GPIO_PIN]	gpio-kempld pin number [GPIO_PIN]
Output 1 Isolated (COMe GPIO)	22	2		4
Output 2 Isolated	30	4	2	
Output 3 Isolated	14	6	3	
Output 4 Isolated	9	8	4	
Output 5 Isolated	29	10	5	

J2 GPIO mapping

There are 3 GPIs, 3 GPOs and 7 GPIOs in the J2 connector:

- 3 GPIs coming from tic12400
- 3 GPOs coming from mcp23s17-0
- 7 GPIOs coming from mcp23s17-1

GPIs

Name used in the connector pinout section	J2 pin	P6 pin of the DB25 connector of the Kontron test cable for J2	tic12400 pin number [GPIO_PIN]
Input 7 Isolated	21	1	7
Input 8 Isolated	44	3	8
Input 9 Isolated	34	5	9

GPOs

Name used in the connector pinout section	J2 pin	P6 pin of the DB25 connector of the Kontron test cable for J2	mcp23s17-0 pin number [GPIO_PIN]
Output 6 Isolated	15	13	6
Output 7 Isolated	49	2	7
Output 8 Isolated	53	4	8

GPIOs

Name used in the connector pinout section	J2 pin	P6 pin of the DB25 connector of the Kontron test cable for J2	mcp23s17-1 pin number [GPIO_PIN]
GPIO 1 Isolated	26	6	1
GPIO 2 Isolated	19	8	2
GPIO 3 Isolated	41	10	3
GPIO 4 Isolated	20	12	4
GPIO 5 Isolated	12	7	5
GPIO 6 Isolated	35	9	6
GPIO 7 Isolated	42	11	7

Controlling CAN buses

Table of contents

Relevant section:
Configuring the AURIX MCU (to configure communication)

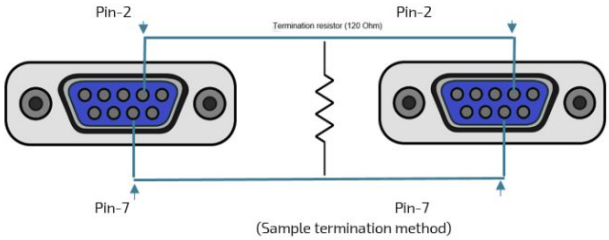
The control of CAN buses is different depending on the option selected for the platform:

- Four CAN buses
- AURIX ™ TC387 MCU with f our CAN buses

Controlling CAN buses when no AURIX MCU is installed

Overview

A Controller Area Network bus (CAN bus) is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other's applications without a host computer, so the separate electronic control units (ECUs) inside a vehicle could communicate with only a single pair of wires.
The customer OS uses the Socket CAN interface and the can-utils package to support control of the CAN bus.
The following demo shows the output of a basic loopback test using the CAN bus connection.
a) Using appropriate CAN bus termination, (physically) connect CAN0 and CAN1 together. Note that there is no end-of-line resistor installed inside the platform. Any end-of-line resistor must be connected externally.



Common applications

Refer to Accessing the operating system of a server for access instructions.

Verifying CAN interface presence

Step_1	4 or more CAN bus interfaces should be present within the platform depending on Model of system. Use the following command to verify their presence. LocalServer_OSPrompt:~# **ip a**	[redacted]

Enabling a CAN link

Step_1	Enable a CAN interface using the following command. LocalServer_OSPrompt:~# **ip link set [INTERFACE_NAME] up type can bitrate 1000000**

Disabling a CAN link

Step_1	Disable a CAN interface using the following command. LocalServer_OSPrompt:~# **ip link set [INTERFACE_NAME] down**

Verifying communication

The following section was documented using the following loopback configuration:

- can0 ↔ can1
- can2 ↔ can3

Sending and listening to messages between interfaces

In this example, two Bash shells are used.

	Sender shell	Receiver shell
Step_1		Listen for messages emitted on the CAN buses. LocalServer_OSPrompt:~$ **candump -x any**
Step_2	LocalServer_OSPrompt:~$ **ip link set can0 up type can bitrate 1000000**	
Step_3	LocalServer_OSPrompt:~$ **ip link set can1 up type can bitrate 1000000**	
Step_4	LocalServer_OSPrompt:~$ **ip link set can2 up type can bitrate 1000000**	
Step_5	LocalServer_OSPrompt:~$ **ip link set can3 up type can bitrate 1000000**	
Step_6	LocalServer_OSPrompt:~$ **cansend can0 2AA#1122334455667788**	can0 TX - - 2AA [8] 11 22 33 44 55 66 77 88 can1 RX - - 2AA [8] 11 22 33 44 55 66 77 88
Step_3	LocalServer_OSPrompt:~$ **cansend can1 2AA#1122334455667788**	can1 TX - - 2AA [8] 11 22 33 44 55 66 77 88 can0 RX - - 2AA [8] 11 22 33 44 55 66 77 88
Step_4	LocalServer_OSPrompt:~$ **cansend can2 2AA#1122334455667788**	can2 TX - - 2AA [8] 11 22 33 44 55 66 77 88 can3 RX - - 2AA [8] 11 22 33 44 55 66 77 88
Step_5	LocalServer_OSPrompt:~$ **cansend can3 2AA#1122334455667788 8**	can3 TX - - 2AA [8] 11 22 33 44 55 66 77 88 can2 RX - - 2AA [8] 11 22 33 44 55 66 77 88

Where:

- TX is the transmitter interface
- RX is the receiver interface

More information

The following links provide more information about SocketCAN and can-utils:

- https://www.kernel.org/doc/html/latest/networking/can.html
- https://github.com/linux-can/can-utils

Controlling CAN buses when an AURIX MCU is installed

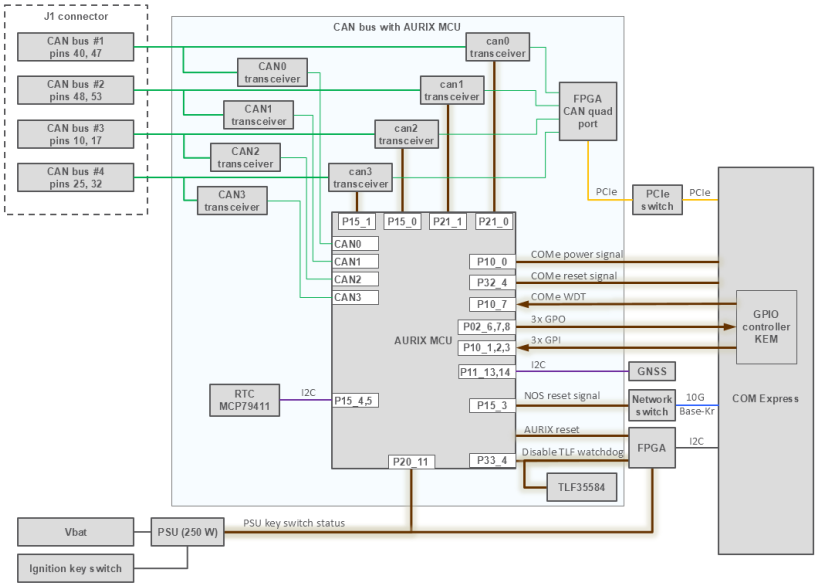
This command is used to enable, disable or view the status of a CAN bus transceiver.

Step_1	Enable or disable a specific module. Shell> **canenable [MODULE] [STATUS]** Where: [MODULE] is a number from 0 to 3 [STATUS] is **true** to enable the module or **false** to disable the module	

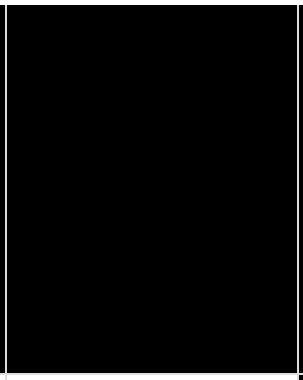
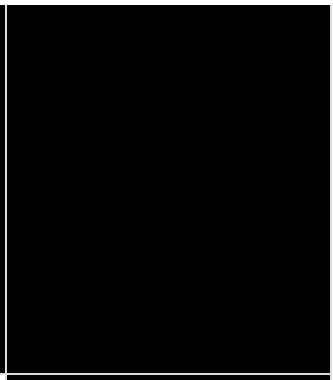
NOTE: Once a CAN bus transceiver is disabled using the AURIX MCU, the Linux interface will be set to DOWN under the Linux OS. To make it work again, start by enabling the transceiver with the AURIX MCU **canenable** command and then set the link UP under the Linux OS.

The table below provides an example of a sequence of commands that illustrate how to use the **canenable** command.
For this example:

- The table below shows what happens when CAN buses are enabled or disabled on each shell at each step.
- 3 shells are used (Linux sender shell, Linux receiver shell, AURIX MCU shell).
- In the 3 shells, the CAN buses do not have the same name.
  - The physical CAN bus #1 (J1 pins 48 and 53) is called:
    - /dev/can1 in Linux
    - CAN1 in the AURIX MCU
  - The physical CAN bus #2 (J1 pins 40 and 47) is called:
    - /dev/can2 in Linux
    - CAN2 in the AURIX MC U
- Connections are as follows:
  - An external connection is added from pin 40 to pin 48 and from pin 47 to pin 53. This connection links CAN bus #1 to CAN bus #2.
  - A 120-ohm end of line resistor is connected between CAN-High and CAN-Low.



Steps	Commands	Linux sender shell	Linux receiver shell	AURIX MCU shell
Step_1	On the Linux receiver shell, listen for messages emitted on the CAN buses.		[redacted]	
Step_2	On the Linux sender shell, enable can0.	[redacted]		
Step_3	On the Linux sender shell, enable can1.	[redacted]		
Step_4	On the Linux sender shell, send the following data on can0: 1234567890123456. In the Linux receiver shell, this data will be shown as transmitted (TX) for can0 and as received (RX) for can1. In the AURIX MCU shell, this data will be shown as received for both CAN0 and CAN1.	[redacted]	[redacted]	[redacted]
Step_5	On the AURIX MCU shell, disable the transceiver associated to can0. This will cut the communication for the Linux device on the can0 port. **NOTE:** This will set the can0 link to DOWN in the Linux OS.			[redacted]
Step_6	On the Linux sender shell, confirm the can0 link is set to DOWN.	[redacted]		
Step_7	On the Linux sender shell, send the following data on can0: 9876543210987654. In the Linux receiver shell, this data will only be shown as transmitted (TX) for can0. Because the can0 port is disabled, no data will be received by can1 and the AURIX MCU CAN0.	[redacted]	[redacted]	
Step_8	On the Linux sender shell,	[redacted]	[redacted]	[redacted]

	send the following data on can1: 0246813579024681. In the Linux receiver shell, this data will only be shown as transmitted (TX) for can1. Because the can0 port is disabled, no data will be received by can0. Since the data from can1 is transmitted, it will be shown as received for both CAN0 and CAN1 in the AURIX MCU shell.			
Step_9	On the AURIX MCU shell, enable the transceiver associated to can0. This will allow communication for the Linux device on can0 port. **NOTE:** This will not set the can0 link to UP in the Linux OS.			
Step_10	On the Linux sender shell, set the can0 link to DOWN to make sure a clean shutdown is performed. Then set the can0 link to UP.			
Step_11	On the Linux sender shell, confirm communication is reestablished.			

Controlling the AURIX MCU

Refer to Installing the AURIX MCU development environment and demo code for information on installation.
Refer to AURIX MCU demo code for information on how to control the AURIX MCU.

AURIX MCU demo code

Table of contents

This section describes the demo code of the AURIX MCU.

Clients are responsible for implementing cybersecurity functions in the AURIX MCU code that will be used in their platforms.
The demo code provided by Kontron does not include cybersecurity functions.

Overview

The AURIX M CU default firmware performs the following:

- Upon boot-up, CAN bus ports are automatically enabled. This ensures that the CAN bus mezzanine with the AURIX safety MCU provides the same functionalities as a CAN bus mezzanine.
- The RTC watchdog is disabled.
- The state of the 3 GPOs from the COMe are reproduced on the 3 GPIs of the COMe.
- A command-line interface (CLI) with a shell is provided on the AURIX MCU's UART. This CLI is used to enter commands available with the demo code.

Relevant sections:

Product architecture
Linux devices

Commands available with the demo code

Command	Description
canenable	Enable/disable a CAN transceiver
gpioconfig	Configure Px.y gpio as an input/output and its mode
gpioget	Show the state of Px.y gpio
gpioset	Set the state of Px.y gpio (possible states: 0/1)
help	Display command list and command help
keyswitch	Read/monitor the state of a key switch
nosreboot	Reboot the NOS
reboot	Reboot the system
status	Show the application status

Using the serial shell to interact with the AURIX MCU's UART

Setting up a serial connection to the device

Step_1	Open the Linux terminal and install minicom. LocalServer_OSPrompt:~# **sudo apt-get install minicom**
Step_2	Set up a serial connection to access the device through minicom. LocalServer_OSPrompt:~# **resize ; minicom -c on -w -D /dev/ttyS1**
Step_3	Press **Enter** once to start the shell. The following will be displayed: **Shell>** . **TIP:** To exit minicom when finished, enter **Ctrl-A** followed by **X** .

Using the AURIX MCU demo code commands

Canenable command

Relevant sections:
Product architecture
Controlling CAN buses

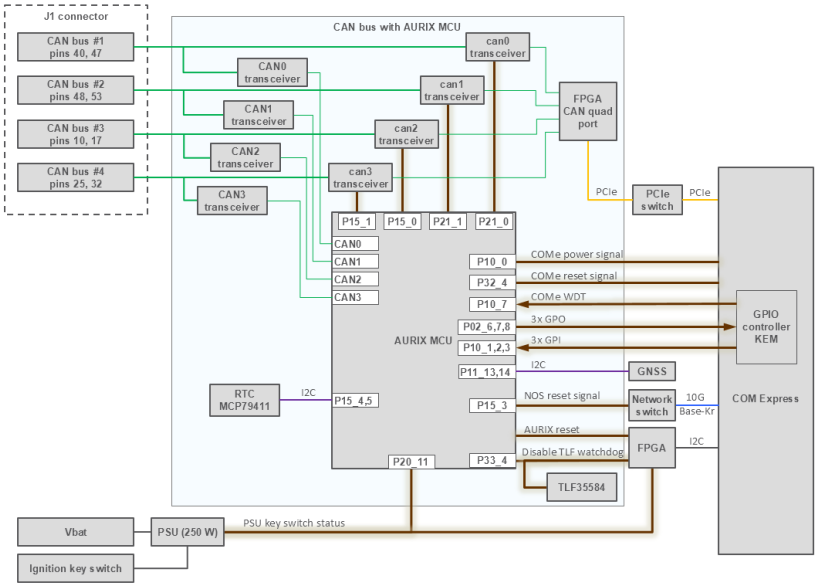
This command is used to enable, disable or view the status of a CAN bus transceiver.

Step_1	Enable or disable a specific module. Shell> **canenable [MODULE] [STATUS]** Where: [MODULE] is a number from 0 to 3 [STATUS] is **true** to enable the module or **false** to disable the module	[redacted]

NOTE: Once a CAN bus transceiver is disabled using the AURIX MCU, the Linux interface will be set to DOWN under the Linux OS. To make it work again, start by enabling the transceiver with the AURIX MCU **canenable** command and then set the link UP under the Linux OS.

The table below provides an example of a sequence of commands that illustrate how to use the **canenable** command.
For this example:

- The table below shows what happens when CAN buses are enabled or disabled on each shell at each step.
- 3 shells are used (Linux sender shell, Linux receiver shell, AURIX MCU shell).
- In the 3 shells, the CAN buses do not have the same name.
  - The physical CAN bus #1 (J1 pins 48 and 53) is called:
    - /dev/can1 in Linux
    - CAN1 in the AURIX MCU
  - The physical CAN bus #2 (J1 pins 40 and 47) is called:
    - /dev/can2 in Linux
    - CAN2 in the AURIX MC U
- Connections are as follows:
  - An external connection is added from pin 40 to pin 48 and from pin 47 to pin 53. This connection links CAN bus #1 to CAN bus #2.
  - A 120-ohm end of line resistor is connected between CAN-High and CAN-Low.



Steps	Commands	Linux sender shell	Linux receiver shell	AURIX MCU shell
Step_1	On the Linux receiver shell, listen for messages		[redacted]	

	emitted on the CAN buses.		[redacted]	
Step_2	On the Linux sender shell, enable can0.	[redacted]		
Step_3	On the Linux sender shell, enable can1.	[redacted]		
Step_4	On the Linux sender shell, send the following data on can0: 1234567890123456. In the Linux receiver shell, this data will be shown as transmitted (TX) for can0 and as received (RX) for can1. In the AURIX MCU shell, this data will be shown as received for both CAN0 and CAN1.	[redacted]	[redacted]	[redacted]
Step_5	On the AURIX MCU shell, disable the transceiver associated to can0. This will cut the communication for the Linux device on the can0 port. **NOTE:** This will set the can0 link to DOWN in the Linux OS.			[redacted]
Step_6	On the Linux sender shell, confirm the can0 link is set to DOWN.	[redacted]		
Step_7	On the Linux sender shell, send the following data on can0: 9876543210987654. In the Linux receiver shell, this data will only be shown as transmitted (TX) for can0. Because the can0 port is disabled, no data will be received by can1 and the AURIX MCU CAN0.	[redacted]	[redacted]	
Step_8	On the Linux sender shell, send the following data on can1: 0246813579024681. In the Linux receiver shell, this data will only be shown as transmitted (TX) for can1. Because the can0 port is disabled, no data will be received by can0. Since the data from can1 is transmitted, it will be shown as received for both CAN0 and CAN1 in the AURIX MCU shell.	[redacted]	[redacted]	[redacted]
Step_9	On the AURIX MCU shell, enable the transceiver associated to can0. This will allow communication for the Linux device on can0 port. **NOTE:** This will not set the can0 link to UP in the Linux OS.			[redacted]
Step_10	On the Linux sender shell, set the can0 link to DOWN to make sure a clean shutdown is performed. Then set the can0 link to	[redacted]		

	UP.	[redacted]		
Step_11	On the Linux sender shell, confirm communication is reestablished.	[redacted]	[redacted]	[redacted]

Gpioconfig command

Relevant sections:
Product architecture
Controlling GPIOs

The AURIX MCU provides the following ports:

AURIX MCU	Platform carrier board
IfxPort_P02_6, output	COMe input kempld #1
IfxPort_P02_7, output	COMe input kempld #2
IfxPort_P02_8, output	COMe input kempld #3
IfxPort_P10_1, input	COMe output kempld #5
IfxPort_P10_2, input	COMe output kempld #6
IfxPort_P10_3, input	COMe output kempld #7

Configuring a GPIO as an input

This command is used to configure a GPIO as an input.

Step_1	Configure a GPIO as an input. Shell> **gpioconfig [X] [Y] in [MODE]** Where: [X] is the port number [Y] is the pin number within the port (Optional) [MODE] Use **pu** if a pull-up resistor is connected or **pd** if a pull-down resistor is connected. No argument = no pull device connected	[redacted]

Configuring a GPIO as an output

This command is used to configure a GPIO as an output.

Step_1	Configure a GPIO as an output. Shell> **gpioconfig [X] [Y] out [MODE]** Where: [X] is the port number [Y] is the pin number within the port (optional) [MODE] Use **od** to enable open-drain output mode. No argument = push/pull output mode.	[redacted]

Gpioget command

i	GPIO pins must be configured as an input before using the gpioget command. Use the gpioconfig command to do so.

This command is used to display the state of a GPIO pin.

Step_1	Display the state. Shell> **gpioget [X] [Y]** Where: [X] is the port number [Y] is the pin number within the port	[redacted]

Gpioset command

	GPIO pins must be configured as an output before using the gpioset command. Use the gpioconfig command to do so.

This command is used to set the state of a GPIO pin .

Step_1	Set the state of a GPIO pin. Shell> **gpioset [X] [Y] [Z]** Where: [X] is the port number [Y] is the pin number within the port [Z] specifies the desired state of the pi n (0 for low or 1 for high)	

Help command

This command is used to list the commands available.

	To get detailed information about a specific command, add a **?** after the command name (e.g., Shell> **status ?**).

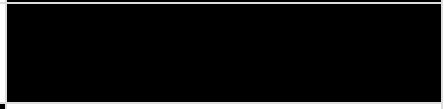
Step_1	List the commands available. Shell> **help**	

Keyswitch command

Relevant sections:
Configuring software handling ignition key switch
J1 connector pinout

	The Ignition Key Switch must be in software mode. If it is in the default hardware mo de, the shutdown sequence will be performed.

This command is used view the state of the Ignition Key Switch. This state is used by the AURIX MCU to determine whether it shuts down the platform or not.

Steps	Commands	AURIX MCU shell	Linux shell
Step_1	On the Linux shell, get the SMBUS number. **NOTE:** The answer will be SMBUS is on [SMBUS_NO].		
Step_2	On the Linux shell, set the Ignition Key Switch to use software management.		
Step_3	On the AURIX MCU shell, read the Ignition Key Switch KeySoft state.		
Step_4	Trigger the Ignition Key Switch by disconnecting pin 38 of connector J1 (Vbat+) from pin 37 of connector J1 (Ignition Key Switch input).		
Step_5	On the AURIX MCU shell, read the Ignition Key Switch KeySoft state. By confirming the state of the Ignition Key Switch, you can implement the logic required in the AURIX MCU to determine when the platform should be turned OFF.		

Nosreboot command

This command is used to reboot the network switch of the platform. It uses AURIX MCU GPIO P15.3 connected to the reset pin of the network switch. The **nosreboot** command sets P15.3 to zero for one second and sets it back to 1 to release the network switch reset pin. It includes all the steps for proper NOS reboot, compared to having to send unique commands successively, as shown in the procedure described in the next section.

Step_1	Reboot the NOS. Shell> **nosreboot**	[redacted]

Disabling the network switch via the AURIX MCU

The AURIX MCU GPIO P15.3 is connected to the NOS reset pin. Maintaining the AURIX MCU GPIO P15.3 to 0 disables the network switch.
The example below shows how to maintain GPIO P15.3 to 0 and how to reenable it.

Steps	Commands	AURIX MCU	Linux shell
Step_1	On the Linux shell, ping the IP of the network switch.		[redacted]
Step_2	On the AURIX MCU shell, configure GPIO P15.3 as an output.	[redacted]	
Step_3	On the AURIX MCU shell, s et GPIO P15.3 to 0 to disable the network switch. **NOTE:** This command disables the network switch. It will not automatically restart.	[redacted]	[redacted]
Step_4	On the AURIX MCU shell, s et GPIO P15.3 to 1 to release the reset pin and restart the network switch.	[redacted]	
Step_5	The network switch will reboot. This may take up to 2 minutes.		[redacted]

Reboot command

The AURIX MCU shell uses the serial port from the COMe. For that reason, if the COMe is shut down, serial communication is lost with the AURIX MCU shell.
The **reboot** command ensures the pin used to shut down the COMe is released to complete the reboot. This ensures serial communication with the AURIX MCU shell will be available again once the reboot sequence is complete.

This command is used to reboot the platform COMe. It uses the AURIX MCU GPIO P32.4 connected to the power signal pin of the COMe. The program sets GPIO P32.4 to 1 for one second and sets it back to 0. This is equivalent to performing a short press of the power button, triggering a shutdown in Ubuntu. Other functionalities could be implemented by clients using the reset and power buttons (see the section below for a brief explanation pertaining to the reset button).

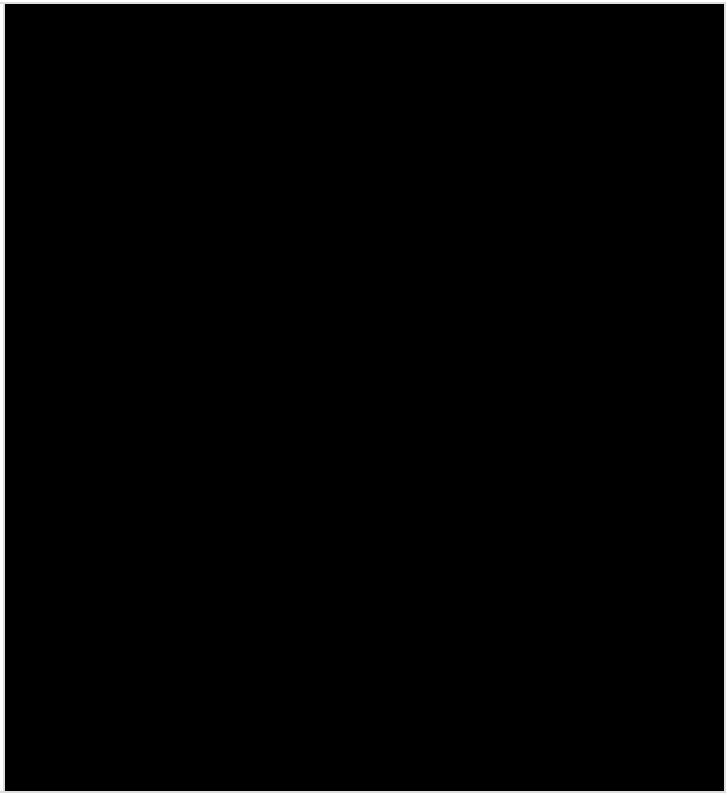
Steps	Commands	AURIX MCU shell
Step_1	Reboot the COMe. Shell> **reboot**	[redacted]

Disabling the platform COMe via the AURIX MCU

The AURIX MCU GPIO P10.0 is connected to the COMe reset pin. Maintaining the AURIX MCU GPIO P10.0 to 0 disables the platform COMe. The AURIX MCU shell uses the serial port of the COMe. For that reason, if the COMe is shut down, serial communication is lost with the shell.
A programmed command must be used to disable the COMe via GPIO P10.0. This program must implement a way to release the COMe reset to restart the COMe. The program could be similar to the reboot example of the source code (e.g. set GPIO pin to 1, wait one second and set GPIO pin to 0).

Status command

This command is used to view information about the application.

Step_1	View information about the application. Shell> **status**	

Description of the tasks executed by each CPU

CPU 0

- Disables the CPU watchdog
- Initializes the GPI/GPO pins to interact with COMe pins
- Enables CAN transceivers by default
- Waits for all other cores to be ready
- Initializes the AsclinShellInterface module for UART shell communication
- Initializes the RTC module
- In main while loop:
  - Computes current core load by reading CPU instruction count (access with serial command **status**)
  - Runs the AsclinShellInterface UART polling and command parsing

CPU 1

- Disables the CPU watchdog
- Initializes the TLF power switcher
- Tells the TLF to go in normal state after 60 seconds
- In main while loop:
  - Computes current core load by reading CPU instruction count (access with serial command **status**)
  - Continuously reproduces the state of GPO pins on GPI pins for demo purpose

CPU 2

- Disables the CPU watchdog
- Initializes voltage, temperature measurement and lifehold LED module
- Initializes the performance measurement module
- Initializes EVADC module for adc reading
- In main while loop:
  - Computes current core load by reading CPU instruction count (access with serial command **status**)

CPU 3

- Disables the CPU watchdog
- In main while loop:
  - Computes current core load by reading CPU instruction count (access with serial command **status**)

Source code

This section includes information to help with source code development.
Code organization is simple and most of it is in the src folder. The code has been adapted for the AURIX MCU from the Application Kit TC3X7 evaluation board.

Code organization

The code is organized as follows:

- Configurations
- Debug
- Lcf_Tasking_Tricore_Tc.lsl
- Libraries
- poc-safety-mezz.launch
- src
  - Tricore
    - Can
      * Can.c
      * CanCmds.c
      * CanCmds.h
      * Can.h
    - Cfg_Illd
      * Configuration.h
      * ConfigurationIsr.h
    - Cfg_Ssw
      * Ifx_Cfg_SswBmhd.c
      * Ifx_Cfg_Ssw.c
      * Ifx_Cfg_Ssw.h
    - Demo_Illd
      * Measurement.c
      * Measurement.h
      * Perf_Meas.c
      * Perf_Meas.h
    - Evadc
      * EvadcAutoScan.c
      * EvadcAutoScan.h
      * EvadcCmds.c
      * EvadcCmds.h
    - Gpio
      * GpioCmds.c
      * GpioCmds.h
    - KeySwitch
      * KeySwitchCmds.c
      * KeySwitchCmds.h
    - Main
      * Cpu0_Main.c
      * Cpu1_Main.c
      * Cpu2_Main.c
      * Cpu3_Main.c
    - Power
      * TLF3xx8x.c
      * TLF3xx8xCmds.c
      * TLF3xx8xCmds.h
      * TLF3xx8x.h
    - Rtc
      * Mcp79411.c
      * Rtc.h
      * RtcCmds.c
      * Rtc.Cmds.h
    - ScrCArray
      * CompileScrArray.mk
    - Shell
    - Power
      * AsclinShellInterface.c
      * AsclinShellInterface.h

Folder content

This section details the content of some folders.

src\Tricore\Cfg_Illd

This folder includes configuration files for the Interrupt Library module.
Configuration.h contains the definition of the hardware pin mapping and a description of which service will run on each CPU core.

src\Tricore\Main

This folder contains main files for different CPUs in the system.
All CPUs supported by the AURIX MCU have their main files. The main files of the cores are named: Cpu[X]_Main.c, where [X] is the CPU number starting with Cpu0. When calling external c file modules within a core make sure that this file has the #pragma header that specifies where the binary code will be located in the flash. This will allow the core to access it.

For an example, see #pragma header in line 64 of AsclinShellInterface.c.

src\Tricore\Gpio

The "GPIO Follower" demo software provided with the AURIX MCU reads the outputs of the COMe and changes the associated input of the COMe so it matches the states of the COMe output.
The logic implement is as follow:

```
Output P02_6 = Input P10_1
Output P02_7 = Input P10_2
Output P02_8 = Input P10_3
```

Controlling the Automotive Ethernet

Table of contents

Relevant sections:

Linux devices (for the value of the [LINUX_DEVICE] parameter)
Connector pinouts for building custom cables
Installing the board support package
Accessing the operating system of a server

Prerequisites

1	BSP version 17 or higher must be installed for M.2 automotive Ethernet support.

Verifying automotive Ethernet interface presence

Step_1	LocalServer_OSPrompt:~$ **ip addr | grep enp -A 1**	[redacted]

Enabling an a utomotive Ethernet link

NOTE: Command **ip up** must be used for **ethtool** to be operational.

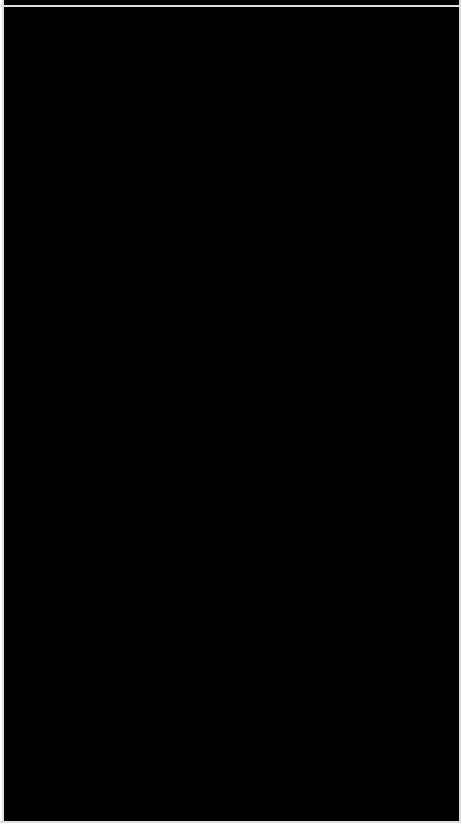
Step_1	Enable the automotive Ethernet link. LocalServer_OSPrompt:~$ **sudo ip link set dev [LINUX_DEVICE] up**	[redacted]

Disabling an a utomotive Ethernet link

Step_1	Disable the automotive Ethernet link. LocalServer_OSPrompt:~$ **sudo ip link set dev [LINUX_DEVICE] down**	[redacted]

Configuring an automotive Ethernet link

NOTE: Command **ip up** must be used for **ethtool** to be operational.

Step_1	Set the speed (in Mbps), the duplex mode, the control mode and the negotiation mode. LocalServer_OSPrompt:~$ **sudo ethtool -s [LINUX_DEVICE] speed [SPEED] duplex [DUPLEX_MODE] master-slave [CONTROL_MODE] autoneg [NEGOTIATION_MODE]** Where: [SPEED] is 100 or 1000 [DUPLEX_MODE] is full (half duplex is not supported) [CONTROL_MODE] is forced-master or forced-slave [NEGOTIATION_MODE] is off (on is not supported) **NOTE:** Because autonegotiation cannot be enabled, the interface needs to cycle to be able to proceed with configurations. This is why the interface needs to be set to down and then to up again (see the example below in the Verifying communication section). There is no automatic calibration when a cable is changed when the system is up. A system cycle is also required in this case.	
Step_2	Verify the current configuration of the automotive Ethernet link. LocalServer_OSPrompt:~$ **sudo ethtool [LINUX_DEVICE]** **NOTE:** The result shown in this example assumes that a slave is connected to enp17s0.	

Verifying communication

To verify communication, messages are sent between the interfaces to confirm they are properly sent and received. Ensure one port is configured as a master and one is configured as a slave. Also ensure they both have the same speed.

Steps	Commands	Linux master shell	Linux slave shell
Step_1	Configure the network of the automotive Ethernet interfaces.		
Step_2	Configure the CONTROL_MODE of each automotive Ethernet interface.		
Step_3	Start a iperf test to confirm connectivity.		
Step_4	Check the SQI _Status. **NOTE:** If the SQI is 0/7, there is a problem. The cable should be checked.		

Monitoring

Monitoring platform components

Table of contents

As a best practice, it is recommended to create a software tool to monitor various platform parameters and execute necessary actions.
As an example, actions could be programmed based on temperature values provided.
Command: **sensors nct7802-***
Results:
...
CPU_Temp: +44.2°C (low = -40.0°C, high = +95.0°C) (crit = +100.0°C)
...

Monitoring sensors

There are several methods to monitor platform sensors, including:

- Using the OS
- Using CBIT/kehm

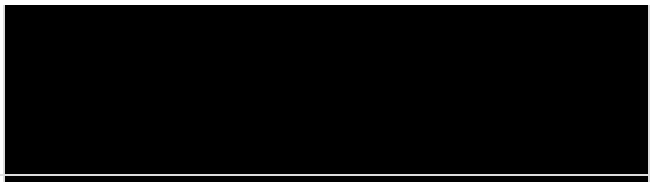
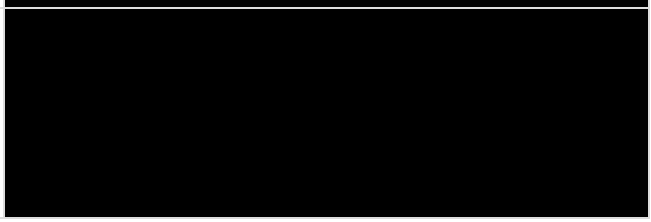
Monitoring sensors using the OS

Refer to Accessing the operating system of a server for access instructions.

Step_1	Access the operating system console.
Step_2	Display all the sensors. LocalServer_OSPrompt:~# **sensors** OR Read sensor data for a specific device, which can be specified as a parameter. LocalServer_OSPrompt:~# **sensors [DEVICE_COMMAND]** **NOTE:** Refer to the table below for examples of commands for certain devices.

Device	Description	Type	Address	Command	Output
ADS7830	8-Channel Sampling A/D Converter	Voltage Current Temperature via thermistors	SMBus 0x48	sensors ads7830-i2c-*-48	[redacted]
			SMBus 0x49	sensors ads7830-i2c-*-49 **NOTE** : "Divide by zero" errors can occur when thermistors are not populated. Sensors NTC1 and NTC2 are internal, and NTC3 is the external sensor on the J2 connector. You can use command: sensors ads7830-i2c-*-49 2> /dev/null to remove error from output.	[redacted]
CoreTemp	Intel Digital Thermal	Temperature	ISA 0	sensors coretemp-*	[redacted]

	Thermal Sensor				[redacted]
lm5056	Power supply sensor	Voltage Current Temperature	SMBus 0x58	sensors lm5056-* NOTE: PSU is in two stages. Vout reported by this command is the mid-stage voltage and is equal to approximately 18 V. The actual PSU Vout can be read from the carrier board (ADS7830) using variable V_12V_S0.	[redacted]
MAX6581	8-Channel Temperature Sensor	Temperature	SMBus 0x4d	sensors max6581-i2c-*-4d	[redacted]
NCT7802	Hardware Monitoring IC	Temperature Voltage	SMBus 0x2c	sensors nct7802-i2c-*-2c	[redacted]
SiC451	DC/DC Converter with PMBus Interface	Voltage Current Power Temperature	SMBus 0x10	sensors sic451-i2c-*-10	[redacted]
			SMBus 0x11	sensors sic451-i2c-*-11	[redacted]
			SMBus 0x12	sensors sic451-i2c-*-12	[redacted]
			SMBus	sensors sic451-i2c-*-13	[redacted]

			0x13		
			SMBus 0x14	sensors sic451-i2c-*-14	

Monitoring platform components using CBIT/kehm

Kehm is a Continuous Built-In Test monitoring framework for computer designers, integrators and end users. It aggregates information from sensors, PBIT and other operating system metrics to provide a comprehensive view of the system health. Every minute, CBIT creates the kehm-RESULT.xml file, which contains the status of sensors and other metrics related to the system. This .xml file can be used to create a full log of system statuses in time that can be exported to different analysis tools.
NOTE: Kehm is provided in evaluation form, which allows monitoring, but no customization. Contact your Kontron sales representative to obtain a full license giving access to all kehm features.

Accessing CBIT/kehm

CBIT/kehm can be accessed using:

- A browser with the IP of eno2 (this functionality is enabled if CBIT/kehm was installed in demo mode during BSP installation)
- The OS CLI

Relevant sections:
Installing the board support package
Discovering platform IP addresses
Accessing the operating system of a server

Monitoring using a browser

Step_1	Open a browser and enter the IP of eno2 to access the CBIT/kehm Web interface.
Step_2	Monitor using the information provided in the Web interface.

Monitoring using the OS CLI

Access the OS CLI. Refer to Accessing the operating system of a server for access instructions.

Step_1	Use the following command to run a single evaluation of the default sensors (requires administrative privileges (sudo)).
	LocalServer_OSPrompt:~#
	/usr/share/kehm/kehm_example.exe -i /usr/share/kehm/S1901-sensors.xml -o kehm-RESULT.xml
		`LocalServer_OSPrompt:~# /usr/share/kehm/kehm_example.exe -i /usr/share/kehm/S1901-sensors.xml -o kehm-RESULT.xml`
		`Using default sensor-status-file: kehm-RESULT.xml`
		`kehm suggested loop time is 10 seconds, sensors evaluation`
		`timeout is set to 5 second(s)`
		`evaluating kehm sensors ... done`
		`/usr/share/kehm/kehm_example.exe: GLOBAL STATUS for S1901 is ALARM`

Step_2	Use the following commands to get details about sensor statuses.
	
	LocalServer_OSPrompt:~#
	/usr/share/kehm/kehm_walktree.exe -f kehm-RESULT.xml
		`/usr/share/kehm/kehm_walktree.exe -f kehm-RESULT.xml`
		`> Top level Computer Health Status using KEHM : ALARM`
		`--> Vital Product Data : WARNING`
		`----> COMexpress Manufacturer : SUCCESS`
		`----> COMexpress BoardName : SUCCESS`
		`----> COMexpress Board Serial Number : SUCCESS`
		`----> COMexpress Hardware Version : SUCCESS`
		`----> COMexpress Manufacturing Date : SUCCESS`
		`----> COMexpress last Repair Date : SUCCESS`
		`----> System Product Name : SUCCESS`
		`...`

Monitoring voltages and temperatures using the AURIX MCU

Relevant section:
AURIX MCU demo code

	Note that AN8 (in the table below) is not available in S1901 platforms.

Access the network switch Web UI. Refer to Accessing the switch network operating system for access instructions.
When an AURIX MCU is installed, it can monitor some of its components, but also some platform components, for example:

- Voltages
- Temperatures

To monitor these elements, clients need to program functionalities in the AURIX MCU. The value read for an AURIX port can be interpreted using its value range listed in the table below.

The table below gives a correspondence between AURIX voltages and platform voltages.

AURIX port	AURIX voltage	Value range	Source
AN0	V_1V8_S0	0 to 5 VDC	Platform
AN1	V_3V3_S0	0 to 5 VDC	Platform
AN2	V_2V8_S0	0 to 5 VDC	Platform
AN3	V_3V3_RT	0 to 5 VDC	Mezzanine
AN4	V_5V0_S5	0 to 10 VDC	Platform
AN5	V_2V5_S0	0 to 5 VDC	Platform
AN6	V_12V0_S5	0 to 15 VDC	Platform
AN7	V_1V0_S0	0 to 5 VDC	Mezzanine
AN8	VBAT	0 to 49.4 VDC	Platform
AN9	V_1V1A_S0	0 to 5 VDC	Mezzanine
AN10	V_VDD (1.25 V)	0 to 5 VDC	Mezzanine
AN11	V_1V2_S0	0 to 5 VDC	Platform
AN12	V_1V1_S0	0 to 5 VDC	Mezzanine
AN13	V_1V0_AQR	0 to 5 VDC	Platform
AN14	V_2V5_FPGA_S0	0 to 5 VDC	Mezzanine
AN15	V_1V8_FT	0 to 5 VDC	Mezzanine
AN16	V_2V0_AQR	0 to 5 VDC	Platform
AN20	12V_S0_IMON	I = V x 4.065 (A)	Platform
AN21	V_12V_S0	0 to 31.1 VDC	Platform
AN22	V_1V5_S0	0 to 5 VDC	Platform
AN36	V_VDD_Current -	AURIX current (mA) (AN44-AN36)/25	Mezzanine
AN44	V_VDD_Current +		

Monitoring PBIT results

PBIT is an acronym for Power on Built-In Test. Its main purpose is to test a computer platform prior to launching the main software to assess its proper behavior and health status.
The results will reflect system status when the last boot occurred.
To use the PBIT tool, it must be enabled in the EFI Shell. Refer to Configuring PBIT for instructions.
NOTE: The tests enabled within the PBIT tool are the ones that are the most time-efficient to run. Other tests can be enabled using the kdiag command (run "kdiag help" for details).

Refer to Accessing the operating system of a server for access instructions.

Step_1	Access the operating system console.	
Step_2	Use the following command to view the PBIT results. LocalServer_OSPrompt:~# **kdiag stat**	LocalServer_OSPrompt:~# kdiag stat Display status Status of PBITs configured to run from command line : PASSED : mem_data (fast,simple) PASSED : mem_addr (fast,simple) PASSED : mem_pattern1 (fast,simple) PASSED : mem_pattern2 (fast,simple) PASSED : mem_pattern3 (fast,simple) PASSED : mem_pattern4 (fast,simple) PASSED : cpu_dmi (fast,simple) PASSED : tpm (fast,simple) PASSED : com2 (fast,simple) PASSED : rtc (fast,simple) PASSED : cpld (fast,simple) PASSED : smbus (fast,simple) PASSED : hwmon (fast,simple) PASSED : jida_eeprom (fast,simple) PASSED : vpd (fast,simple) PASSED : gbe0_loop (slow,simple) PASSED : 10gbe0_loop (slow,simple) PASSED : sata0_ctrl (fast,simple) PASSED : xhci_ctrl (fast,simple) PASSED : system (fast,simple) RUN : 20 PASSED : 20 FAILED : 0 NOT_RUN: 0

Maintenance

System event log

Operating system system event log

The operating system of the platform includes a system event log (SEL).
To access it, refer to the OS manufacturer's documentation.

Typical commands in Linux

Getting the boot message log	dmesg
Getting logs from applications	cat /var/log/syslog

Network switch system event log

Access the network switch Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu of the switch NOS Web UI, select **Monitor** , then **System** , and then **Log** .	

Component replacement

Refer to Components installation and assembly for component replacement procedures.

Backup and restore

Table of contents

On an S1901 platform, the switch NOS configuration can be backed up and restored.

Switch NOS configuration

This section describes how to backup and restore the switch NOS configuration.
NOTE: To restore the factory default configuration, refer to Factory default .

Backing up the switch NOS configuration

This operation can be achieved:

- Using SCP
- Using the switch NOS Web UI

Backing up the switch NOS configuration using SCP

Prerequisites

1	A server configured for the desired protocol is available and accessible from the switch NOS.

	The URL following the server IP address is a path relative to the user home folder provided ("~/"). To specify an absolute path, use a double slash after the IP address (e.g. scp://<SERVER_USERNAME>:<SERVER_PASSWORD>@<SERVER_IP>//<path/to/configfile>).

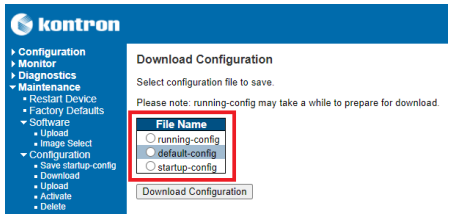
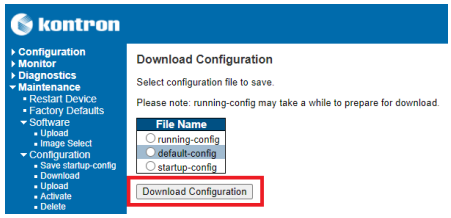
Procedure

Refer to Accessing the switch network operating system for access instructions.

Step_1	Access the switch network operating system using SSH or a serial connection.	
Step_2	Copy the desired configuration to the remote server. • **running-config** : Configuration currently active (may differ from startup-config if changes were made since the last boot, but not saved). • **startup-config** : Saved configuration applied at switch boot. • **default-config** : Configuration applied when the default configuration is reloaded. LocalSwitchNOS_OSPrompt:~# **copy <running-config|startup-config> scp://<SERVER_USERNAME>:<SERVER_PASSWORD>@<SERVER_IP>/<FILE_PATH> save-host-key**	

Backing up the switch NOS configuration using the switch NOS Web UI

Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu of the switch NOS Web UI, select **Maintenance** , then **Configuration** , and then **Download** . Choose the configuration to back up: • **running-config** : Configuration currently active (may differ from startup-config if changes were made since the last boot, but not saved). • **default-config** : Configuration applied when the default configuration is reloaded. • **startup-config** : Saved configuration applied at switch boot.	
Step_2	Click **Download Configuration** , then select where to save the configuration file.	

Restoring the switch NOS configuration

This operation can be achieved:

- Using SCP
- Using the switch NOS Web UI

Relevant section:

Network switch configuration load error messages (to troubleshoot error messages associated with a restore procedure)

If error messages are generated when restoring the switch NOS configuration or upgrading its firmware, refer to the Troubleshooting section.

Restoring the switch NOS configuration using SCP

Prerequisites

1	A server configured for the desired protocol is available and accessible from the switch NOS.
2	If restoring a configuration, the corresponding configuration file is present on the server.

The URL following the server IP address is a path relative to the user home folder provided ("~/"). To specify an absolute path, use a double slash after the IP address (e.g. scp://<SERVER_USERNAME>:<SERVER_PASSWORD>@<SERVER_IP>//<path/to/configfile>).

Procedure

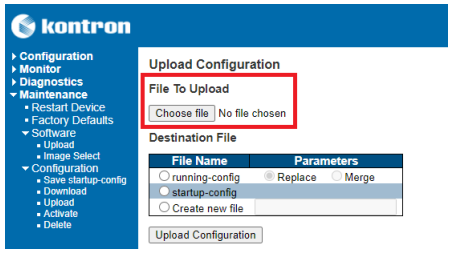
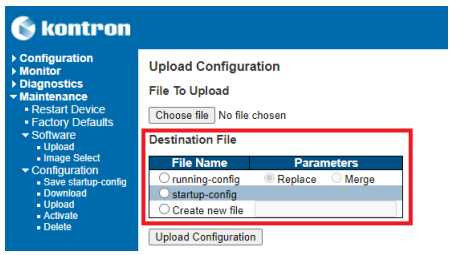
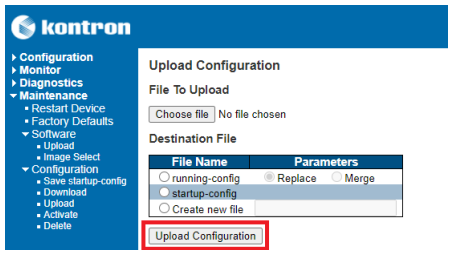
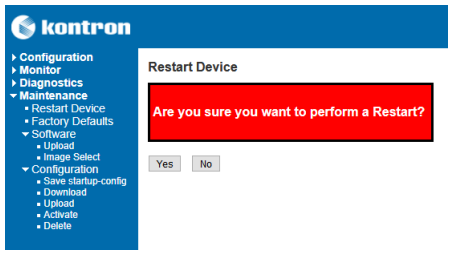
Refer to Accessing the switch network operating system for access instructions.

Step_1	Access the switch network operating system using SSH or a serial connection.	
Step_2	Copy the configuration file from the remote server specifying the destination file, which can be one of the following: • **running-config** : Configuration currently active (volatile until saved as startup-config). • **startup-config** : Saved configuration applied at switch boot. LocalSwitchNOS_OSPrompt:~# **copy scp://<SERVER_USERNAME>:<SERVER_PASSWORD>@ <SERVER_IP>/<FILE_PATH> <running-config|startup-config> save-host-key**	[redacted]
Step_3	If the configuration was written to the startup-config, the switch NOS must be rebooted for the changes to take effect. LocalSwitchNOS_OSPrompt:~# **reload cold**	[redacted]

Restoring the switch configuration using the switch NOS Web UI

If the procedure generates error messages, they will not be shown in the switch NOS Web UI. They are only visible from a CLI interface.

Access the switch NOS Web UI. Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu of the switch NOS Web UI, select **Maintenance** , then **Configuration** , and then **Upload** . Click **Choose file** . Then, using the pop-up file browser, select the desired configuration file to restore.	
Step_2	Choose the configuration to restore: • **running-config** : configuration currently active (volatile until saved as the startup-config). This selection allows fully replacing or merging on top of the current running-config. • **startup-config** : saved configuration applied at switch boot. • **Create new file** : creates a new configuration entry that can be subsequently activated using the **Maintenance** → **Configuration** → **Activate** path of the menu. **NOTE:** A default-config cannot be written to, but a previously backed up default-config can be written to as one of these options.	
Step_3	Click **Upload Configuration** .	
Step_4	If the configuration was written to as startup-config, the switch NOS must be rebooted for changes to take effect. This can be achieved by selecting **Maintenance** , then **Restart Device** from the left-side menu. Then, confirm that a restart is to be performed by clicking **Yes** .	

Upgrading

Table of contents

Upgrading the BSP

To upgrade the BSP on Ubuntu 18.08 or Ubuntu 20.04, proceed the same way as a fresh BSP installation as the installation scripts will remove previously installed BSP modules. Refer to Installing the board support package for instructions.

Upgrading the UEFI/BIOS firmware

UEFI/BIOS failsafe update has been implemented on this platform using two SPI flashes:

- The working SPI flash, which is the one being written to during an upgrade
- The alternate (or golden) SPI flash, which contains a working, proven version of the firmware

The failsafe update is designed to safely boot from a secondary boot source until the user has successfully installed a valid UEFI/BIOS.

Prerequisites

1	The **.zip** archive provided by Kontron has been decompressed in a folder on a Linux OS installed on the platform .
2	Secure boot must be disabled.
3	The COMe watchdog must be disabled.

Relevant sections:
Installing the board support package
Accessing the operating system of a server
Configuring UEFI BIOS options (for instructions to disable secure boot and the COMe watchdog)

Procedure

NOTICE	Before upgrading the UEFI/BIOS, save all your work as the utility will reboot the unit to complete the upgrade.

Step_1	Access the operating system and open a command line interface.
Step_2	Access the folder where the **.zip** archive was extracted. LocalServer_OSPrompt:~# **cd [FOLDER_NAME]**
Step_3	E xecute the upgrade script using the extracted S1901Rxxx.bin file. LocalServer_OSPrompt:~# **failsafe-update S1901Rxxx.bin** **NOTE:** It may take a moment for the UEFI/BIOS firmware upgrade to complete.
Step_4	The failsafe-update script will ask: `Would you like to reboot now? [y/n]` Type **y** to reboot now.
Step_5	Following the reboot, the UEFI/BIOS will attempt to boot from the newly programmed bank or fallback to the golden one. To verify the version, use the following command: LocalServer_OSPrompt:~# **dmidecode -s bios-version** If the upgrade was successful, the [FILENAME] version will be displayed. If the upgrade failed, use command **journalctl -u failsafe-update-post-installation -b** to find out why or return to Step_4 to try again.

NOTE:
The failsafe update is designed to safely boot from a secondary boot source until the user has successfully installed a valid UEFI/BIOS.
For more information on using the failsafe-update utility and the expected output, refer to BSP's README.md file.

Upgrading the network switch firmware

The network switch firmware can be upgraded using:

- SCP
- The switch NOS Web UI – This method can only be used if the Web connectivity is highly reliable. If the file transfer stops, simply start again.

Relevant section:
Network switch configuration load error messages (to troubleshoot error messages associated with an upgrade procedure)

	If error messages are generated when restoring the switch NOS configuration or upgrading its firmware, refer to the Troubleshooting section.

Upgrading the switch firmware using SCP

Prerequisites

1	A server configured for the desired protocol is available and accessible from the switch NOS.
2	The . **itb** upgrade file provided by Kontron was downloaded on the server.
3	The NOS configuration has been backed up.

Relevant sections:
Accessing the switch network operating system
Backup and restore (to backup the NOS configuration)
Factory default (to restore the switch configuration to factory defaults)

Procedure

	The URL following the server IP address is a path relative to the user home folder provided ("~/"). To specify an absolute path, use a double slash after the IP address (e.g. scp://[SERVER_USERNAME]:[SERVER_PASSWORD]@[SERVER_IP]//[path/to/filename.itb]).

Step_1	Access the switch NOS using SSH or a serial connection.	
Step_2	Initiate firmware download and upgrade. LocalSwitchNOS_OSPrompt:~# **firmware upgrade scp://[SERVER_USERNAME]:[SERVER_PASSWORD]@[SERVER_IP]/[FILE_PATH] save-host-key**	[redacted]
Step_3	Wait for the switch NOS to reboot after the upgrade completes.	
Step_4	Confirm the upgrade was successful by checking the firmware version. LocalSwitchNOS_OSPrompt:~# **show version** In the results, look for the version in the Primary Image section. In the image, the version is 2.26.016a3532.	[redacted]

Upgrading the switch firmware using the switch NOS Web UI

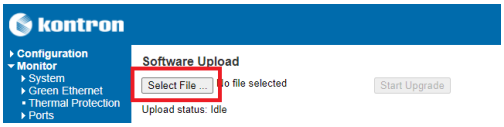
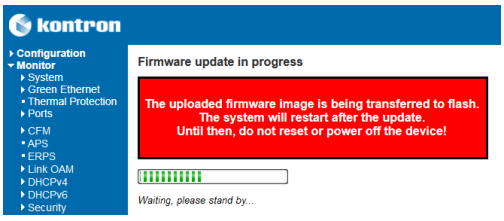
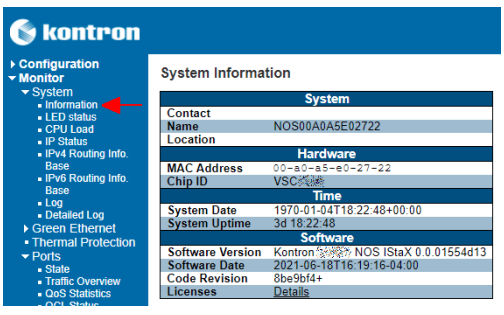
Prerequisites

1	Access to the switch NOS Web UI is required.
2	The . **itb** upgrade file provided by Kontron was downloaded on the remote computer.

Relevant section:
Accessing the switch network operating system

Procedure

	If the procedure generates error messages, they will not be shown in the switch NOS Web UI. They are only visible from a CLI interface.

Step_1	From the left-side menu of the switch NOS Web UI, select **Maintenance** , **Software** and then **Upload** .	
Step_2	Click the **Select File** button and then choose the desired **.itb** file.	
Step_3	After selecting the file for the upgrade, click on **Start Upgrade** .	
Step_4	Wait for the upload and upgrade process to complete.	
Step_5	Once the upgrade is done, from the left-side menu, select **Monitor** , **System** and then **Information** . Confirm that the **Software Version** corresponds to that of the **.itb** file.	

Upgrading the FPGA

The FPGA can be upgraded:

- From the OS CLI

NOTICE	**Once the FGPA transfer begins, it must not be stopped.** Otherwise, the platform will need to be shipped back to Kontron for a factory update. Make sure power or communication is not lost during the upgrade. **We also recommend performing the upgrade via the console on RS-232 #1.** This will ensure that the session is not interrupted should there be a network failure. It is however **not recommended to run the session in the background** , as with background execution, it will not be possible to know when the FPGA upgrade is completed.

Prerequisites

1	The following software tools are installed: • sudo apt install -y i2c-tools • sudo apt install -y python3-pip • python3 -m pip install smbus2 • python3 -m pip install ply
2	The FPGA upgrade file was obtained from Kontron. The file will be in a format similar to: S1901-FPGA_v1.2.0.zip.

Relevant section:

Accessing the operating system of a server using a physical connection

Checking the FPGA version

To perform the upgrade, the FPGA version must be **higher than 1.0.080045EC** .

Step_1	Switch to the root user. LocalServer_OSPrompt:~# **sudo su**	
Step_2	Use the following command lines to determine the FPGA version . LocalServer_OSPrompt:~# **i2cbus="$(i2cdetect -l | grep "SMBus I801" | cut -f 1 | cut -d - -f 2)"** LocalServer_OSPrompt:~# **i2cdump -y $i2cbus 0x55 i | grep 00\: | awk '{printf "%d.%d.%s%s%s%s\n", $17, $16, $15, $14, $13, $12 }'**	

Transferring the upgrade file on the FPGA

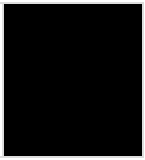
If the FPGA version is appropriate, proceed with the transfer.

Step_1	Decompress the upgrade file. LocalServer_OSPrompt:~# **unzip [FILE_NAME].zip**
Step_2	Access the directory where the file was decompressed. LocalServer_OSPrompt:~# **cd [FILE_NAME] /**
Step_3	Make the script executable. LocalServer_OSPrompt:~# **chmod +x svfplayer**
Step_4	Perform the FPGA upgrade. LocalServer_OSPrompt:~# **i2cbus="$(i2cdetect -l \| grep "SMBus I801" \| cut -f 1 \| cut -d - -f 2)"** LocalServer_OSPrompt:~# **zcat RD10031-fpga-1.2.08002246.svf.gz \| ./svfplayer -d /dev/i2c-$i2cbus -a 0x55 -b 0x80** **NOTE:** Numbers indic ating the transfer progression will be displayed. Once transfer is completed, the CLI will be accessible again.

Performing a power cycle

Step_1	Send a shutdown command. LocalServer_OSPrompt:~# **shutdown now**
Step_2	When the following is displayed: **[OK] Reached target Power-Off.** **[346.449149] reboot: Power down** Remove the power connector from the S1901. Wait 5 seconds and reconnect the power connector to the S1901. This will make the new FPGA version usable.

Confirming proper FPGA upgrade

Step_1	Confirm the new FPGA version. LocalServer_OSPrompt:~# **i2cbus="$(i2cdetect -l | grep "SMBus I801" | cut -f 1 | cut -d - -f 2)"** LocalServer_OSPrompt:~# **sudo i2cdump -y $i2cbus 0x55 i | grep 00\: | awk '{printf "%d.%d.%s%s%s%s\n", $17, $16, $15, $14, $13, $12 }'**	

Upgrading the AURIX MCU

The following procedure uses a Windows virtual machine.

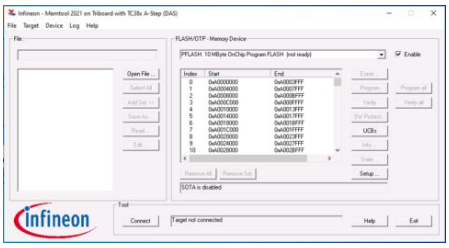
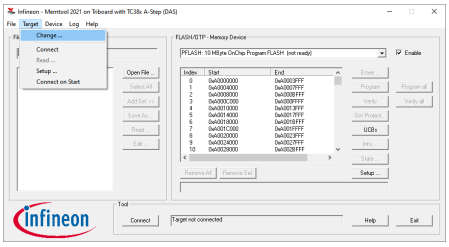
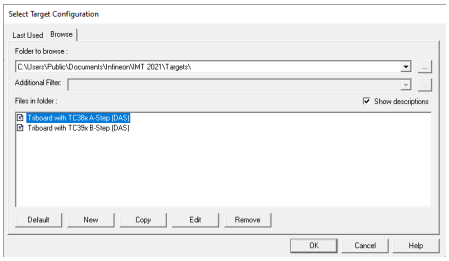
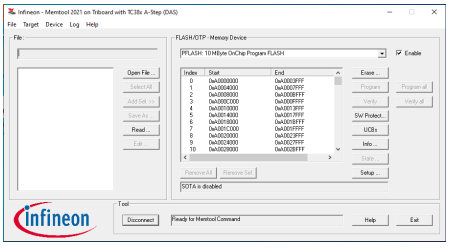
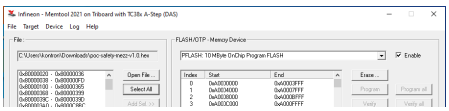
Prerequisites

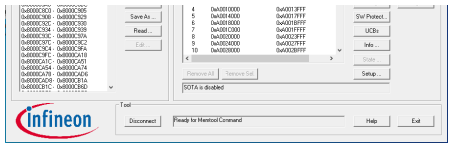
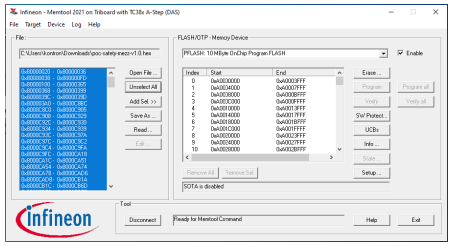
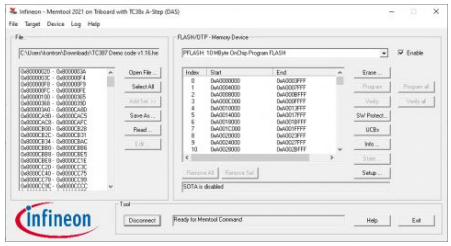
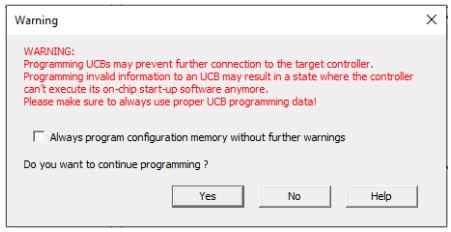
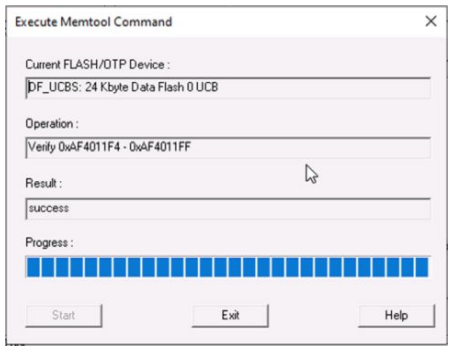
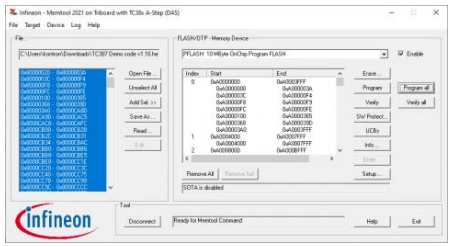
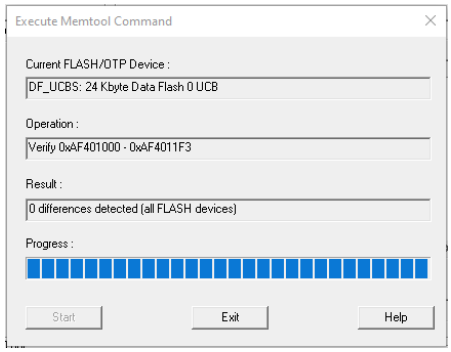
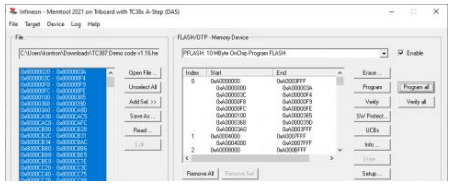
1	The AURIX MCU development environment must be installed.
2	A new demo code version (to upgrade the demo code) or a new custom code version (to upgrade custom code) is available and accessible from MemTool. The code must be compiled.

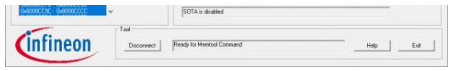
Relevant sections:
Installing the AURIX MCU development environment and demo code
AURIX MCU demo code (for instructions on how to compile the code)

Procedure

Downloading the code compiled to the AURIX MCU

Step_1	From the Linux CLI, run the aurix-pre-prog.sh script. This will disable the watchdog timer of the Multi Voltage Safety Micro Processor Supply (TLF35584) installed on the CAN bus mezzanine with the AURIX safety MCU and perform a reset. LocalServer_OSPrompt:~# **sudo aurix-pre-prog.sh**	
Step_2	From the VM desktop, double click on the **Infineon Memtool** icon.	
Step_3	Click on **Connect** .	
Step_4	Confirm the AURIX target is the right one. From the **Target** tab, click on **Change...**	
Step_5	Select the proper AURIX target and click on **OK** . The possible targets are: • TC387: Select Triboard with TC38x A-Step (DAS) • TC397: Select Triboard with TC39x B-Step (DAS)	
Step_6	Click on **Connect** and confirm that **Ready for MemTool Command** is displayed in the **Tool** status text box.	
Step_7	Click on **Open File** . Select the appropriate .hex file and click on **Open** .	

		
Step_8	Click on **Select All** .	
Step_9	Click on **Add Sel >>** and click on **Program all** .	
Step_10	Click on **Yes** .	
Step_11	Once **success** is displayed in the **Result** text box, click on **Exit** .	
Step_12	Click on **Verify all** .	
Step_13	Message **0 differences detected (all FLASH devices)** should be displayed in the **Result** text box. Once it is, click on **Exit** .	
Step_14	Click on **Exit** .	

		
Step_15	From the Linux CLI, run the aurix-post-prog.sh script. This will re-enable the watchdog timer of the Multi Voltage Safety Micro Processor Supply (TLF35584) installed on the CAN bus mezzanine with the AURIX safety MCU and perform a reset. LocalServer_OSPrompt:~# **sudo aurix-post-prog.sh**	

Validating the demo code installation

This step is required only when the demo code is upgraded. For custom code upgrade, clients must create their own validation method.

Step_1	Open the OS CLI and connect via minicom. LocalServer_OSPrompt:~# **resize ; sudo minicom -w -D /dev/ttyS1**	
Step_2	Open the shell to confirm installation. The version number will be displayed. Shell> **status** **NOTE:** Version number should be 1.16.	

Platform cooling and thermal management

Table of contents

Defining the clock speeds of the GPU for maximum ambient temperature

This section explains how to define the maximum clock speeds allowed to avoid GPU throttling.

Prerequisites

1	Three distinct consoles must be open: • One for monitoring GPU temperature and clock frequencies • One for configuring the maximum clock speeds of the GPU • One for performing the GPU load test to see the actual performance
2	The ambient temperature outside the platform must be known. **NOTE:** A thermistor could be attached to the 70-pin connector (pins 17 and 18 of J15). The NTC thermistor should be a 10 kΩ thermistor with a bead of 3976K (TE GA10K3A1IA). If the test is performed on an S1901 platform, the Kontron cable (1068-5062) that provides this thermistor could be used.

Monitoring the GPU

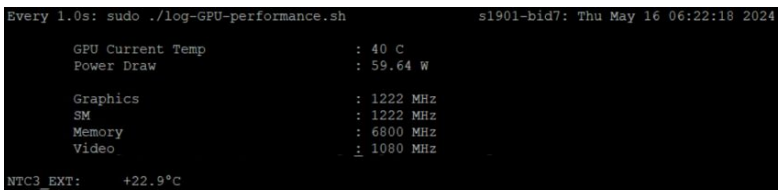
Creating the monitoring script

To monitor the GPU create the script "log-GPU-performance.sh" with the following content.

```
#!/bin/bash
TMP_OUTPUT=$(nvidia-smi -q)

echo "$TMP_OUTPUT" | grep -i gpu\ current\ temp
echo "$TMP_OUTPUT" | grep -i gpu\ power\ reading -A 2 | grep -i power\ draw
echo " "
echo "$TMP_OUTPUT" | grep -i \ \ clocks -A 4 | grep -i graphics
echo "$TMP_OUTPUT" | grep -i \ \ clocks -A 4 | grep -i sm
echo "$TMP_OUTPUT" | grep -i \ \ clocks -A 4 | grep -i memory
echo "$TMP_OUTPUT" | grep -i \ \ clocks -A 4 | grep -i video
echo " "
TMP_OUTPUT=$(sensors 2> \dev\null | grep -i ntc)
echo "$TMP_OUTPUT" | grep -i ntc3
```

Establishing continuous monitoring

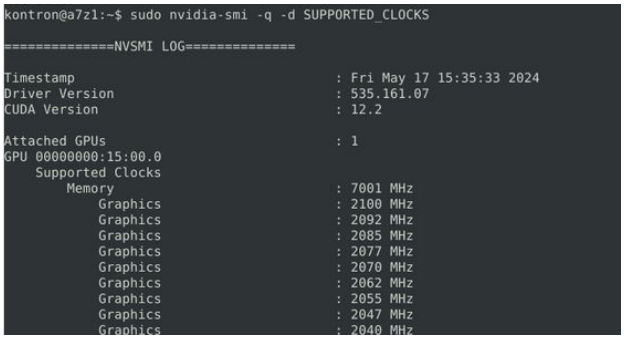
Step_1	Execute the following commands to continuously monitor the GPU temperature, power usage and clock frequencies. LocalServer_OSPrompt:~$ **chmod 766 log-GPU-performance.sh** LocalServer_OSPrompt:~ $ **watch -n 1 "sudo ./log-GPU-performance.sh"**	

Configuring the clock speeds

The GPU has 4 clocks. Two are adjusted automatically and 2 are configurable:

- Memory clock
- GPU clock

Listing supported clock speeds

Step_1	Not all combinations of memory clock and GPU clock speeds are supported. To see the list of supported values use the following command. LocalServer_OSPrompt:~ $ **sudo nvidia-smi -q -d SUPPORTED_CLOCKS** NOTE: The image shows the beginning of the table displayed.	

Setting the maximum GPU clock speed

Step_1	Set the maximum GPU clock speed. In the example to the right, the speed is set to 1.0 GHz. LocalServer_OSPrompt:~ $ **sudo nvidia-smi -lgc [GPU_CLOCK_VALUE]** Where: [GPU_CLOCK_VALUE] is a supported value listed in the table displayed in section Listing supported clocks	

Resetting the GPU clock to its default value

Step_1	Reset the GPU clock to its default value. LocalServer_OSPrompt:~ $ **sudo nvidia-smi -rgc**

Setting the maximum memory clock speed

Step_1	Set the maximum memory clock speed. In the example to the right, the speed is set to 5.5 GHz. LocalServer_OSPrompt:~ $ **sudo nvidia-smi -lmc [MEM_ CLOCK_VALUE]** Where: [MEM_CLOCK_VALUE] is a supported value listed in the table displayed in section Listing supported clocks	

Resetting the memory clock to its default value

Step_1	Reset the memory clock to its default value. LocalServer_OSPrompt:~ $ **sudo nvidia-smi -rmc**

Performing the GPU load test

One of the consoles open can be used to execute a load test on the GPU to see the actual performance achieved with the memory and GPU clock speeds set. Kontron recommends using program "GPU burn" from https://github.com/wilicc/gpu-burn .

Step_1	Run a load test using GPU burn. In the example to the right, measured performance is 15.9 Tflop/s. LocalServer_OSPrompt:~ **/gpu-burn$ ./gpu_burn -tc 99999** NOTE: A value of 99999 seconds is used to make the test last a very long time.	

Troubleshooting

Collecting diagnostics

Table of contents

Viewing the system information

Refer to Accessing the operating system of a server for access instructions.

Step_1	Collect the system information from within the operating system. LocalServer_OSPrompt:~# **sudo dmidecode -t 1** Information provided includes: • Product Name • Version • Serial Number • UUID

Collecting logs and hardware information

When the support team is contacted, the following data is required to make the proper board health diagnostics:

- Operating system event log and system information
- Network switch information
- CAN bus mezzanine with the AURIX safety MCU information
- Hardware and cabling configuration

Other information is required before contacting Kontron:

- Serial number and part number of the platform (located on the label)
- Part number of the Kontron cables used with the platform

Collecting all this data beforehand can accelerate the process. A command is provided at the end of the section to compress all the files created.

Setting up the environment to collect the logs

Refer to Accessing the operating system of a server for access instructions to the operating system.

Step_1	Set up the environment to create the logs. LocalServer_OSPrompt:~# **mkdir log** LocalServer_OSPrompt:~# **cd log** LocalServer_OSPrompt:~# **sudo su** LocalServer_OSPrompt:~# **apt install -y gpsd-clients ethtool nvme-cli**

Collecting operating system event log and system information

The operating system error messages and logs can be used to make health diagnostics. Refer to the operating system's documentation for instructions.

Step_1	Create the log files for the operating system. LocalServer_OSPrompt:~# **i2cbus="$(i2cdetect -l | grep "SMBus I801" | cut -f 1 | cut -d - -f 2)"** LocalServer_OSPrompt:~# **i2cdump -y $i2cbus 0x55 i | grep 00\: | awk '{printf "%d.%d.%s%s%s%s\n", $17, $16, $15, $14, $13, $12 }' >fpga.txt** LocalServer_OSPrompt:~# **lspci >lspci.txt** LocalServer_OSPrompt:~# **lspci -xxxx -vvv >lspci-all.txt** LocalServer_OSPrompt:~# **lspci -vvv >lspci-verbose.txt** LocalServer_OSPrompt:~# **lspci -tv >lspci-tree.txt** LocalServer_OSPrompt:~# **dmesg >dmesg.txt** LocalServer_OSPrompt:~# **ll /var/log/ | grep -i dmesg >list-of-dmesg.txt** LocalServer_OSPrompt:~# **dmidecode > dmidecode.txt** LocalServer_OSPrompt:~# **uname -a >uname.txt** LocalServer_OSPrompt:~# **lsb_release -a > lsb_release.txt** LocalServer_OSPrompt:~# **lshw -short > lshw-short.txt** LocalServer_OSPrompt:~# **lshw -numeric > lshw-numeric.txt** LocalServer_OSPrompt:~# **lscpu > lscpu.txt** LocalServer_OSPrompt:~# **cat /var/log/syslog > syslog.txt** LocalServer_OSPrompt:~# **cat /var/log/syslog.1 >> syslog.txt** LocalServer_OSPrompt:~# **zcat /var/log/syslog*gz >> syslog.txt** LocalServer_OSPrompt:~# **sensors >sensors.txt** LocalServer_OSPrompt:~# **ip a > ip_a.txt** LocalServer_OSPrompt:~# **ip route > ip_route.txt** LocalServer_OSPrompt:~# **ethtool -i eno1 > eno1-driver.txt** LocalServer_OSPrompt:~# **ethtool -i eno2 > eno2-driver.txt** LocalServer_OSPrompt:~# **ethtool -d eno1 > eno1-register.txt** LocalServer_OSPrompt:~# **ethtool -d eno2 > eno2-register.txt** LocalServer_OSPrompt:~# **nvidia-smi -q > nvidia-smi.txt** LocalServer_OSPrompt:~# **sudo kdiag stat > kdiag-stat.txt** LocalServer_OSPrompt:~# **lspcan -a > lspcan.txt** LocalServer_OSPrompt:~# **ls /dev > devices.txt** LocalServer_OSPrompt:~# **sudo nvme id-ctrl /dev/nvme0 > nvme.txt** LocalServer_OSPrompt:~# **sudo nvme get-feature --feature-id=0x10 --human-readable /dev/nvme0 >>nvme.txt** LocalServer_OSPrompt:~# **ubxtool -f /dev/ttyACM0 -p MON-VER > ubxtool.txt** LocalServer_OSPrompt:~# **cat /sys/class/pcan/pcanpcifd*/adapter_version >peak-can.txt**

Collecting network switch information

In step 1, variable **[xx]** is 6 (ttyS6). Refer to Linux devices for more information.

Step_1	Access the network switch and open a log file. LocalServer_OSPrompt:~# **sudo minicom -D /dev/ttyS[xx] -C NOS-LOGS.txt**
Step_2	Generate the log files for the network switch. **NOTE:** You must press **g** after each command to display all lines. LocalSwitchNOS_OSPrompt:~# **show running-config** LocalSwitchNOS_OSPrompt:~# **show logging** LocalSwitchNOS_OSPrompt:~# **show version** LocalSwitchNOS_OSPrompt:~# **show running-config** LocalSwitchNOS_OSPrompt:~# **show ip interface** LocalSwitchNOS_OSPrompt:~# **show interface * statistics**
Step_3	Exit minicom by pressing **Ctrl-A** and then **x** . This will save the log file.

Collecting information on the CAN bus mezzanine with the AURIX safety MCU

In step 1, variable **[xx]** is 6 (ttyS6). Refer to Linux devices for more information.

Step_1	Access the AURIX MCU switch and open a log file. LocalServer_OSPrompt:~# **sudo minicom -D /dev/ttyS[xx] -C MEZZANINE-LOGS.txt**
Step_2	Generate the log files for the AURIX MCU. Shell> **status** **NOTE:** The **status** command is included in the demo code provided for the AURIX MCU.
Step_3	Exit minicom by pressing **Ctrl-A** and then **x** . This will save the log file.

Compressing all the text files generated

Step_1	Compress all the log files generated to create a single diagnostic package. LocalServer_OSPrompt:~# **cd ..** LocalServer_OSPrompt:~# **zip -u logs-`date --rfc-3339 date`.zip log/*.txt**

Collecting hardware and cabling configuration information

Hardware and cabling configuration information might be required to make the proper board health diagnostics. The following list contains examples of information that could help the Kontron support team.

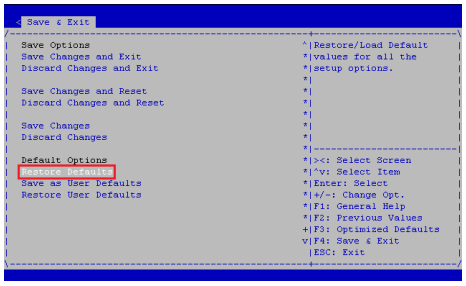
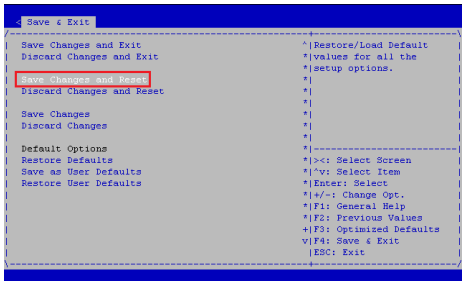
- A list of devices connected to the platform
- A list of cables plugged into the platform
- A list of interfaces used (serial, network, CAN bus, etc.)

Factory default

Table of contents

Restoring default UEFI/BIOS settings

Refer to Accessing the UEFI BIOS for access instructions.

Step_1	From the UEFI/BIOS setup menu, navigate to the **Save & Exit** menu and select **Restore Defaults** .	
Step_2	Select **Save Changes and Reset** .	
Step_3	Wait for the system to reset. The UEFI/ BIOS settings should have been reset to default values.	

Restoring default switch NOS settings

Use caution when restoring default settings. Your access to system components could be interrupted because of changes to various elements, including:

- NOS access via network IP addresses
- NOS user configuration
- Other system components, due to switch forwarding configurations (e.g., VLAN)

Refer to Description of system access methods to select an appropriate path to access the platform components.
It is also **recommended to back up the startup configuration before restoring the default settings** . The backed up file could serve as a reference for future configuration.

Changes to the switch NOS configuration are not persistent after rebooting the switch NOS.
To preserve configurations, the current configuration needs to be saved to startup-config.
From the switch NOS Web UI:

- Select **Maintenance** , **Configuration** and then **Save startup-config** . Click on **Save Configuration** to confirm the change.

From the switch NOS CLI:

- LocalSwitchNOS_OSPrompt:~(config-if)# **end**
- LocalSwitchNOS_OSPrompt:~# **copy running-config startup-config**

Relevant sections:
Description of system access methods
Backup and restore

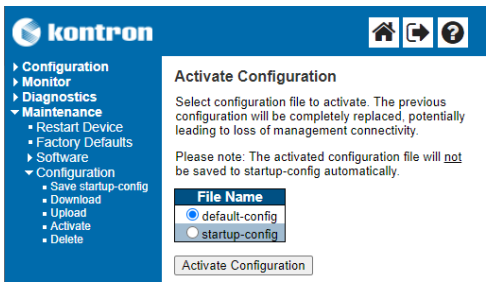
Restoring default switch NOS settings using the CLI

Refer to Accessing the switch network operating system for access instructions.

Step_1	Restore the default configuration. LocalSwitchNOS_OSPrompt:~# **reload defaults**	`# reload defaults` `% Reloading defaults. Please stand by.`
Step_2	(Optional) To make the change persistent, save running-config to startup-config.	

Restoring default switch NOS settings using the Web UI

Refer to Accessing the switch network operating system for access instructions.

Step_1	From the left-side menu, select **Maintenance, Configuration** and then **Activate** .	
Step_2	Click on the **default-config** radio button.	
Step_3	Press on the **Activate Configuration** button to confirm.	
Step_4	(Optional) To make the change persistent, save running-config to startup-config.	

Network switch configuration load error messages

This section describes how to proceed if error messages are generated when:

- The NOS firmware is upgraded. In rare instances, configuration commands may change format in a new firmware version and therefore need correcting.
- The NOS configuration is restored or uploaded using configuration commands that have changed format or was modified remotely with errors.

NOTE: Configuration load errors may only be printed on the serial console interface of the switch NOS.

Relevant section:
Backup and restore (to have a reference of the startup configuration)

Access the switch NOS CLI. Refer to Accessing the switch network operating system for access instructions.

Step_1	Back up the startup configuration to have a reference.	
Step_2	Restore factory default values. LocalSwitchNOS_OSPrompt:~# **reload defaults**	[redacted]
Step_3	Using the reference startup configuration, manually enter the configuration items that differ from the original configuration, and correct errors if needed.	
Step_4	Make the change persistent by saving the running-config to startup-config. LocalSwitchNOS_OSPrompt:~# **copy running-config startup-config**	[redacted]
Step_5	Reboot the NOS to make sure the configuration was applied correctly. LocalSwitchNOS_OSPrompt:~# **reload cold**	

Minicom problems

Table of contents

Line feed problem

When the console is not 80x24, there could be a line feed problem with minicom.
The solution is as follows:

Step_1	Exit minicom by pressing **Ctrl-A** and then **x** .
Step_2	Determine the size of the console screen. LocalServer_OSPrompt:~# **resize**
Step_3	Launch minicom again with the wrap parameter. This will take into account the resize command of the previous step. LocalServer_OSPrompt:~# **sudo minicom -D /dev/ttyS6**

Color problem

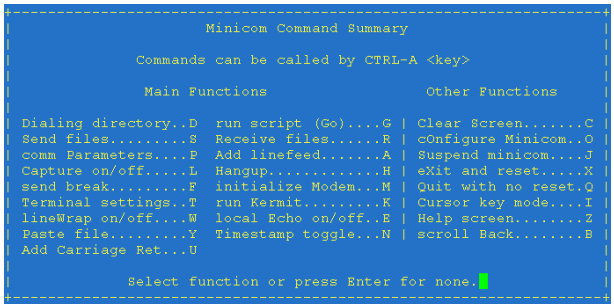
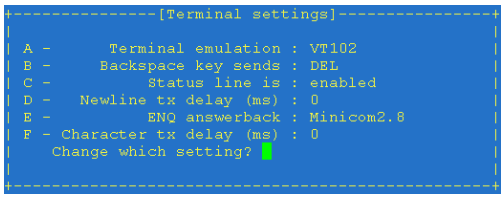
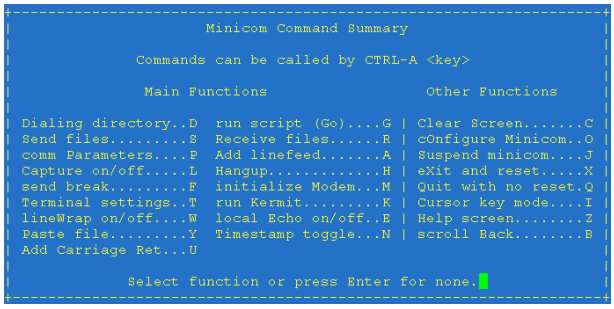
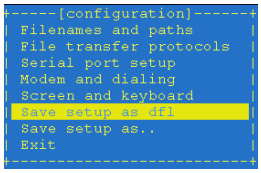
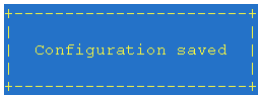
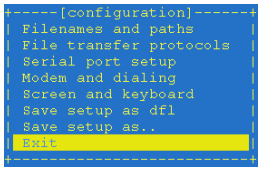
In the UEFI/BIOS, lines could sometimes not be displayed properly because of color contrasts.
The solution is as follows:

Step_1	Exit minicom by pressing **Ctrl-A** and then **x** .
Step_2	Launch minicom again with the color parameter. This will ensure proper color contrast. LocalServer_OSPrompt:~# **sudo minicom -c on -D /dev/ttyS6**

Backspace characters

By default, the backspace key parameter of some terminal emulators (e.g. minicom) is set to BS (ASCII 0x08 or CTRL-H). For communication to work properly, the backspace key parameter of the terminal emulator used must be set to DEL (ASCII 0x0F or CTRL-?).

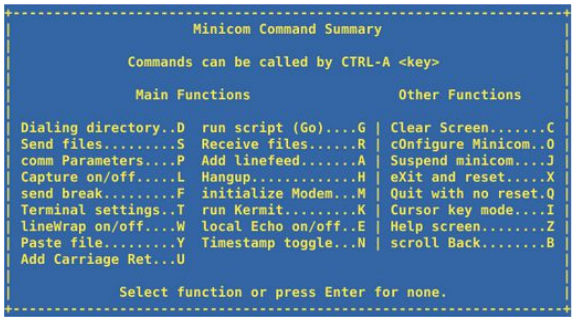
To configure the backspace key parameter, use the following procedure:

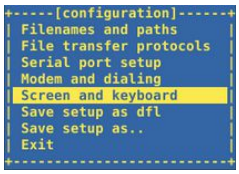
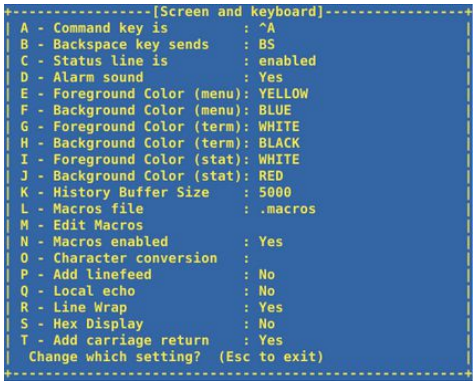
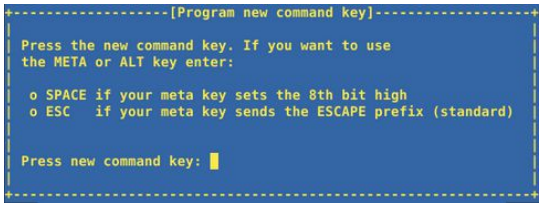
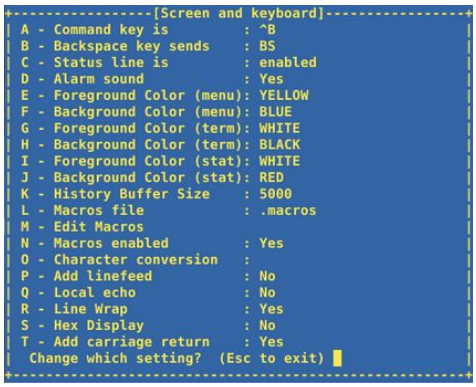
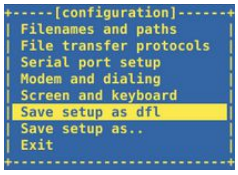
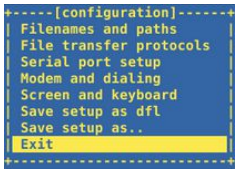
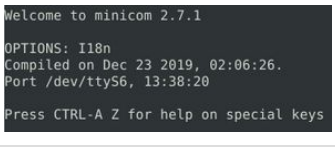
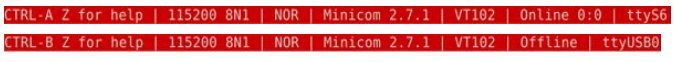
Step_1	Open minicom and edit the settings by typing **Ctrl-A** and then pressing on **Z** .	
Step_2	Edit **Terminal settings** by pressing on **T** .	
Step_3	Press on **B** to set the **Backspace key sends** parameter to **DEL** .	
Step_4	Press on **ESC** to go back to the terminal emulator.	
Step_5	To permanently save the configuration, edit the settings by typing **Ctrl-A** and then pressing on **Z** .	
Step_6	**Configure Minicom** by pressing on **O** .	
Step_7	Select **Save setup as dfl** and press **Enter** .	
Step_8	A message will be displayed indicating the configuration was saved.	
Step_9	Select **Exit** to go back to the terminal emulator.	

Minicom in minicom

If you connect to the platform with minicom, but also need to use minicom to communicate with the platform serial port (NOS or AURIX MCU), you will need to change the command key, so each instance of minicom has its own.

To configure the command key to access the menu use the following procedure.

Step_1	Open minicom and edit the settings by typing **Ctrl-A** and then pressing on **Z** .	
Step_2	Access the minicom configuration menu by pressing on **O** .	

Step_3	Select **Screen and keyboard** and press **En ter** .	
Step_4	Press on **A** to set parameter **Command key is** .	
Step_5	Type **Ctrl-B** as the new command key.	
Step_6	Press on **ESC** to go back to the previous menu.	
Step_7	Select **Save setup as dfl** and press **Enter** .	
Step_8	A message will be displayed indicating the configuration was saved.	
Step_9	Select **Exit** to go back to the terminal emulator.	
Step_10	If you type **Ctrl-B** , the minicom menu bar will be displayed at the bottom of the screen.	
Step_11	To confirm that a new minicom in minicom instance will work, use minicom to connect to the NOS. LocalServer_OSPrompt:~# **sudo minicom -c on -D /dev/[NOS LINUX DEVICE]**	
Step_12	If you type **Ctrl-A** , the menu for the NOS will be displayed (ttyS6). If you type **Ctrl-B** , the menu for the platform will be displayed (ttyUSB0).	

Support information

Kontron's technical support team can be reached through the following means:

- By phone: 1-888-835-6676
- By email: support-na@kontron.com
- Via the website: www.kontron.com

Application notes

Generating custom secure boot keys

Relevant section:
Provisioning custom secure boot keys

To provision custom secure boot keys, keys may have to be generated. This article provides an example using Ubuntu.

Prerequisites

1	Packages efitools and sbsigntools must be available.
2	Tool rodsbooks must be available.

Procedure

Step_1	Run the following commands on the system you need to generate keys for. apt install -y efitools sbsigntool mkdir make_keys cd make_keys wget https://www.rodsbooks.com/efi-bootloaders/mkkeys.sh chmod +x mkkeys.sh ./mkkeys.sh
Step_2	The commands will generate a lot of files. You need the *.cer file to use in the provisioning procedure.

Provisioning custom secure boot keys

Table of contents

Introduction

This article describes how to provision a custom set of Secure Variables used as part of the Secure Boot feature.
Secure Boot is a UEFI-defined feature used to authenticate a UEFI executable, such as an OS loader, using digital signing mechanisms based on the Public Key Infrastructure process, reducing the risks of pre-boot malware attacks. The feature uses a database of authorized signatures to confirm the UEFI executable integrity prior to execution.
Boards will typically have a pre-loaded set of Platform Key (PK), Key Exchange Keys (KEK), authorized signature database (db) and blacklisted / revoked signature database (dbx) as defined by the OEM, as well as some industry-standard certificates issued by Microsoft that allow booting Windows or well-known Linux distributions such as Ubuntu. It may be desirable for an end customer to update these keys with their own set for security reasons.
This document assumes the reader has some knowledge about the Secure Boot process, and that the required set of keys and certificates has been properly generated. The following link provides guidelines on creating and managing such keys and certificates:
https://docs.microsoft.com/en-us/windows-hardware/manufacture/desktop/windows-secure-boot-key-creation-and-management-guidance

Updating secure boot keys from the UEFI setup utility

Prerequisites

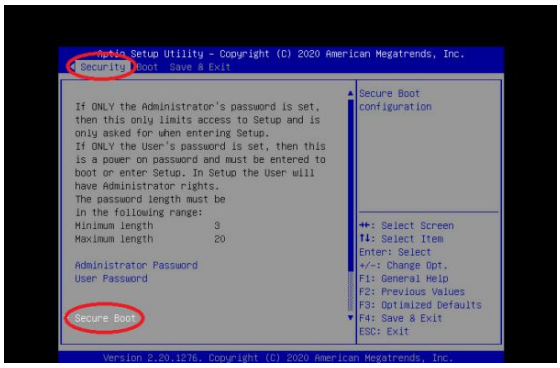
1	A set of Secure Boot keys has been created (PK, KEK and db).
2	Public Key certificates that are to be provisioned are in DER format.
3	Public Key certificates are present on a FAT-partitioned USB drive, which is connected to the board. If Virtual Media redirection is available, it is also possible to use a corresponding ISO image instead.

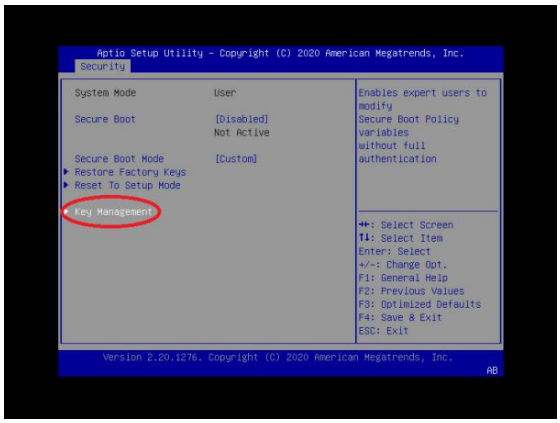
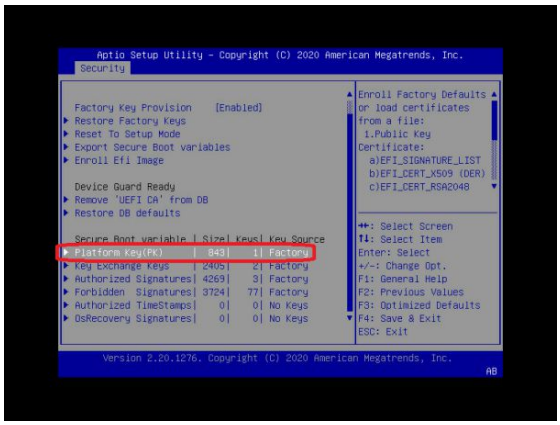
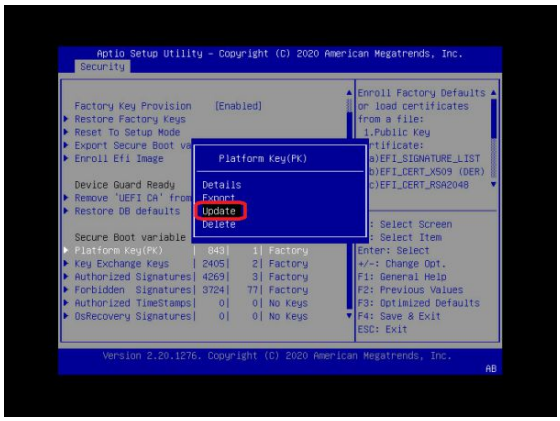
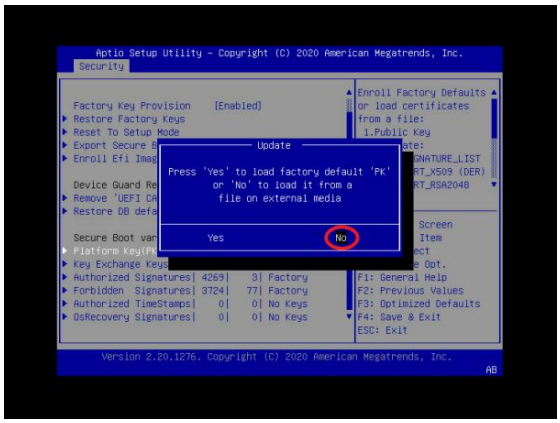
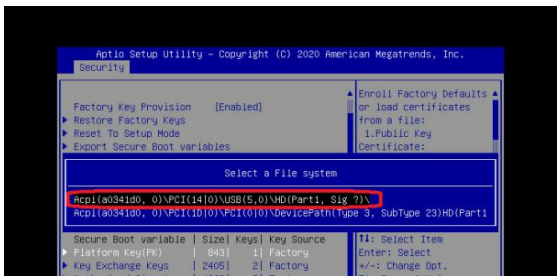
Relevant section:
Generating custom secure boot keys

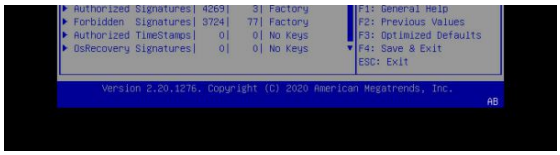
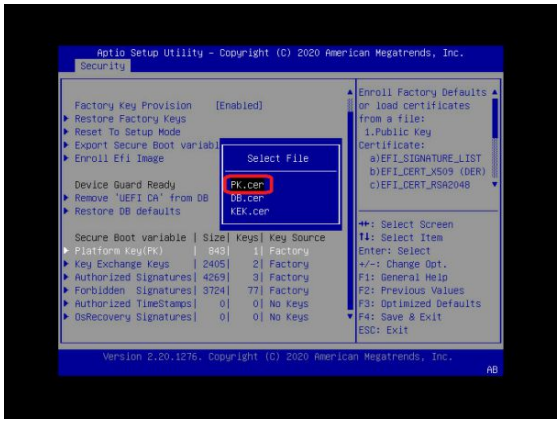
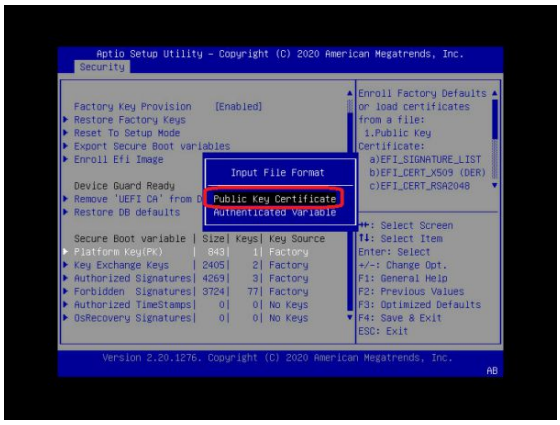
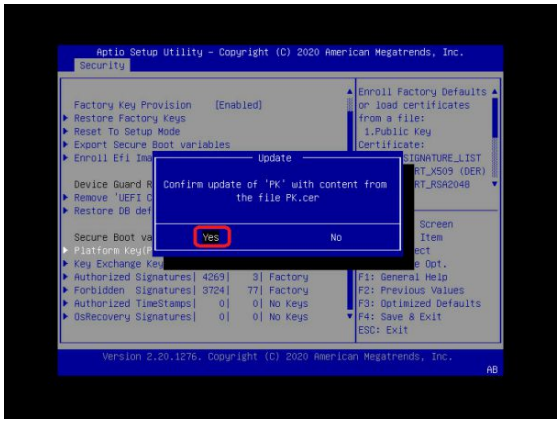
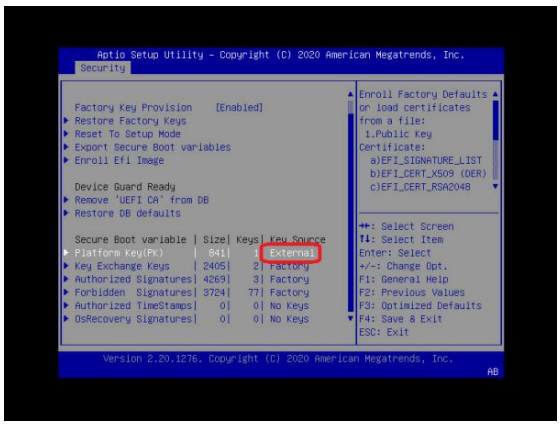
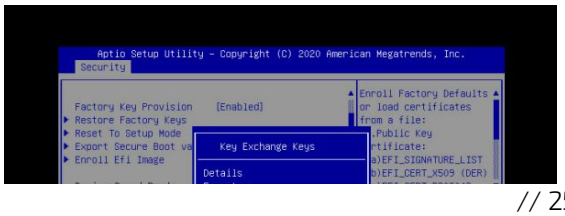
i	As the current time is verified against certificate timestamps as a security measure, make sure the system time is valid prior to manipulating Secure Boot variables. Otherwise, a Security Violation error will be obtained and no change will be possible.

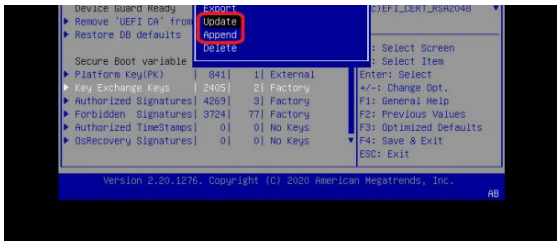
Procedure

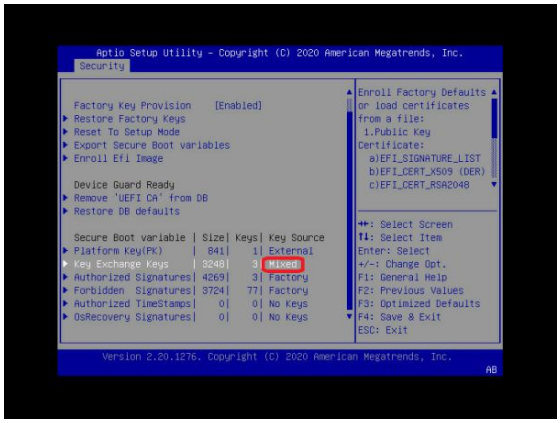
Refer to Accessing the UEFI BIOS for access instructions.

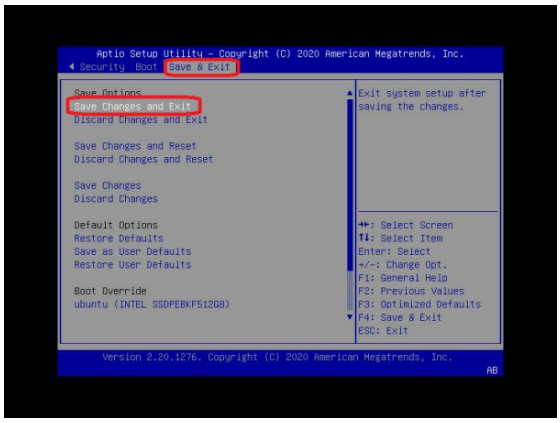
Step_1	Access the UEFI Setup Utility by pressing F2 or DEL when the sign-on screen is displayed during boot.	
Step_2	Access the **Secure Boot** submenu from the **Security** tab.	

		
Step_3	Access the Key Management page by selecting the **Key Management** menu item.	
Step_4	Default Factory Keys should already be provisioned, as identified by the "Factory" attribute in the Key Source column in the Secure Boot variable table. To replace the default Platform Key with your own, select **Platform Key(PK)** .	
Step_5	Select **Update** from the pop-up window.	
Step_6	Select **No** to load a key from an external media.	
Step_7	A list of available file systems will be displayed, using their corresponding UEFI device path. Select the USB device where the Public Key certificates are located. Note that if Virtual Media redirection is used, the device will be identified as a CDROM.	

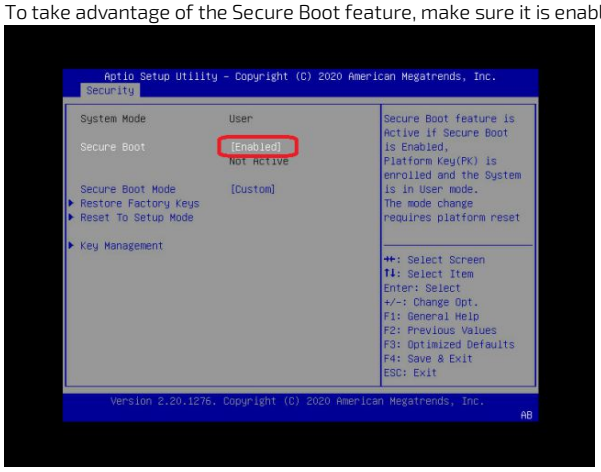
		
Step_8	From the list of files, select the Public Certificate file for the Platform Key (PK.cer in this example).	
Step_9	Specify that the file format is **Public Key Certificate** .	
Step_10	Select **Yes** to confirm Platform Key update.	
Step_11	Confirm that the update completed successfully. The table should now show that a key was added from an "External" Key Source.	
Step_12	Select **Key Exchange Keys** to update or append the KEK database with your own. In this case: • Selecting **Update** from the pop-up window will erase the pre-provisioned KEK entries and add a new KEK as a single entry; • Selecting **Append** will add the new KEK to the database.	



Step_13	Follow steps 4 to 11 to add a new KEK entry. If the KEK was appended to the database, the Key Source will be "Mixed".	

Step_14	Select **Authorized Signatures** to add an authorized Public Key certificate to the db. As for KEK: • Selecting **Update** from the pop-up window will erase the pre-provisioned db entries and add a new certificate as a single entry; • Selecting **Append** will add the new certificate to the database. Follow steps 4 to 11 to add a new db entry. If the certificate was appended to the database, the Key Source will be "Mixed".
Step_15	Select **Save Changes and Exit** from the Setup Utility. 

To take advantage of the Secure Boot feature, make sure it is enabled in the Security → Secure Boot submenu.



User guide for PBIT

Overview

This document describes the Power-On Built-In Tests (PBIT) for the Kontron **COMe-bBD7** board installed in an S1901 product.
PBIT is available under the UEFI Shell environment, embedded in the BIOS stored in the SPI Boot Flash of the board.
It is implemented as the UEFI command **kdiag** as an extension of the standard UEFI Shell commands.
PBIT includes the following services:

- A list of tests that can be added or removed from a run list by a command according to the desired trade-off between time to boot, coverage rate and system configuration.
- An automatic mode when booting firmware or an interactive mode at UEFI Shell prompt.
- Storage of test configurations and results in a non-volatile storage EEPROM device accessible, for example, with a software API to be easily reconfigured by an OS like Linux.

Configuration

Interactive Mode

The following section describes how to run PBIT in interactive mode.
The **UEFI Built-in EFI Shell** must be booted to invoke PBIT **kdiag** command.
You can select the Shell when invoking the Boot Manager or by entering into the Setup in Boot menu:

- To execute the Shell automatically at each startup, set **UEFI: Built-in EFI Shell** as Boot Option #1:
- Then, move to "Save & Exit" menu, select **Save Changes and Reset** and select "Yes".

After reseting, the EFI Shell prompt is displayed, allowing to enter PBIT " **kdiag** " command.
Verify PBIT version:

```
Shell> kdiag version
PBIT VERSION 1.2 ID17069
```

Configure the system:

```
Shell> kdiag learn system
   Seg  Bus  Dev  Func
   ---  ---  ---  ----
    00   00   00    00 ==> Bridge Device - Host/PCI bridge
              Vendor 8086 Device 6F00 Prog Interface 0
    00   00   01    00 ==> Bridge Device - PCI/PCI bridge
              Vendor 8086 Device 6F02 Prog Interface 0
    00   00   01    01 ==> Bridge Device - PCI/PCI bridge
              Vendor 8086 Device 6F03 Prog Interface 0
    00   00   02    00 ==> Bridge Device - PCI/PCI bridge
              Vendor 8086 Device 6F04 Prog Interface 0
    00   00   02    02 ==> Bridge Device - PCI/PCI bridge
              Vendor 8086 Device 6F06 Prog Interface 0
    00   00   03    00 ==> Bridge Device - PCI/PCI bridge
              Vendor 8086 Device 6F08 Prog Interface 0
    00   00   05    00 ==> Base System Peripherals - Other system peripheral
              Vendor 8086 Device 6F28 Prog Interface 0
    00   00   05    01 ==> Base System Peripherals - Other system peripheral
              Vendor 8086 Device 6F29 Prog Interface 0
    00   00   05    02 ==> Base System Peripherals - Other system peripheral
              Vendor 8086 Device 6F2A Prog Interface 0
    00   00   05    04 ==> Base System Peripherals - PIC
              Vendor 8086 Device 6F2C Prog Interface 20
    00   00   14    00 ==> Serial Bus Controllers - USB
              Vendor 8086 Device 8C31 Prog Interface 30
    00   00   16    00 ==> Simple Communications Controllers - Other communi
cation device
              Vendor 8086 Device 8C3A Prog Interface 0
    00   00   16    01 ==> Simple Communications Controllers - Other communi
cation device
              Vendor 8086 Device 8C3B Prog Interface 0
    00   00   1C    00 ==> Bridge Device - PCI/PCI bridge
              Vendor 8086 Device 8C10 Prog Interface 0
    00   00   1C    02 ==> Bridge Device - PCI/PCI bridge
              Vendor 8086 Device 8C14 Prog Interface 0
    00   00   1C    04 ==> Bridge Device - PCI/PCI bridge
              Vendor 8086 Device 8C18 Prog Interface 0
    00   00   1C    07 ==> Bridge Device - PCI/PCI bridge
              Vendor 8086 Device 8C1E Prog Interface 0
    00   00   1F    00 ==> Bridge Device - PCI/ISA bridge
              Vendor 8086 Device 8C54 Prog Interface 0
    00   00   1F    02 ==> Mass Storage Controller - UNDEFINED
              Vendor 8086 Device 8C02 Prog Interface 1
    00   00   1F    03 ==> Serial Bus Controllers - System Management Bus
              Vendor 8086 Device 8C22 Prog Interface 0
    00   01   00    00 ==> Mass Storage Controller - UNDEFINED
              Vendor 1D79 Device 2263 Prog Interface 2
```

```
    00   03   00    00 ==> Base System Peripherals - Other system peripheral
              Vendor 8086 Device 6F50 Prog Interface 0
    00   03   00    01 ==> Base System Peripherals - Other system peripheral
              Vendor 8086 Device 6F51 Prog Interface 0
    00   03   00    02 ==> Base System Peripherals - Other system peripheral
              Vendor 8086 Device 6F52 Prog Interface 0
    00   03   00    03 ==> Base System Peripherals - Other system peripheral
              Vendor 8086 Device 6F53 Prog Interface 0
    00   04   00    00 ==> Network Controller - Ethernet controller
              Vendor 8086 Device 15AB Prog Interface 0
    00   07   00    00 ==> Bridge Device - PCI/PCI bridge
              Vendor 12D8 Device 2608 Prog Interface 0
    00   08   01    00 ==> Bridge Device - PCI/PCI bridge
              Vendor 12D8 Device 2608 Prog Interface 0
    00   08   02    00 ==> Bridge Device - PCI/PCI bridge
              Vendor 12D8 Device 2608 Prog Interface 0
    00   08   03    00 ==> Bridge Device - PCI/PCI bridge
              Vendor 12D8 Device 2608 Prog Interface 0
    00   08   04    00 ==> Bridge Device - PCI/PCI bridge
              Vendor 12D8 Device 2608 Prog Interface 0
    00   08   05    00 ==> Bridge Device - PCI/PCI bridge
              Vendor 12D8 Device 2608 Prog Interface 0
    00   0D   00    00 ==> Simple Communications Controllers - Serial contro
ller
              Vendor 13A8 Device 0358 Prog Interface 2
    00   11   00    00 ==> Network Controller - Ethernet controller
              Vendor 8086 Device 1533 Prog Interface 0
FPGA Version 0x4020046
DRAM size 16 MB
BIOS Setup Checksum : 131293
BIOS Version : S1901R121
ETH Dev 0 (COMe GbE0) connected device speed 1000Mb/s
      3 Hubs
USB Dev 1 connected
USB Dev 3 connected
USB Dev 5 connected
USB Dev 10 connected
USB Dev 11 connected
USB Dev 12 connected

Number of System Test Elements detected : 42
DRAM area [ 0x7313D220  0x7313E328 ] will be stored in EEPROM
Storing system infos...

Storing system configuration...
Shell>
```

Configure default run for PBIT:

```
Shell> kdiag default
```

Clear all previous results:

```
Shell> kdiag clrallstat
```

Run PBIT:

```
Shell> kdiag run
PBIT "mem_data" (fast,simple)  PASSED
PBIT "mem_addr" (fast,simple)  PASSED
PBIT "mem_pattern1" (fast,simple)  PASSED
PBIT "mem_pattern2" (fast,simple)  PASSED
PBIT "mem_pattern3" (fast,simple)  PASSED
PBIT "mem_pattern4" (fast,simple)  PASSED
PBIT "cpu_dmi" (fast,simple)  PASSED
PBIT "tpm" (fast,simple)  PASSED
PBIT "com2" (fast,simple)  PASSED
PBIT "rtc" (fast,simple)  PASSED
PBIT "cpld" (fast,simple)  PASSED
PBIT "smbus" (fast,simple)  PASSED
PBIT "hwmon" (fast,simple)  PASSED
PBIT "jida_eeprom" (fast,simple)  PASSED
PBIT "vpd" (fast,simple)  PASSED
PBIT "secure_chip" (fast,simple)  PASSED
PBIT "gbe0_loop" (slow,simple)  PASSED
PBIT "cgbe1_loop" (slow,simple)  PASSED
PBIT "sata_ctrl" (fast,simple)  PASSED
PBIT "xhci_ctrl" (fast,simple)  PASSED
PBIT "system" (fast,simple)
PCI...  HWCONF... SATA...  SMBUS...  BIOS...  ETH...  USB...
PASSED
```

Check PBIT results:

```
Shell> kdiag stat
Status of PBITs configured to run from command line :
 PASSED : mem_data(1) (fast,simple)
 PASSED : mem_addr(2) (fast,simple)
 PASSED : mem_pattern1(6) (fast,simple)
 PASSED : mem_pattern2(7) (fast,simple)
 PASSED : mem_pattern3(8) (fast,simple)
 PASSED : mem_pattern4(9) (fast,simple)
 PASSED : cpu_dmi(10) (fast,simple)
 PASSED : tpm(11) (fast,simple)
 PASSED : com2(12) (fast,simple)
 PASSED : rtc(13) (fast,simple)
 PASSED : cpld(14) (fast,simple)
 PASSED : smbus(15) (fast,simple)
 PASSED : hwmon(16) (fast,simple)
 PASSED : jida_eeprom(17) (fast,simple)
 PASSED : vpd(18) (fast,simple)
 PASSED : secure_chip(20) (fast,simple)
 PASSED : gbe0_loop(50) (slow,simple)
 PASSED : cgbe1_loop(52) (slow,simple)
 PASSED : sata_ctrl(68) (fast,simple)
 PASSED : xhci_ctrl(78) (fast,simple)
 PASSED : system(97) (fast,simple)

RUN    : 21
PASSED : 21
FAILED : 0
NOT_RUN: 0

Shell>
```

Automated Self-Test Mode

Setting UEFI environment variables allows PBIT to be run automatically at the end of the POST and before booting of the Operating System:
Configure PBIT to be launched at boot time:
The automatic start is activated using the environment variable " ***BootCmd*** ".

```
Shell> set BootCmd ”kdiag run”
```

The delay before executing the " ***BootCmd*** " is given by the environment variable " ***BootDelay*** ".
The delay is expressed in seconds.
Default value is "1" for one second.
Value "0" is possible to disable delay.

```
Shell> set BootDelay 1
```

Display UEFI environment variables:

```
Shell> set
 BootCmd = kdiag run
BootDelay = 1
    path = .\;FS0:\efi\tools\;FS0:\efi\boot\;FS0:\;FS1:\efi\tools\;FS1:\efi\
boot\;FS1:\
lasterror = 0x0
profiles = ;Install;Debug1;Driver1;network1;
uefishellsupport = 3
uefishellversion = 2.0
uefiversion = 2.40
```

Then reset the system to have the effect on next boot:

```
Shell> reset
...
UEFI Interactive Shell v2.0
EDK II
UEFI v2.40 (American Megatrends, 0x0005000B)
Mapping table
      FS0: Alias(s):HD3s0b:;BLK1:
          PciRoot(0x0)/Pci(0x14,0x0)/USB(0x12,0x0)/HD(1,MBR,0x1097C1A0,0x3F,
0x1CE7FC1)
      FS1: Alias(s):HD6b65535a1:;BLK6:
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x1,0xFFFF,0x0)/HD(1,GPT,061B34B2-
3250-42E0-A82A-2AD2E3C662A1,0x800,0x100000)
     BLK0: Alias(s):
          PciRoot(0x0)/Pci(0x14,0x0)/USB(0x12,0x0)
     BLK2: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0xFFFF,0x0)
     BLK5: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x1,0xFFFF,0x0)
     BLK3: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0xFFFF,0x0)/HD(1,MBR,0x000BD2C
5,0x800,0xFA000)
     BLK4: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0xFFFF,0x0)/HD(2,MBR,0x000BD2C
5,0xFA800,0x1291F000)
     BLK7: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x1,0xFFFF,0x0)/HD(2,GPT,D4995729-
5EAE-4F44-BFB6-E4334EE6323C,0x100800,0x1A2B800)
     BLK8: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x1,0xFFFF,0x0)/HD(3,GPT,429FAE77-
8666-4D82-87E9-36D354399484,0x1B2C000,0x1FD3800)
Boot command is present : kdiag run
Press ESC in 1 seconds to skip boot string or any other key to continue.
PBIT "mem_data" (fast,simple)  PASSED
PBIT "mem_addr" (fast,simple)  PASSED
PBIT "mem_pattern1" (fast,simple)  PASSED
PBIT "mem_pattern2" (fast,simple)  PASSED
PBIT "mem_pattern3" (fast,simple)  PASSED
PBIT "mem_pattern4" (fast,simple)  PASSED
PBIT "cpu_dmi" (fast,simple)  PASSED
PBIT "tpm" (fast,simple)  PASSED
PBIT "com2" (fast,simple)  PASSED
PBIT "rtc" (fast,simple)  PASSED
PBIT "cpld" (fast,simple)  PASSED
PBIT "smbus" (fast,simple)  PASSED
PBIT "hwmon" (fast,simple)  PASSED
PBIT "jida_eeprom" (fast,simple)  PASSED
PBIT "vpd" (fast,simple)  PASSED
PBIT "secure_chip" (fast,simple)  PASSED
PBIT "gbe0_loop" (slow,simple)  PASSED
PBIT "cgbe1_loop" (slow,simple)  PASSED
PBIT "sata_ctrl" (fast,simple)  PASSED
PBIT "xhci_ctrl" (fast,simple)  PASSED
PBIT "system" (fast,simple)
PCI...  HWCONF... SATA...  SMBUS...  BIOS...  ETH...  USB...
PASSED
```

PBIT will be launched automatically.
When finished, the BIOS boots from the next available device in the boot order list defined in the Setup.

```
...
Ubuntu 20.04.1 LTS s1901-evotrac ttyS0

s1901-evotrac login:
```

Configure PBIT to stop under UEFI shell

To stop under the UEFI shell after PBIT execution, you have to set the environment variable " ***StopEfiShell*** " to value '1':

```
Shell> set StopEfiShell 1
```

After PBIT have run, the prompt of the UEFI shell is displayed.

PBIT Tests List

PBIT tests list comes in two parts: a default list of selected tests and a list of additional tests that are not selected. This list of tests can be changed by customer to fulfill his specific coverage and execution time requirements. The " ***kdiag*** " command displays both lists.
Note:
The initial default tests list can be restored with the " ***kdiag default*** " command.
Each PBIT test has a number associated to the name.
For instance, the " ***mem_data*** **"** test is associated with the number " **1** ".
Each PBIT test can be run with either the name of the test or with its number.
For example :
The command " ***kdiag run 1*** " is equivalent to the command " ***kdiag run mem_data*** ".

Kontron's Default Tests

The default tests list contains all the diagnostics that can be run without any specific equipment or hardware. All the tests have been designed to be safe for the system board.
No signal on any connector is modified during the default test execution.
Hereunder is the list of tests available on the **COMe-bBD7** module displayed by the " ***kdiag*** " command:

```
Shell> kdiag
PBITs configured to run from command line :
 mem_data (1) - Checks Memory/ECC data lines
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,allresets
 mem_addr (2) - Checks Memory/ECC address lines
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,allresets
 mem_pattern1 (6) - Checks Memory/ECC using pattern 0xFFFFFFFF
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,allresets
 mem_pattern2 (7) - Checks Memory/ECC using pattern 0x55555555
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,allresets
 mem_pattern3 (8) - Checks Memory/ECC using pattern 0xAAAAAAAA
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,allresets
 mem_pattern4 (9) - Checks Memory/ECC using pattern 0x00000000
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,allresets
 cpu_dmi (10) - Checks SKU and DMI link
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,allresets
 tpm (11) - Checks TPM access and vendorID/deviceID
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,allresets
 com2 (12) - Checks the FPGA COM2 serial line
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,allresets
 rtc (13) - Checks the RTC time
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,allresets
 cpld (14) - Checks CPLD registers, watchdog
```

```
 cpld (14) - Checks CPLD registers, watchdog
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,allresets
 smbus (15) - Probe devices on SMBUS (internal only)
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,allresets
 hwmon (16) - Checks HW monitoring
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,allresets
 jida_eeprom (17) - Checks JIDA EEPROM (0xA0) access
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,allresets
 vpd (18) - Checks product data (VPD)
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,allresets
 gbe0_loop (50) - Checks 1GbE interface 0 on COMe
   capabilities : slow/fast,simple,allresets
   run mode 1   : slow,simple,allresets
 10gbe0_loop (55) - Checks 10GbE interface 0
   capabilities : slow/fast,simple,allresets
   run mode 1   : slow,simple,allresets
 sata0_ctrl (68) - Checks SATA controller 0 (D31:F2)
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,allresets
 xhci_ctrl (71) - Check xHCI controller (D20:F0)
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,allresets
 system (97) - Checks system configuration SETUP,PCIe,SATA,USB,ETH stability
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,allresets
```

Advanced Tests

The second part of the list includes all the tests not currently selected for execution. These tests appear at the end of the " ***kdiag*** " command after the message "Other PBITs available but not yet configured":

```
Other PBITs available but not yet configured :

 mem_bitflip (3) - Checks Mem/ECC using bit-flip pattern ((1 << (offset % 64
))
   capabilities : slow/fast,simple,allresets
 mem_addrpat (4) - Checks Memory/ECC using address pattern (offset)
   capabilities : slow/fast,simple,allresets
 mem_addrpat2 (5) - Checks Memory/ECC using address pattern (~offset)
   capabilities : slow/fast,simple,allresets
 pcie0_dev_see (20) - Checks if a device is connected to PCH PCIe Root Port
#0
   capabilities : fast,simple,allresets
 pcie2_dev_see (22) - Checks if a device is connected to PCH PCIe Root Port
#2
   capabilities : fast,simple,allresets
 pcie4_dev_see (24) - Checks if a device is connected to PCH PCIe Root Port
#4
   capabilities : fast,simple,allresets
 pcie7_dev_see (27) - Checks if a device is connected to PCH PCIe Root Port
#7
   capabilities : fast,simple,allresets
 pcie3a_dev_see (30) - Checks if a device is connected to CPU PCIe Root Port
 #3a
```

```
  capabilities : fast,simple,allresets
pcie3c_dev_see (32) - Checks if a device is connected to CPU PCIe Root Port
#3c
  capabilities : fast,simple,allresets
pcie1a_dev_see (34) - Checks if a device is connected to CPU PCIe Root Port
#1a
  capabilities : fast,simple,allresets
pcie1b_dev_see (35) - Checks if a device is connected to CPU PCIe Root Port
#1b
  capabilities : fast,simple,allresets
x_pcie1_dev_see (37) - Checks if a device is connected to EXT PCIe Port #1
  capabilities : fast,simple,allresets
x_pcie2_dev_see (38) - Checks if a device is connected to EXT PCIe Port #2
  capabilities : fast,simple,allresets
x_pcie3_dev_see (39) - Checks if a device is connected to EXT PCIe Port #3
  capabilities : fast,simple,allresets
x_pcie4_dev_see (40) - Checks if a device is connected to EXT PCIe Port #4
  capabilities : fast,simple,allresets
x_pcie5_dev_see (41) - Checks if a device is connected to EXT PCIe Port #5
  capabilities : fast,simple,allresets
x_pcie6_dev_see (42) - Checks if a device is connected to EXT PCIe Port #6
  capabilities : fast,simple,allresets
x_pcie7_dev_see (43) - Checks if a device is connected to EXT PCIe Port #7
  capabilities : fast,simple,allresets
cgbe1_loop (52) - Checks 1GbE interface 1 on Carrier Board
  capabilities : slow/fast,simple,allresets
cgbe2_loop (53) - Checks additional 1GbE interface 2
  capabilities : slow/fast,simple,allresets
sata0_dev_see (60) - Checks device is present on CPU SATA#0 (COMe SATA0)
  capabilities : fast,simple,allresets
sata2_dev_see (62) - Checks device is present on CPU SATA#2 (COMe SATA1)
  capabilities : fast,simple,allresets
usb0_dev_see (72) - Check device is present on USB 2.0 port #0
  capabilities : fast,simple,allresets
usb1_dev_see (73) - Check device is present on USB 2.0 port #1
  capabilities : fast,simple,allresets
usb2_dev_see (74) - Check device is present on USB 2.0 port #2
  capabilities : fast,simple,allresets
usb3_dev_see (75) - Check device is present on USB 2.0 port #3
  capabilities : fast,simple,allresets
usbss0_dev_see (76) - Check device is present on USB 3.0 port #0
  capabilities : fast,simple,allresets
usbss2_dev_see (78) - Check device is present on USB 3.0 port #2
  capabilities : fast,simple,allresets
x3_usb0_dev_see (80) - Check device is present on USB 2.0 Hub 3 port #0
  capabilities : slow/fast,simple,allresets
x3_usb1_dev_see (81) - Check device is present on USB 2.0 Hub 3 port #1
  capabilities : slow/fast,simple,allresets
x3_usb2_dev_see (82) - Check device is present on USB 2.0 Hub 3 port #2
  capabilities : slow/fast,simple,allresets
x3_usb3_dev_see (83) - Check device is present on USB 2.0 Hub 3 port #3
  capabilities : slow/fast,simple,allresets
x1_usb0_dev_see (84) - Check device is present on USB 2.0 Hub 1 port #0
  capabilities : slow/fast,simple,allresets
x1_usb1_dev_see (85) - Check device is present on USB 2.0 Hub 1 port #1
  capabilities : slow/fast,simple,allresets
x1_usb2_dev_see (86) - Check device is present on USB 2.0 Hub 1 port #2
  capabilities : slow/fast,simple,allresets
```

```
  faultytest (98) - A dummy test that returns FAIL
    capabilities : fast,simple,allresets
  hangtest (99) - A dummy test that will hang
    capabilities : fast,simple,allresets

Use "help kdiag" to get more info.

Shell>
```

PBIT Command Line Reference Guide

PBITs are configured and executed using the UEFI Shell command " ***kdiag*** ".
The following section describes the various " ***kdiag*** " command parameters.

On-line Help

At UEFI Shell prompt, enter the command " ***help kdiag*** " to display usage of the " ***kdiag*** " command.
The command formats are:

- [ ] meaning optional parameters
- | meaning a OR choice between possible parameters
- ... meaning an undetermined number of repeated previous parameters

```
Shell> help kdiag
Kontron board diagnostics command
Print list of PBITs and infos about them :
  kdiag [<PBITname>|<PBITnum> ...]
    <PBITname>|<PBITnum> ...
             list of PBIT(s) to display. All if the list is empty.
             PBIT(s) are referenced using their name or their number.
Run PBIT(s) from command line :
  kdiag run [loop <count>] [<PBITname>|<PBITnum> ...]
    loop <count>
             run PBIT(s) <count> times instead of once
    <PBITname>|<PBITnum> ...
             list of PBIT(s) to run. All if the list is empty.
             PBIT(s) are referenced using their name or their number.
Print PBIT(s) status :
  kdiag stat [<PBITname>|<PBITnum> ...]
    <PBITname>|<PBITnum> ...
             list of PBIT(s) to display. All if the list is empty.
             PBIT(s) are referenced using their name or their number.
Clear PBIT(s) status :
  kdiag clrstat|clrallstat [<PBITname>|<PBITnum> ...]
    clrstat    : Reset status to NOTRUN
    clrallstat : Reset status to NOTRUN and clear the (FAILED once) flag
    <PBITname>|<PBITnum> ...
             list of PBIT(s) to clear. All if the list is empty.
             PBIT(s) are referenced using their name or their number.
Restore default PBIT configuration :
  kdiag default
Delete all PBITs from configuration :
  kdiag deleteall
Configure PBIT(s) :

  kdiag cfg <cfgarg> ... [<PBITname>|<PBITnum>] ...
    cfg <cfgarg> : Configure one or several PBIT(s).
                   <cfgarg> is either :
"delete" to delete PBIT(s) from the list of configured PBITs
"default" to configure PBIT(s) with a default run mode
a comma separated list of runflags defining a PBIT
```

```
                  run mode; for example : fast,complex.
                valid runflags are :
"SPEED" flags (can NOT be mixed)
                  slow          : run in slow mode (full testing)
                  fast          : run in fast mode (fast testing)
"CONFIG" flags (can NOT be mixed)
                  simple        : run in simple mode (no external hardware)

                  complex       : run in complex mode (needs external hardw
are)
"HALT" flags (can NOT be mixed)
                  haltonfail    : halt immediately (hang) if test fails
                  promptonfail  : halt at Firmware prompt (no OS boot) if t
est fails
"RESET" flags (can be mixed together)
                  normalreset   : run after a normal reset
                  poweronreset  : run after a power-on reset
                  allresets     : run after all resets listed above
    [<PBITname>|<PBITnum>] ...
                list of PBIT(s) to configure.
                PBIT(s) are referenced using their name or their number.
                All configured tests if the list is empty
Toggle PBIT running log information:
  kdiag silent
Toggle PBIT Bypassed:
  kdiag bypass
Record System Configuration for system Test:
  kdiag system_learn or kdiag learn system
Edit System Configuration for system Test:
 (cf documentation for details)
  kdiag system_edit or kdiag edit system
Clear System Configuration for system Test:
  kdiag system_clear or kdiag clear system
Display PBIT version :
  kdiag version
```

Display the List of Selected Tests

To display the selected tests list and their configuration, use the following command:

```
"kdiag [<PBITname>|<PBITnum> ...]".
```

Running the command " ***kdiag*** " with no argument prints the list of the tests that are selected to run with the command " ***kdiag run*** " and also the other tests not selected by default.

```
Shell> kdiag
PBITs configured to run from command line :
 mem_data (1) - Checks Memory/ECC data lines
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,allresets
 mem_addr (2) - Checks Memory/ECC address lines
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,allresets
 mem_pattern1 (6) - Checks Memory/ECC using pattern 0xFFFFFFFF
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,allresets
 mem_pattern2 (7) - Checks Memory/ECC using pattern 0x55555555
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,allresets
 mem_pattern3 (8) - Checks Memory/ECC using pattern 0xAAAAAAAA
```

```
    capabilities : slow/fast,simple,allresets
    run mode 1   : fast,simple,allresets
  mem_pattern4 (9) - Checks Memory/ECC using pattern 0x00000000
    capabilities : slow/fast,simple,allresets
    run mode 1   : fast,simple,allresets
  cpu_dmi (10) - Checks SKU and DMI link
    capabilities : fast,simple,allresets
    run mode 1   : fast,simple,allresets
  tpm (11) - Checks TPM access and vendorID/deviceID
    capabilities : fast,simple,allresets
    run mode 1   : fast,simple,allresets
  com2 (12) - Checks the FPGA COM2 serial line
    capabilities : fast,simple,allresets
    run mode 1   : fast,simple,allresets
  rtc (13) - Checks the RTC time
    capabilities : fast,simple,allresets
    run mode 1   : fast,simple,allresets
  cpld (14) - Checks CPLD registers, watchdog
    capabilities : fast,simple,allresets
    run mode 1   : fast,simple,allresets
  smbus (15) - Probe devices on SMBUS (internal only)
    capabilities : fast,simple,allresets
    run mode 1   : fast,simple,allresets
  hwmon (16) - Checks HW monitoring
    capabilities : fast,simple/complex,allresets
    run mode 1   : fast,simple,allresets
  jida_eeprom (17) - Checks JIDA EEPROM (0xA0) access
    capabilities : slow/fast,simple,allresets
    run mode 1   : fast,simple,allresets
  vpd (18) - Checks product data (VPD)
    capabilities : fast,simple,allresets
    run mode 1   : fast,simple,allresets
  secure_chip (20) - Check security chip presence on USB #9
    capabilities : fast,simple,allresets
    run mode 1   : fast,simple,allresets
  gbe0_loop (50) - Checks 1GbE interface on COMe
    capabilities : slow/fast,simple,allresets
    run mode 1   : fast,simple,allresets
  cgbe1_loop (52) - Checks 1GbE interface 1 on Carrier Board
    capabilities : slow/fast,simple,allresets
    run mode 1   : fast,simple,allresets
  sata_ctrl (68) - Checks SATA controller (D23:F0)
    capabilities : fast,simple,allresets
    run mode 1   : fast,simple,allresets
  xhci_ctrl (78) - Check xHCI controller (D20:F0)
    capabilities : fast,simple,allresets
    run mode 1   : fast,simple,allresets
  system (97) - Checks system configuration SETUP,PCIe,SATA,USB,ETH stability

    capabilities : fast,simple,allresets
    run mode 1   : fast,simple,allresets

Other PBITs available but not yet configured :

  mem_bitflip (3) - Checks Mem/ECC using bit-flip pattern ((1 << (offset % 64
))
    capabilities : slow/fast,simple,allresets
```

```
 mem_addrpat (4) - Checks Memory/ECC using address pattern (offset)
   capabilities : slow/fast,simple,allresets
 mem_addrpat2 (5) - Checks Memory/ECC using address pattern (~offset)
   capabilities : slow/fast,simple,allresets
 pcie0_dev_see (30) - Checks if a device is connected to PCH PCIe root port
#1
   capabilities : fast,simple,allresets
 pcie1_dev_see (31) - Checks if a device is connected to PCH PCIe root port
#2
   capabilities : fast,simple,allresets
 pcie2_dev_see (32) - Checks if a device is connected to PCH PCIe root port
#3
   capabilities : fast,simple,allresets
 pcie3_dev_see (33) - Checks if a device is connected to PCH PCIe root port
#4
   capabilities : fast,simple,allresets
 pcie4_dev_see (34) - Checks if a device is connected to PCH PCIe root port
#9
   capabilities : fast,simple,allresets
 pcie5_dev_see (35) - Checks if a device is connected to PCH PCIe root port
#10
   capabilities : fast,simple,allresets
 pcie6_dev_see (36) - Checks if a device is connected to PCH PCIe root port
#11
   capabilities : fast,simple,allresets
 pcie7_dev_see (37) - Checks if a device is connected to PCH PCIe root port
#12
   capabilities : fast,simple,allresets
 sata0_dev_see (60) - Checks device is present on SATA port #0
   capabilities : fast,simple,allresets
 sata1_dev_see (61) - Checks device is present on SATA port #1
   capabilities : fast,simple,allresets
 sata2_dev_see (62) - Checks device is present on SATA port #2
   capabilities : fast,simple,allresets
 sata3_dev_see (63) - Checks device is present on SATA port #3
   capabilities : fast,simple,allresets
 usb0_dev_see (70) - Check device is present on USB 2.0 port #0
   capabilities : fast,simple,allresets
 usb1_dev_see (71) - Check device is present on USB 2.0 port #1
   capabilities : fast,simple,allresets
 usb2_dev_see (72) - Check device is present on USB 2.0 port #2
   capabilities : fast,simple,allresets
 usb3_dev_see (73) - Check device is present on USB 2.0 port #3
   capabilities : fast,simple,allresets
 usbss0_dev_see (74) - Check device is present on USB 3.0 port #0
   capabilities : fast,simple,allresets
 usbss1_dev_see (75) - Check device is present on USB 3.0 port #1
   capabilities : fast,simple,allresets
 usbss2_dev_see (76) - Check device is present on USB 3.0 port #2
   capabilities : fast,simple,allresets
 usbss3_dev_see (77) - Check device is present on USB 3.0 port #3
   capabilities : fast,simple,allresets
 otg_ctrl (79) - Check USB OTG controller (D20:F1)
   capabilities : fast,simple,allresets
 gbe0_link (80) - Checks link for 1GbE interface on COMe
   capabilities : fast,simple,allresets
 cgbe1_link (82) - Checks link for 1GbE interface 1 on Carrier Board
   capabilities : fast,simple,allresets
```

```
 faultytest (98) - A dummy test that returns FAIL
   capabilities : fast,simple,allresets
 hangtest (99) - A dummy test that will hang
   capabilities : fast,simple,allresets

Use "help kdiag" to get more info.
```

Execute PBIT from the Command Line

To run PBIT selected tests list from the command line, enter " *kdiag run* ":

```
Shell> kdiag run
PBIT "mem_data" (fast,simple)  PASSED
PBIT "mem_addr" (fast,simple)  PASSED
PBIT "mem_pattern1" (fast,simple)  PASSED
PBIT "mem_pattern2" (fast,simple)  PASSED
PBIT "mem_pattern3" (fast,simple)  PASSED
PBIT "mem_pattern4" (fast,simple)  PASSED
PBIT "cpu_dmi" (fast,simple)  PASSED
PBIT "tpm" (fast,simple)  PASSED
PBIT "com2" (fast,simple)  PASSED
PBIT "rtc" (fast,simple)  PASSED
PBIT "cpld" (fast,simple)  PASSED
PBIT "smbus" (fast,simple)  PASSED
PBIT "hwmon" (fast,simple)  PASSED
PBIT "jida_eeprom" (fast,simple)  PASSED
PBIT "vpd" (fast,simple)  PASSED
PBIT "secure_chip" (fast,simple)  PASSED
PBIT "gbe0_loop" (fast,simple)  PASSED
PBIT "cgbe1_loop" (fast,simple)  PASSED
PBIT "sata_ctrl" (fast,simple)  PASSED
PBIT "xhci_ctrl" (fast,simple)  PASSED
PBIT "system" (fast,simple)
PCI...  HWCONF... SATA...  SMBUS...  BIOS...  ETH...  USB...
PASSED
```

To run a single test or a limited list of tests, enter:

```
Shell> kdiag run <PBIT number | PBIT name ...>
```

For example:

```
Shell> kdiag run xhci_ctrl cpu_dmi
PBIT "xhci_ctrl" (fast,simple)  PASSED
PBIT "cpu_dmi" (fast,simple)  PASSED
This command is equivalent to:
Shell> kdiag run 78 10
PBIT "xhci_ctrl" (fast,simple)  PASSED
PBIT "cpu_dmi" (fast,simple)  PASSED
```

The "PBIT number" is displayed by the " *kdiag* " command (with no parameter) or with the " *kdiag <PBIT name>* " command.

Execute PBIT in Loop Mode

To run PBIT in loop mode, enter:

```
Shell> kdiag run loop <count>
```

with **<count>** being the number of loop to execute.
To run a single test in loop mode, enter:

```
Shell> kdiag run loop <count> <PBIT number | PBIT name ...>
```

- Example:

To run 10 times the test named "gbe0_loop" number 50, enter the command:

```
Shell> kdiag run loop 10 50
```

or:

```
Shell> kdiag run 50 loop 10
Those commands are equivalent to:
Shell> kdiag run loop 10 gbe0_loop
```

or:

```
Shell> kdiag run loop 10 gbe0_loop
```

Get PBIT Results

To get PBIT results, use the " ***kdiag stat*** " command:

```
Shell> kdiag stat
Status of PBITs configured to run from command line :
 PASSED : mem_data(1) (fast,simple)
 PASSED : mem_addr(2) (fast,simple)
 PASSED : mem_pattern1(6) (fast,simple)
 PASSED : mem_pattern2(7) (fast,simple)
 PASSED : mem_pattern3(8) (fast,simple)
 PASSED : mem_pattern4(9) (fast,simple)
 PASSED : cpu_dmi(10) (fast,simple)
 PASSED : tpm(11) (fast,simple)
 PASSED : com2(12) (fast,simple)
 PASSED : rtc(13) (fast,simple)
 PASSED : cpld(14) (fast,simple)
 PASSED : smbus(15) (fast,simple)
 PASSED : hwmon(16) (fast,simple)
 PASSED : jida_eeprom(17) (fast,simple)
 PASSED : vpd(18) (fast,simple)
 PASSED : secure_chip(20) (fast,simple)
 PASSED : gbe0_loop(50) (fast,simple)
 PASSED : cgbe1_loop(52) (fast,simple)
 PASSED : sata_ctrl(68) (fast,simple)
 PASSED : xhci_ctrl(78) (fast,simple)
 PASSED : system(97) (fast,simple)

RUN    : 21
PASSED : 21
FAILED : 0
NOT_RUN: 0
```

Clear PBIT Results

Upon failure of any test, a specific " FAILED ONCE " flag is set. This flag is kept even if this test is successfully PASSED later. This feature has been designed to keep track of sporadic failures.

- To clear PBIT results enter:

```
Shell> kdiag clrstat
```

- To clear all PBIT history including the " FAILED ONCE " flags, enter:

```
Shell> kdiag clrallstat
```

Configure PBIT Tests List to Execute

The list of tests to execute can be modified with the " ***kdiag*** " command.
Each test can be added, removed and configured with a specific run mode. If no run mode is specified then the default run mode (fast,simple) is applied.

Run mode parameters

The possible specific run modes are defined with the following test flags:

- HALT flag (can NOT be mixed):

haltonfail : halt immediately (hang) if test fails
promptonfail : halt at BIOS prompt (no OS boot) if test fails
This flag offers the possibility to halt all test execution when an error is detected.

- SPEED flag (can NOT be mixed):

slow : run in slow mode (full testing)
fast : run in fast mode (fast testing)

- CONFIG flag (can NOT be mixed):

simple : run in simple mode (no external hardware)
complex : run in complex mode (needs external hardware)

- RESET flag:

normalreset : run after a normal reset (means a board warm boot).
poweronreset : run after a power-on reset (PBIT will run only after a board power-on or a cold boot)
allresets : run after all resets listed above

Adding a Test to the Current Run List

To add a test to the run list, enter:

```
Shell> kdiag cfg <cfgarg> ... <PBITname>|<PBITnum> ...
```

" **cfgarg** " allows to choose the test run mode.
Use the keyword " **default** " to set the default mode (typically fast and simple).
To add test 80 (gbe0_link) with default mode, enter:

```
Shell> kdiag cfg default 80
```

To add this test with the promptonfail flag (and the other default flags), enter:

```
Shell> kdiag cfg promptonfail 80
```

Execute the command " **kdiag 80** " to check the configuration:

```
Shell> kdiag 80
 gbe0_link (80) - Checks link for 1GbE interface on COMe
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,allresets

Shell> kdiag run 80
PBIT "gbe0_link" (fast,simple,promptonfail)   Link speed 1000Mb/s PASSED
```

Removing a Test from the Current Run List

For example, to remove test number 80, enter:

```
Shell> kdiag cfg delete 80
```

Verify:

```
Shell> kdiag 80
```

PBIT "80" is not configured to run from command line
To remove all the tests from the current run list, enter:

```
Shell> kdiag deletall
```

Don't forget to add some tests to the current list before running " **kdiag run** ":

```
Shell> kdiag run
WARNING: No PBIT will be run because run list is empty
```

Set a RUN mode parameter to All the Tests of the Current Run List

The command " **kdiag cfg <runflag>** " allows to set a "runflag" (HALT, SPEED, CONFIG, RESET) described in the previous section to all current run list tests.

- For example to set "default" run mode i.e "fast,simple,allresets" for all tests of current run list:

```
Shell> kdiag cfg default
Configuration of current tests in asked mode:
command,fast,simple,allresets
List of test to configure
--> mem_data (1) - Checks Memory/ECC data lines
--> mem_addr (2) - Checks Memory/ECC address lines
--> mem_pattern1 (6) - Checks Memory/ECC using pattern 0xFFFFFFFF
--> mem_pattern2 (7) - Checks Memory/ECC using pattern 0x55555555
--> mem_pattern3 (8) - Checks Memory/ECC using pattern 0xAAAAAAAA
--> mem_pattern4 (9) - Checks Memory/ECC using pattern 0x00000000
--> cpu_dmi (10) - Checks SKU and DMI link
--> tpm (11) - Checks TPM access and vendorID/deviceID
--> com2 (12) - Checks the FPGA COM2 serial line
--> rtc (13) - Checks the RTC time
--> cpld (14) - Checks CPLD registers, watchdog
--> smbus (15) - Probe devices on SMBUS (internal only)
--> hwmon (16) - Checks HW monitoring
--> jida_eeprom (17) - Checks JIDA EEPROM (0xA0) access
--> vpd (18) - Checks product data (VPD)
--> secure_chip (20) - Check security chip presence on USB #9
--> gbe0_loop (50) - Checks 1GbE interface on COMe
--> cgbe1_loop (52) - Checks 1GbE interface 1 on Carrier Board
--> sata_ctrl (68) - Checks SATA controller (D23:F0)
--> xhci_ctrl (78) - Check xHCI controller (D20:F0)
--> system (97) - Checks system configuration SETUP,PCIe,SATA,USB,ETH stabil
ity
```

- For example to set promptonfail, poweronreset run mode for all tests of current run list:

```
Shell> kdiag cfg promptonfail,poweronreset
Configuration of current tests in asked mode:
command,fast,simple,promptonfail,poweronreset
List of test to configure
--> mem_data (1) - Checks Memory/ECC data lines
--> mem_addr (2) - Checks Memory/ECC address lines
--> mem_pattern1 (6) - Checks Memory/ECC using pattern 0xFFFFFFFF
--> mem_pattern2 (7) - Checks Memory/ECC using pattern 0x55555555
--> mem_pattern3 (8) - Checks Memory/ECC using pattern 0xAAAAAAAA
--> mem_pattern4 (9) - Checks Memory/ECC using pattern 0x00000000
--> cpu_dmi (10) - Checks SKU and DMI link
--> tpm (11) - Checks TPM access and vendorID/deviceID
--> com2 (12) - Checks the FPGA COM2 serial line
--> rtc (13) - Checks the RTC time
--> cpld (14) - Checks CPLD registers, watchdog
--> smbus (15) - Probe devices on SMBUS (internal only)
--> hwmon (16) - Checks HW monitoring
--> jida_eeprom (17) - Checks JIDA EEPROM (0xA0) access
--> vpd (18) - Checks product data (VPD)
--> secure_chip (20) - Check security chip presence on USB #9
--> gbe0_loop (50) - Checks 1GbE interface on COMe
--> cgbe1_loop (52) - Checks 1GbE interface 1 on Carrier Board
--> sata_ctrl (68) - Checks SATA controller (D23:F0)
--> xhci_ctrl (78) - Check xHCI controller (D20:F0)
--> system (97) - Checks system configuration SETUP,PCIe,SATA,USB,ETH stabil
ity
Shell>
```

Verify with "**kdiag**" command, the run mode is promptonfail, poweronreset for all tests in the run list:

```
Shell> kdiag
PBITs configured to run from command line :
 mem_data (1) - Checks Memory/ECC data lines
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 mem_addr (2) - Checks Memory/ECC address lines
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 mem_pattern1 (6) - Checks Memory/ECC using pattern 0xFFFFFFFF
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 mem_pattern2 (7) - Checks Memory/ECC using pattern 0x55555555
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 mem_pattern3 (8) - Checks Memory/ECC using pattern 0xAAAAAAAA
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 mem_pattern4 (9) - Checks Memory/ECC using pattern 0x00000000
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 cpu_dmi (10) - Checks SKU and DMI link
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 tpm (11) - Checks TPM access and vendorID/deviceID
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 com2 (12) - Checks the FPGA COM2 serial line
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 rtc (13) - Checks the RTC time
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 cpld (14) - Checks CPLD registers, watchdog
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 smbus (15) - Probe devices on SMBUS (internal only)
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 hwmon (16) - Checks HW monitoring
   capabilities : fast,simple/complex,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 jida_eeprom (17) - Checks JIDA EEPROM (0xA0) access
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 vpd (18) - Checks product data (VPD)
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 secure_chip (20) - Check security chip presence on USB #9
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 gbe0_loop (50) - Checks 1GbE interface on COMe
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 cgbe1_loop (52) - Checks 1GbE interface 1 on Carrier Board
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 sata_ctrl (68) - Checks SATA controller (D23:F0)
   capabilities : fast,simple,allresets
```

```
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 xhci_ctrl (78) - Check xHCI controller (D20:F0)
   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 system (97) - Checks system configuration SETUP,PCIe,SATA,USB,ETH stability

   capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset

Other PBITs available but not yet configured :

 mem_bitflip (3) - Checks Mem/ECC using bit-flip pattern ((1 << (offset % 64
))
   capabilities : slow/fast,simple,allresets
 mem_addrpat (4) - Checks Memory/ECC using address pattern (offset)
   capabilities : slow/fast,simple,allresets
 mem_addrpat2 (5) - Checks Memory/ECC using address pattern (~offset)
   capabilities : slow/fast,simple,allresets
 pcie0_dev_see (30) - Checks if a device is connected to PCH PCIe root port
#1
   capabilities : fast,simple,allresets
 pcie1_dev_see (31) - Checks if a device is connected to PCH PCIe root port
#2
   capabilities : fast,simple,allresets
 pcie2_dev_see (32) - Checks if a device is connected to PCH PCIe root port
#3
   capabilities : fast,simple,allresets
 pcie3_dev_see (33) - Checks if a device is connected to PCH PCIe root port
#4
   capabilities : fast,simple,allresets
 pcie4_dev_see (34) - Checks if a device is connected to PCH PCIe root port
#9
   capabilities : fast,simple,allresets
 pcie5_dev_see (35) - Checks if a device is connected to PCH PCIe root port
#10
   capabilities : fast,simple,allresets
 pcie6_dev_see (36) - Checks if a device is connected to PCH PCIe root port
#11
   capabilities : fast,simple,allresets
 pcie7_dev_see (37) - Checks if a device is connected to PCH PCIe root port
#12
   capabilities : fast,simple,allresets
 sata0_dev_see (60) - Checks device is present on SATA port #0
   capabilities : fast,simple,allresets
 sata1_dev_see (61) - Checks device is present on SATA port #1
   capabilities : fast,simple,allresets
 sata2_dev_see (62) - Checks device is present on SATA port #2
   capabilities : fast,simple,allresets
 sata3_dev_see (63) - Checks device is present on SATA port #3
   capabilities : fast,simple,allresets
 usb0_dev_see (70) - Check device is present on USB 2.0 port #0
   capabilities : fast,simple,allresets
 usb1_dev_see (71) - Check device is present on USB 2.0 port #1
   capabilities : fast,simple,allresets
 usb2_dev_see (72) - Check device is present on USB 2.0 port #2
   capabilities : fast,simple,allresets
 usb3_dev_see (73) - Check device is present on USB 2.0 port #3
```

```
    capabilities : fast,simple,allresets
  usbss0_dev_see (74) - Check device is present on USB 3.0 port #0
    capabilities : fast,simple,allresets
  usbss1_dev_see (75) - Check device is present on USB 3.0 port #1
    capabilities : fast,simple,allresets
  usbss2_dev_see (76) - Check device is present on USB 3.0 port #2
    capabilities : fast,simple,allresets
  usbss3_dev_see (77) - Check device is present on USB 3.0 port #3
    capabilities : fast,simple,allresets
  otg_ctrl (79) - Check USB OTG controller (D20:F1)
    capabilities : fast,simple,allresets
  gbe0_link (80) - Checks link for 1GbE interface on COMe
    capabilities : fast,simple,allresets
  cgbe1_link (82) - Checks link for 1GbE interface 1 on Carrier Board
    capabilities : fast,simple,allresets
  faultytest (98) - A dummy test that returns FAIL
    capabilities : fast,simple,allresets
  hangtest (99) - A dummy test that will hang
    capabilities : fast,simple,allresets

Use "help kdiag" to get more info.

Shell>
```

Verify for test 1:

```
Shell> kdiag 1
 mem_data (1) - Checks Memory/ECC data lines
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
```

The command will not include the test number for hangtest(98) and faulty(99) because these tests are reserved for PBIT validation.

Quickly configure all tests in specific mode

The fastest method to configure all PBIT in a specific mode (like promptonfail, poweronreset ...) is:

1. delete all tests with command: ***kdiag deleteall***
2. add all default tests with command: ***kdiag default***
3. configure tests with command: ***kdiag cfg <mode1,mode2,...>***

- Example to configure all tests in **poweronreset** and **promptonfail** mode :

```
Shell> kdiag deleteall

Shell> kdiag default

Shell> kdiag cfg promptonfail,poweronreset
Configuration of current tests in asked mode:
command,fast,simple,promptonfail,poweronreset
List of test to configure
--> mem_data (1) - Checks Memory/ECC data lines
--> mem_addr (2) - Checks Memory/ECC address lines
--> mem_pattern1 (6) - Checks Memory/ECC using pattern 0xFFFFFFFF
--> mem_pattern2 (7) - Checks Memory/ECC using pattern 0x55555555
--> mem_pattern3 (8) - Checks Memory/ECC using pattern 0xAAAAAAAA
--> mem_pattern4 (9) - Checks Memory/ECC using pattern 0x00000000
--> cpu_dmi (10) - Checks SKU and DMI link
--> tpm (11) - Checks TPM access and vendorID/deviceID
--> com2 (12) - Checks the FPGA COM2 serial line
--> rtc (13) - Checks the RTC time
--> cpld (14) - Checks CPLD registers, watchdog
--> smbus (15) - Probe devices on SMBUS (internal only)
--> hwmon (16) - Checks HW monitoring
--> jida_eeprom (17) - Checks JIDA EEPROM (0xA0) access
--> vpd (18) - Checks product data (VPD)
--> secure_chip (20) - Check security chip presence on USB #9
--> gbe0_loop (50) - Checks 1GbE interface on COMe
--> cgbe1_loop (52) - Checks 1GbE interface 1 on Carrier Board
--> sata_ctrl (68) - Checks SATA controller (D23:F0)
--> xhci_ctrl (78) - Check xHCI controller (D20:F0)
--> system (97) - Checks system configuration SETUP,PCIe,SATA,USB,ETH stabil
ity

Shell>
```

Verify for test 1:

```
Shell> kdiag 1
mem_data (1) - Checks Memory/ECC data lines
   capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
```

Then don't forget to remove unwanted test with command **kdiag cfg delete <test_name>** or **<test_number>** .

Restore the Default Run List

To restore the default run tests list, enter:

```
Shell> kdiag default
```

Run PBIT in Silent Mode

To avoid PBIT to display test messages during execution, use the toggle command:

```
Shell> kdiag silent
PBIT set in silent mode
```

To disable the silent mode, execute again the same command:

```
Shell> kdiag silent
PBIT silent mode removed
```

1: In this mode, error messages are displayed anyway.
2: To prevent any output messages to the serial line, use the BIOS setup configuration and disable Console Redirection

Display PBIT Version

To display PBIT version, enter:

```
Shell> kdiag version
PBIT VERSION 1.1 ID18165
```

Bypass PBIT Tests

The following toggle command can be used to avoid PBIT to run the tests:

```
Shell> kdiag bypass
PBIT BYPASS set
```

To disable the bypass mode, execute again the same command:

```
Shell> kdiag bypass
PBIT BYPASS removed
```

This is useful to bypass PBIT tests if they are configured to run automatically.
This feature is also accessible through the kdiag Linux utility to enable/disable the tests under OS at the next boot.
Refer to the next section "Linux Kdiag Utility" for details.

Linux kdiag utility

PBIT configuration and results may also be accessible under Linux Operating System delivered with the COMe-bBD7 board through a software API.
It is distributed in the BSP and includes the following features:

- Listing and modifying PBIT test list to be run at the next boot
- Getting and clearing PBIT status
- Bypassing PBIT tests at the next boot

The Linux kdiag command syntax is similar to PBIT kdiag command used under the BIOS EFI shell.

Linux kdiag configuration

The Linux " **kdiag** " utility can be used to configure PBIT directly under Linux OS and available at next boot.
Hereunder is an example of the command:

```
# kdiag help

kdiag    - perform board diagnostics

  ----- Usage -----

Print list of PBITs and infos about them :

  kdiag <PBITname>|<PBITnum> ...

    <PBITname>|<PBITnum> ...
             list of PBIT(s) to display. All if the list is empty.
             PBIT(s) are referenced using their name or their number.


Print PBIT(s) status :

  kdiag stat [<PBITname>|<PBITnum> ...]

    <PBITname>|<PBITnum> ...
             list of PBIT(s) to display. All if the list is empty.
             PBIT(s) are referenced using their name or their number.


Clear PBIT(s) status :

  kdiag clrstat|clrallstat [<PBITname>|<PBITnum> ...]
```

```
    clrstat    : Reset status to NOTRUN
    clrallstat : Reset status to NOTRUN and clear the (FAILED once) flag

    <PBITname>|<PBITnum> ...
              list of PBIT(s) to clear. All if the list is empty.
              PBIT(s) are referenced using their name or their number.


Delete all PBITs from configuration :

  kdiag deleteall


Configure PBIT(s) :

  kdiag cfg <cfgarg> ... <PBITname>|<PBITnum> ...

    cfg <cfgarg> : Configure one or several PBIT(s).

                  <cfgarg> is either :
"delete" to delete PBIT(s) from the list of configured PBITs
"default" to configure PBIT(s) with a default run mode
a comma separated list of runflags defining a PBIT
                    run mode; for example : fast,complex.

                  valid runflags are :
"SPEED" flags (can NOT be mixed)
                    slow          : run in slow mode (full testing)
                    fast          : run in fast mode (fast testing)
"CONFIG" flags (can NOT be mixed)
                    simple        : run in simple mode (no external hardware)
                    complex       : run in complex mode (needs external hardw
are)
"HALT" flags (can NOT be mixed)
                    haltonfail    : halt immediately (hang) if test fails
                    promptonfail  : halt at U-Boot prompt (no OS) if test fai
ls
"RESET" flags (can be mixed together)
                    normalreset   : run after a normal reset
                    poweronreset  : run after a power-on reset
                    allresets     : run after all resets listed above

    [<PBITname>|<PBITnum>] ...
                 list of PBIT(s) to configure.
                 PBIT(s) are referenced using their name or their number.
                 All missing tests if the list is empty (but the cfgarg is d
efault).


Display kdiag command version :

  kdiag version

Toggle PBIT Bypassed:

  kdiag bypass
```

```
# kdiag version
kdiag command Version:  V1.0    17258
```

```
# kdiag
PBITs configured to run from command line :
 mem_data (1) -    capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 mem_addr (2) -    capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 mem_pattern1 (6) -    capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 mem_pattern2 (7) -    capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 mem_pattern3 (8) -    capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 mem_pattern4 (9) -    capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 cpu_dmi (10) -    capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 tpm (11) -    capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 com2 (12) -    capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 rtc (13) -    capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 cpld (14) -    capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 smbus (15) -    capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 hwmon (16) -    capabilities : fast,simple/complex,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 jida_eeprom (17) -    capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 vpd (18) -    capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 secure_chip (20) -    capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 gbe0_loop (50) -    capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 cgbe1_loop (52) -    capabilities : slow/fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 sata_ctrl (68) -    capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 xhci_ctrl (78) -    capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset
 system (97) -    capabilities : fast,simple,allresets
   run mode 1   : fast,simple,promptonfail,poweronreset


Other PBITs available but not yet configured :

 mem_bitflip (3) -    capabilities : slow/fast,simple,allresets
 mem_addrpat (4) -    capabilities : slow/fast,simple,allresets
 mem_addrpat2 (5) -    capabilities : slow/fast,simple,allresets
 pcie0_dev_see (30) -    capabilities : fast,simple,allresets
 pcie1_dev_see (31) -    capabilities : fast,simple,allresets
 pcie2_dev_see (32) -    capabilities : fast,simple,allresets
 pcie3_dev_see (33) -    capabilities : fast,simple,allresets
```

```
 pcie4_dev_see (34) -    capabilities : fast,simple,allresets
 pcie5_dev_see (35) -    capabilities : fast,simple,allresets
 pcie6_dev_see (36) -    capabilities : fast,simple,allresets
 pcie7_dev_see (37) -    capabilities : fast,simple,allresets
 sata0_dev_see (60) -    capabilities : fast,simple,allresets
 sata1_dev_see (61) -    capabilities : fast,simple,allresets
 sata2_dev_see (62) -    capabilities : fast,simple,allresets
 sata3_dev_see (63) -    capabilities : fast,simple,allresets
 usb0_dev_see (70) -    capabilities : fast,simple,allresets
 usb1_dev_see (71) -    capabilities : fast,simple,allresets
 usb2_dev_see (72) -    capabilities : fast,simple,allresets
 usb3_dev_see (73) -    capabilities : fast,simple,allresets
 usbss0_dev_see (74) -    capabilities : fast,simple,allresets
 usbss1_dev_see (75) -    capabilities : fast,simple,allresets
 usbss2_dev_see (76) -    capabilities : fast,simple,allresets
 usbss3_dev_see (77) -    capabilities : fast,simple,allresets
 otg_ctrl (79) -    capabilities : fast,simple,allresets
 gbe0_link (80) -    capabilities : fast,simple,allresets
 cgbe1_link (82) -    capabilities : fast,simple,allresets
 faultytest (98) -    capabilities : fast,simple,allresets
 hangtest (99) -    capabilities : fast,simple,allresets

Use ' kdiag help' to get more info.
```

Linux kdiag results

The Linux "**kdiag stat** " command can be used to display the last PBIT results directly under Linux OS.
Hereunder is an example of the command:

```
# kdiag stat

kdiag stat
Display status
Status of PBITs configured to run from command line :
 PASSED : mem_data (fast,simple)
 PASSED : mem_addr (fast,simple)
 PASSED : mem_pattern1 (fast,simple)
 PASSED : mem_pattern2 (fast,simple)
 PASSED : mem_pattern3 (fast,simple)
 PASSED : mem_pattern4 (fast,simple)
 PASSED : cpu_dmi (fast,simple)
 PASSED : tpm (fast,simple)
 PASSED : com2 (fast,simple)
 PASSED : rtc (fast,simple)
 PASSED : cpld (fast,simple)
 PASSED : smbus (fast,simple)
 PASSED : hwmon (fast,simple)
 PASSED : jida_eeprom (fast,simple)
 PASSED : vpd (fast,simple)
 PASSED : secure_chip (fast,simple)
 PASSED : gbe0_loop (slow,simple)
 PASSED : cgbe1_loop (slow,simple)
 PASSED : sata_ctrl (fast,simple)
 PASSED : xhci_ctrl (fast,simple)
 PASSED : system (fast,simple)

RUN    : 21
PASSED : 21
FAILED : 0
NOT_RUN: 0
```

Linux kdiag bypass

PBIT can be bypassed at next reset of the COMe-bBD7 module using the kdiag command "**kdiag bypass** ".

```
# kdiag bypass
PBIT BYPASS set
# reboot
...
[  137.943011] reboot: Restarting system
...
UEFI Interactive Shell v2.0
EDK II
UEFI v2.40 (American Megatrends, 0x0005000B)
Mapping table
      FS0: Alias(s):HD3t0b:;BLK1:
          PciRoot(0x0)/Pci(0x14,0x0)/USB(0x13,0x0)/HD(1,MBR,0x1097C1A0,0x3F,
0x1CE7FC1)
      FS1: Alias(s):HD6b65535a1:;BLK3:
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x1,0xFFFF,0x0)/HD(1,GPT,061B34B2-
3250-42E0-A82A-2AD2E3C662A1,0x800,0x100000)
     BLK0: Alias(s):
          PciRoot(0x0)/Pci(0x14,0x0)/USB(0x13,0x0)
     BLK2: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x1,0xFFFF,0x0)
     BLK6: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x3,0xFFFF,0x0)
     BLK7: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x3,0xFFFF,0x0)/HD(1,MBR,0x000BD2C
5,0x800,0xFA000)
     BLK8: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x3,0xFFFF,0x0)/HD(2,MBR,0x000BD2C
5,0xFA800,0x1291F000)
     BLK4: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x1,0xFFFF,0x0)/HD(2,GPT,D4995729-
5EAE-4F44-BFB6-E4334EE6323C,0x100800,0x1A2B800)
     BLK5: Alias(s):
          PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x1,0xFFFF,0x0)/HD(3,GPT,429FAE77-
8666-4D82-87E9-36D354399484,0x1B2C000,0x1FD3800)
Press ESC in 3 seconds to skip startup.nsh or any other key to continue.
Shell> kdiag run
WARNING : PBIT bypassed

Shell>
```

PBIT test description

Memory tests

mem_data (1) - Checks Memory/ECC data lines tests the data bits for each address line and for each DDR channel.
mem_addr (2) - Checks Memory/ECC address lines tests the address lines to detect sticky or unconnected bits.
The following tests fill the memory area under test with a specific pattern and verify the read data coherency:
mem_bitflip (3) - Checks Mem/ECC using bit-flip pattern ((1 << (offset % 64))
mem_addrpat (4) - Checks Memory/ECC using address pattern (offset)
mem_addrpat2 (5) - Checks Memory/ECC using address pattern (~offset)
mem_pattern1 (6) - Checks Memory/ECC using pattern 0xFFFFFFFF
mem_pattern2 (7) - Checks Memory/ECC using pattern 0x55555555
mem_pattern3 (8) - Checks Memory/ECC using pattern 0xAAAAAAAA
mem_pattern4 (9) - Checks Memory/ECC using pattern 0x00000000
The memory tests are run on the memory areas that are « available » for the operating system (see the " **memmap** " UEFI shell command output).
The slow and fast modes are implemented for each memory test and the mode determines the part of the memory area to be tested (whole bytes in the memory area for slow mode, bytes/4 for fast mode).
The memory tests include the ECC check (if it is supported and enabled). In case of test failure, it should be worth disabling the ECC and run the test again.
These tests could be reconfigured by the end user according to the time allowed for PBIT tests.
Refer to **§1.6 PBIT Execution Time** to get test durations according to the running mode.

Ethernet loopback tests

gbe0_loop (50) - Checks 1GbE interface on COMe (I219-LM)
cgbe1_loop (52) - Checks 1GbE interface 1 on Carrier Board
The **gbe0_loop** test controls the Intel i219-LM Ethernet Controller located on the COMe-bBD7 and attached to the **PCH PCIe Root Port #5** .
The **cgbe1_loop** test controls the Intel i210 Ethernet Controller located on the carrier board and attached to the **PCH PCIe Root Port #9** .
The "slow" and "fast" modes are implemented. In "fast" mode, the test runs the "standard" diagnostics of the Intel driver that check the controller registers.
In "slow" mode the test also runs the "extended" diagnostics that include an internal loopback test. Default is "slow" mode.

Ethernet Link tests

gbe0_link (80) - Checks link for 1GbE interface on COMe
cgbe1_link (82) - Checks link for 1GbE interface 1 on Carrier Board
Only "simple" mode is implemented on these tests.
These tests check the interface link is up and returns also the link speed for information.
These tests could be configured by the end user depending on the system.

SATA controller test

sata_ctrl (68) - Checks SATA controller (D23:F0)
This test verifies the access to the Mass Storage Controller at PCI Bus 0 Device 23 Function 0.
The test verifies that there is no error logged in the status register.
The test is running with either SATA configured in AHCI or RAID mode in Setup.

SATA device tests

sata0_dev_see (60) - Checks device is present on SATA port #0
sata1_dev_see (61) - Checks device is present on SATA port #1
sata2_dev_see (62) - Checks device is present on SATA port #2
sata3_dev_see (63) - Checks device is present on SATA port #3
Those tests verify that a SATA device is connected and accessible to the corresponding SATA port.
These tests could be configured by the end user depending on the system.

USB controller tests

xhci_ctrl (78) - Check xHCI controller (D20:F0)
otg_ctrl (79) - Check USB OTG controller (D20:F1)
The xhci_ctrl test verifies the access to the xHCI Controller at PCI Bus 0 Device 20 Function 0.
The otg_ctrl test verifies the access to the OTG Controller at PCI Bus 0 Device 20 Function 1.
Those tests verify also no error is logged in the status register.
Depending on the USB configuration in Setup, controllers may be disabled.
By default the USB OTG controller is disabled by BIOS in Setup.
These tests could be reconfigured by the end user depending on the system.

USB device tests

usb0_dev_see (70) - Check device is present on USB 2.0 port #0
usb1_dev_see (71) - Check device is present on USB 2.0 port #1
usb2_dev_see (72) - Check device is present on USB 2.0 port #2
usb3_dev_see (73) - Check device is present on USB 2.0 port #3
usbss0_dev_see (74) - Check device is present on USB 3.0 port #0
usbss1_dev_see (75) - Check device is present on USB 3.0 port #1
usbss2_dev_see (76) - Check device is present on USB 3.0 port #2
usbss3_dev_see (77) - Check device is present on USB 3.0 port #3
Those test tests verify that an USB device is connected and accessible to the corresponding USB port.
Depending of the USB device connected on the USB port, the end user must select the corresponding test for testing USB2.0 or USB3.0.
These tests could be configured by the end user depending on the system.

CPU/DMI test

cpu_dmi (10) - Checks SKU and DMI link
The test checks:

- AES is disabled in the CPU (for export control)
- the number of cores/thread of the CPU is written in SMBIOS table and correspond to the MSR value
- the DMI bus link speed is Gen3
- the DMI bus link width is x4

TPM test

tpm (11) - Checks TPM access
This test checks that the TPM component is accessible.
The test checks also the ID of the TPM component:

- PCI Vendor ID is 0x15D1 for Infineon
- PCI Device ID is 0x001A for SLB9665X model.

TPM is enabled by default by the BIOS in Setup.
Only "simple" mode with above PCI IDs is supported by the test.

PCIe device tests

There are 18 PCIe ports available on the PCH (CM236) of the COMe-bBD7 module but only 8 ports are connected to the COMe-bBD7 connectors following the table:

PCH Root Port Lanes #	PCI BUS Location	COMe connectors PCIE Lane #
PCH Root Port #1	B0:D1C:F0	COMe PCIe #4
PCH Root Port #2	B0:D1C:F1	COMe PCIe #5
PCH Root Port #3	B0:D1C:F2	COMe PCIe #6
PCH Root Port #4	B0:D1C:F3	COMe PCIe #7
PCH Root Port #5	Connected to LAN I219-LM	
PCH Root Port #6-#8	Not connected	
PCH Root Port #9	B0:D1D:F0	COMe PCIe #0
PCH Root Port #10	B0:D1D:F1	COMe PCIe #1
PCH Root Port #11	B0:D1D:F2	COMe PCIe #2
PCH Root Port #12	B0:D1D:F3	COMe PCIe #3
PCH Root Port #13-#16	Used as SATA#0-#3	
PCH Root Port #17-#18	Not connected	

pcie0_dev_see (30) - Checks if a device is connected to PCH PCIe root port #1
pcie1_dev_see (31) - Checks if a device is connected to PCH PCIe root port #2
pcie2_dev_see (32) - Checks if a device is connected to PCH PCIe root port #3
pcie3_dev_see (33) - Checks if a device is connected to PCH PCIe root port #4
pcie4_dev_see (34) - Checks if a device is connected to PCH PCIe root port #9
pcie5_dev_see (35) - Checks if a device is connected to PCH PCIe root port #10
pcie6_dev_see (36) - Checks if a device is connected to PCH PCIe root port #11
pcie7_dev_see (37) - Checks if a device is connected to PCH PCIe root port #12
Each test verifies a PCI device is connected to the corresponding PCH PCIe Root Port.
The test also controls no error is logged in the status register.
These tests could be configured by the end user depending on the system.

COM2 test

com2 (12) - Checks the FPGA COM2 serial line
The serial test is limited to Read/Write some registers of the CPLD UART corresponding to COM2:

- LSR must have no error
- Rcv Buffer Reg
- IER Reg
- ISR Reg
- LCR Reg
- MCR Reg
- LSR Reg
- MSR Reg
- DLL Reg
- DLM Reg
- Update baudrate from 115200 to 9600
- Restore initial values

There is no internal loopback mode available in the UART.

RTC test

rtc (13) - Checks the RTC time
This test controls the PCH date/time validity stored in the RTC.
A WARNING is displayed if the data/time is set to 01/01/2009 that indicates the date/time has not been set.

CPLD test

cpld (14) - Checks CPLD registers, watchdog
The cpld test is limited to control some registers and test the stage1 of the watchdog with a 12-bit presacler set to 500 ms.
The test checks also the CPLD version and is FAILED if the version is a debug version or if the version is not corresponding to Kontron CPLD versioning rules.
Note that the watchdog feature is started/tested when running PBIT tests (excepted in verbose mode).

SMBUS test

smbus (15) - Probe devices on SMBUS (internal only)
The smbus test probes the HW Monitor at SMBus address 0x5C and checks:

- Vendor ID = 0x50
- Device ID = 0xC3

HW monitor test

hwmon (16) - Checks HW monitoring
The hwmon test verifies:

- Nuvoton identifiers for NCT7802Y model (same as in SMBUS test)
- no temperature alert
- the CPU temperature is not above +80°C for commercial range, +100 °C for industrial range
- 3200mV ≤ VCC(+3V3) ≤ 3500 mV
- 2999 mV ≤ VCORE(+3V3 = Batt Volt at COMe pin) ≤ 3499 mV
- 11009 mV ≤ VSEN2(+12V) ≤ 13008 mV
- 4002 mV ≤ VSEN3(+5V) ≤ 6000 mV
- fan connected to CPU is running if CPU temperature > Fan Trip point set in Setup (default is 50°C)

Note that the CPU temperature is read from PECI, and so, this interface is also checked by this test.

VPD and EEPROM tests

jida_eeprom (17) - Checks JIDA EEPROM (0xA0) access
vpd (18) - Checks product data (VPD)
The vpd test controls the coherency of the product data stored in the SMBIOS table 2 and the JIDA eeprom against expected data and verifies the coherency of the variant in SMBIOS Type 2 in field "AssetTag".
In "fast" mode the jida_eeprom test probes the presence of the JIDA EEPROM (PICMG EEPROM) .
In "slow" mode the test performs EEPROM read/writes accesses and in verbose mode, the test dump the content of the JIDA EEPROM (PICMG EEPROM) .

System test

system (97) - Checks system configuration SETUP,PCIe,SATA,USB,ETH stability
The test checks the current system configuration against a referenced configuration previously recorded.
The reference configuration is recorded with the **kdiag learn system** command.
The features controlled by the test can be managed by using the **kdiag edit system** command.
See next chapter for details.

PBIT System

Recording a System Configuration

The command **kdiag learn system** is used to record the current system configuration. It should be run when the system configuration is the correct one to be saved.
It records the:

- Detected PCI devices (the list is visible with the BIOS shell command "pci"),
- PCIe devices infos (VendorID, DeviceID, ClassCode) from the detected PCI devices,
- PCIe link width and speed for PCI/PCI bridge devices,
- SATA information (port connected, device name, device size),
- USB information (port connected, keyboard, mouse, mass storage device names),
- Ethernet information (port link up, speed),
- BIOS information: BIOS ID and checksum (checksum of BIOS setup),
- Hardware Configuration: CPLD Version, Memory size.

Hereunder is an example of the system configuration with a **COMe-bBD7** module along with **Mojave** and **Test SIB** carrier boards:

```
Shell> kdiag learn system
   Seg  Bus  Dev  Func
   ---  ---  ---  ----
    00   00   00    00 ==> Bridge Device - Host/PCI bridge
              Vendor 8086 Device 1918 Prog Interface 0
    00   00   02    00 ==> Display Controller - VGA/8514 controller
              Vendor 8086 Device 191D Prog Interface 0
    00   00   08    00 ==> Base System Peripherals - Other system peripheral
              Vendor 8086 Device 1911 Prog Interface 0
    00   00   14    00 ==> Serial Bus Controllers - USB
              Vendor 8086 Device A12F Prog Interface 30
    00   00   14    02 ==> Data Acquisition & Signal Processing Controllers
- Other DAQ & SP controllers
              Vendor 8086 Device A131 Prog Interface 0
    00   00   16    00 ==> Simple Communications Controllers - Other communi
cation device
              Vendor 8086 Device A13A Prog Interface 0
    00   00   17    00 ==> Mass Storage Controller - Serial ATA controller
              Vendor 8086 Device A102 Prog Interface 1
    00   00   1D    00 ==> Bridge Device - PCI/PCI bridge
              Vendor 8086 Device A118 Prog Interface 0
    00   00   1D    01 ==> Bridge Device - PCI/PCI bridge
              Vendor 8086 Device A119 Prog Interface 0
    00   00   1E    00 ==> Data Acquisition & Signal Processing Controllers
   Other DAQ & SP controllers
```

```
- Other DAQ & SP controllers
            Vendor 8086 Device A127 Prog Interface 0
    00   00   1F    00 ==> Bridge Device - PCI/ISA bridge
            Vendor 8086 Device A150 Prog Interface 0
    00   00   1F    02 ==> Memory Controller - Other memory controller
            Vendor 8086 Device A121 Prog Interface 0
    00   00   1F    03 ==> Multimedia Device - Mixed mode device
            Vendor 8086 Device A170 Prog Interface 0
    00   00   1F    04 ==> Serial Bus Controllers - System Management Bus
            Vendor 8086 Device A123 Prog Interface 0
    00   00   1F    06 ==> Network Controller - Ethernet controller
            Vendor 8086 Device 15B7 Prog Interface 0
    00   01   00    00 ==> Network Controller - Ethernet controller
            Vendor 8086 Device 1533 Prog Interface 0
    00   02   00    00 ==> Simple Communications Controllers - Serial contro
ller
            Vendor 13A8 Device 0354 Prog Interface 2
FPGA Version P108.0069 Release
DRAM size 32 GB
SATA Dev 0 ST9160314AS     (160.0GB)
SATA Dev 1 LDLC            (31.6GB)
Device detected on SMBUS0, address = 0x38
Device detected on SMBUS0, address = 0x42
Device detected on SMBUS0, address = 0x70
Device detected on SMBUS0, address = 0xA0
Device detected on SMBUS0, address = 0xAE
Device detected on SMBUS0, address = 0xC0
Device detected on SMBUS0, address = 0xC2
Device detected on SMBUS0, address = 0xC4
Device detected on SMBUS0, address = 0xC6
Device detected on SMBUS0, address = 0xE0
Device detected on SMBUS0, address = 0xE2
Device detected on SMBUS0, address = 0xE4
BIOS Setup Checksum : 143647
BIOS Version : BBD7R110
ETH Dev 0 (GBE0 COMe) connected device speed 1000Mb/s
ETH Dev 1 (GBE1 on Carrier) connected device speed 1000Mb/s
      2 Drives, 1 Keyboard, 1 Mouse, 1 Hub
MassStorage Device name (0) : UFD 3.0 Silicon-Pow
MassStorage Device name (1) : KingstonDataTravele
USB Dev 0 connected
USB Dev 6 connected

Number of System Test Elements detected : 42
DRAM area [ 0x864332A0  0x864343A8 ] will be stored in EEPROM
Storing system infos...

Storing system configuration...
Shell>
```

Checking the Current System Configuration

PBIT system test is used to get the current system configuration and check it against the recorded configuration used as the reference.
To check the system, run the **kdiag run system** command:

```
Shell> kdiag run system
PBIT "system" (fast,simple)
PCI...  HWCONF... SATA...  SMBUS...  BIOS...  ETH...  USB...
PASSED
```

By default, the test returns only the errors, and the error detail level is NORMAL.
Hereunder is an example with error outputs:
Ethernet link of the controller on the carrier board recorded as "ETH Dev 1 (GBE1 on Carrier) " has been disconnected from and an USB drive labelled "USB DISK 3.0 MAP" has been plugged to the system.

```
Shell> kdiag run system
PBIT "system" (fast,simple)
PCI...  HWCONF... SATA...  SMBUS...  BIOS...  ETH...  USB...
ERR)ETH Port 1             (R)Link UP speed 1000 Mb/s(D)Not Connected
ERR)USB Drive's Count      (R)2 Drives               (D)3 Drives
ERR)USB MassStorage(0)     (R)No Device              (D) USB DISK 3.0 PMAP
FAILED
```

(R) stand for Recorded
(D) stand for Detected

Editing the System Configuration

Run the **kdiag edit system** command to access a menu and edit the system configuration.
This menu lets you to ignore totally or partially specific items. For example it is possible to ignore checking a specific SATA port or a specific USB device type or a USB port or to totally ignore checking the PCI Bus.
It also lets you to configure a specific level of verbosity when the system test is executed.
The menu is displayed only if a system configuration has been recorded with the **kdiag learn system** command.
The menu is organized in sub-level menu.
Help is available from any sub-menu by typing < **?>** .
The < **Carriage Return>** key takes you back from the current menu.
Two main features can be distinguished:

- the " **ignore** " menu to bypass specific item verification (PCI,HWCONF,SATA,ETH,USB,BIOS entries)
- the " **print debug setting** s" menus to modify the debug level and manage the scroll limit for it (" **s** " entry).

Editing Items

By default, all the items of the system test are checked. The system edit menu allows you to choose which items will be ignored during the next system test executions.
It is possible to ignore all or part of specific elements of the system.
The following snapshot shows an edit system session ignoring the specific errors described in the previous example of §5.2. Errors are marked with the flag **FAILED ONCE** to remember that an error has occurred and the flag **LAST FAILED** which indicates that an error has occurred during the last system test.
The letter ' **p** ' prints the current system configuration for the specific class of devices.

```
Shell> kdiag edit system

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

--SystemEdit>>usb

--SystemEdit-USB>>p

UsbCounts
USB Drive's Count      2 Drives               [FAILED ONCE] [LAST FAILED]
USB Kbd's Count        1 Keyboard
USB Mouse's Count      1 Mouse
USB Hub's Count        1 Hub
USB Point's Count      0 Point
USB Ccid's Count       0 SmartCard Reader
MassNames
USB MassStorage(0)     UFD 3.0 Silicon-Pow    [FAILED ONCE]
USB MassStorage(1)     KingstonDataTravele    [FAILED ONCE]
USB MassStorage(2)     No Device              [FAILED ONCE] [LAST FAILED]
USB MassStorage(3)     No Device
UsbDevs
USB Port 0             Connected
USB Port 1             Not Connected
USB Port 2             Not Connected
```

```
USB Port 3             Not Connected
USB Port 4             Not Connected
USB Port 5             Not Connected
USB Port 6             Connected               [FAILED ONCE]
USB Port 7             Not Connected
--SystemEdit-USB>>0

--SystemEdit-USB-UsbCounts>>?

p : print Recorded values
0,..,5 : edit UsbCounts element
ignoreall : ignore all UsbCounts infos
default : restore default UsbCounts config (remove flags)
? : help
q : go back to USB menu

--SystemEdit-USB-UsbCounts>>p

2 Drives                [FAILED ONCE] [LAST FAILED]
1 Keyboard
1 Mouse
1 Hub
0 Point
0 SmartCard Reader
--SystemEdit-USB-UsbCounts>>ignoreall

UsbCounts entirely ignored
*-SystemEdit-USB-UsbCounts>>

--SystemEdit-USB>>1

--SystemEdit-USB-MassNames>>p

UFD 3.0 Silicon-Pow     [FAILED ONCE]
KingstonDataTravele     [FAILED ONCE]
No Device               [FAILED ONCE] [LAST FAILED]
No Device
--SystemEdit-USB-MassNames>>ignoreall

MassNames entirely ignored
*-SystemEdit-USB-MassNames>>p

UFD 3.0 Silicon-Pow    (i) [FAILED ONCE]
KingstonDataTravele    (i) [FAILED ONCE]
No Device              (i) [FAILED ONCE] [LAST FAILED]
No Device              (i)
*-SystemEdit-USB-MassNames>>

*-SystemEdit-USB>>

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB (*) SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

*-SystemEdit>>eth

*-SystemEdit-ETH>>p
```

```
*-SystemEdit-ETH>>p

ETH Port 0              Link UP speed 1000 Mb/s
ETH Port 1              Link UP speed 1000 Mb/s [FAILED ONCE] [LAST FAILED]
*-SystemEdit-ETH>>1

*-SystemEdit-ETH-Dev 1>>?

p : print Recorded value
i : set ignore flag
r : remove flag
? : help
q : go back to ETH menu

*-SystemEdit-ETH-Dev 1>>i

Ignore flag has been set
*-SystemEdit-ETH-Dev 1>>

*-SystemEdit-ETH>>

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH (*) USB (*) SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

*-SystemEdit>>q
Quit Edit menu

Modifications have been made.
Do you want to save your configuration before leaving? (y/n)
*-SystemEdit-Save>>y
Shell>
```

To ignore all item, use ' **ignoreall** '.
In the item submenus, ignored elements are marked with the (i) label.
At the high level edit menu, an item that is totally ignored is marked with the (i) label, and an item with ignored elements is marked with the (*) label.
To remove the IGNORE flags for specific elements, enter in the element submenu and type ' **r** ' or to remove all the flags once for an item, enter the item submenu and type ' **default** '.
On USB menu, use the following table to make the correspondence between the USB Port # in PBIT system test and the USB Ports on the **Test SIB** carrier board :

USB Port 0	Port #0 **USB2.0**
USB Port 1	Port #1 **USB2.0**
USB Port 2	Port #2 **USB2.0**
USB Port 3	Port #3 **USB2.0**
USB Port 4	Port #0 **USB3.0**
USB Port 5	Port #1 **USB3.0**
USB Port 6	Port #2 **USB3.0**
USB Port 7	Port #3 **USB3.0**

Print Settings

By default, the system test prints ERRORS ONLY and the detail level is NORMAL.
It is possible to modify these parameters by entering the edit system menu and select “s”:

```
Shell> kdiag edit system

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
```

```
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

--SystemEdit>>s

p : Set Print Level
Current Print Lvl is: ERRORS ONLY
d : Set Detail Level
Current Details Lvl is: DETAILED
c : Clear System Stats
--SystemEdit-generalSettings>>?

3 print levels : Errors only, Infos & Errors only, Debug mode (print all)
In debug mode, set the scroll limit (0 no limit, max = 30 lines)
3 Detail levels (expert only): Synthetic, Normal, Detailed
Synthetic mode does not display errors but gives you the number of errors fo
und by type and the total.
Normal mode displays system results depending on the display level.
Detailed mode gives more details for pci devices
c : clear system stats
q : return to main menu

p : Set Print Level
Current Print Lvl is: ERRORS ONLY
d : Set Detail Level
Current Details Lvl is: DETAILED
c : Clear System Stats
--SystemEdit-generalSettings>>d

--SystemEdit-generalSettings-detailsLvl>>?

p : print Current Details Lvl
s : synthethic results
n : normal infos
d : detailed
q : return to general settings menu
--SystemEdit-generalSettings-detailsLvl>>s

Details Lvl SYNTHETHIC set
*-SystemEdit-generalSettings-detailsLvl>>

p : Set Print Level
Current Print Lvl is: ERRORS ONLY
d : Set Detail Level
Current Details Lvl is: SYNTHETHIC
c : Clear System Stats
*-SystemEdit-generalSettings>>

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

*-SystemEdit>>q
Quit Edit menu

Modifications have been made.
Do you want to save your configuration before leaving? (y/n)
*-SystemEdit-Save>>y
```

```
*-SystemEdit-Save>>y
Shell>
```

Three print modes are available: ERRORS ONLY mode (default), ERRORS & INFOS ONLY mode and DEBUG LVL mode.
Each print mode is easily identifiable thanks to 3 labels:

- ERR) for ERRORS ONLY
- INF) for ERRORS & INFOS ONLY
- DBG) for DEBUG LVL

For example:

```
Shell> kdiag edit system

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

--SystemEdit>>s

p : Set Print Level
Current Print Lvl is: ERRORS ONLY
d : Set Detail Level
Current Details Lvl is: SYNTHETHIC
c : Clear System Stats
--SystemEdit-generalSettings>>p

--SystemEdit-generalSettings-printLvl>>?

p : print current print Level
e : ERRORS ONLY mode
i : ERRORS & INFOS ONLY mode
d : DEBUG LVL mode (print all)
q : return to general settings menu
--SystemEdit-generalSettings-printLvl>>p

Current Print Lvl is: ERRORS ONLY
--SystemEdit-generalSettings-printLvl>>d

Print Lvl DEBUG PRINT ALL set, scroll limit is 24 by default
Set Scroll limit (0 = no limit , max = 30)
*-SystemEdit-generalSettings-printLvl-Scroll>>

*-SystemEdit-generalSettings-printLvl>>

p : Set Print Level
Current Print Lvl is: DEBUG PRINT ALL
d : Set Detail Level
Current Details Lvl is: SYNTHETHIC
c : Clear System Stats
*-SystemEdit-generalSettings>>d

*-SystemEdit-generalSettings-detailsLvl>>?

p : print Current Details Lvl
s : synthethic results
n : normal infos
d : detailed
q : return to general settings menu
*-SystemEdit-generalSettings-detailsLvl>>n
```

```
Details Lvl NORMAL set
*-SystemEdit-generalSettings-detailsLvl>>

p : Set Print Level
Current Print Lvl is: DEBUG PRINT ALL
d : Set Detail Level
Current Details Lvl is: NORMAL
c : Clear System Stats
*-SystemEdit-generalSettings>>

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

*-SystemEdit>>

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

*-SystemEdit>>

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

*-SystemEdit>>q
Quit Edit menu

Modifications have been made.
Do you want to save your configuration before leaving? (y/n)
*-SystemEdit-Save>>y
Shell> kdiag run system
PBIT "system" (fast,simple)
Size of one STES = 0x2
Total size = 0x1
Size of global control = 0xC

PCI...  HWCONF... SATA...  SMBUS...  BIOS...  ETH...  USB...
CRC Recorded 0x6 different from CRC Detected 0x6
==>System Configuration area changed

DBG)PCI Bus:Dev:Func       (R)00:00:00              (D)00:00:00
DBG)PCI 00:00:00 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:00:00 DeviceID  (R)0x1918                (D)0x1918
DBG)PCI 00:00:00 ClassCode (R)060000                (D)060000
DBG)PCI Bus:Dev:Func       (R)00:02:00              (D)00:02:00
DBG)PCI 00:02:00 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:02:00 DeviceID  (R)0x191D                (D)0x191D
DBG)PCI 00:02:00 ClassCode (R)030000                (D)030000
DBG)PCI Bus:Dev:Func       (R)00:08:00              (D)00:08:00
DBG)PCI 00:08:00 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:08:00 DeviceID  (R)0x1911                (D)0x1911
DBG)PCI 00:08:00 ClassCode (R)088000                (D)088000
DBG)PCI Bus:Dev:Func       (R)00:14:00              (D)00:14:00
```

```
DBG)PCI Bus:Dev:Func       (R)00:14:00              (D)00:14:00
DBG)PCI 00:14:00 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:14:00 DeviceID  (R)0xA12F                (D)0xA12F
DBG)PCI 00:14:00 ClassCode (R)0C0330                (D)0C0330
DBG)PCI Bus:Dev:Func       (R)00:14:02              (D)00:14:02
DBG)PCI 00:14:02 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:14:02 DeviceID  (R)0xA131                (D)0xA131
DBG)PCI 00:14:02 ClassCode (R)118000                (D)118000
DBG)PCI Bus:Dev:Func       (R)00:16:00              (D)00:16:00
DBG)PCI 00:16:00 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:16:00 DeviceID  (R)0xA13A                (D)0xA13A
DBG)PCI 00:16:00 ClassCode (R)078000                (D)078000
DBG)PCI Bus:Dev:Func       (R)00:17:00              (D)00:17:00
DBG)PCI 00:17:00 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:17:00 DeviceID  (R)0xA102                (D)0xA102
DBG)PCI 00:17:00 ClassCode (R)010601                (D)010601
DBG)PCI Bus:Dev:Func       (R)00:1D:00              (D)00:1D:00
DBG)PCI 00:1D:00 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:1D:00 DeviceID  (R)0xA118                (D)0xA118
DBG)PCI 00:1D:00 ClassCode (R)060400                (D)060400
DBG)PCI 00:1D:00 LinkStat  (R)Link is UP            (D)Link is UP
DBG)PCI 00:1D:00 Wdth/Spd  (R)x1 / 2.5GT/s          (D)x1 / 2.5GT/s
DBG)PCI Bus:Dev:Func       (R)00:1D:01              (D)00:1D:01
DBG)PCI 00:1D:01 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:1D:01 DeviceID  (R)0xA119                (D)0xA119
DBG)PCI 00:1D:01 ClassCode (R)060400                (D)060400
DBG)PCI 00:1D:01 LinkStat  (R)Link is UP            (D)Link is UP
DBG)PCI 00:1D:01 Wdth/Spd  (R)x1 / 2.5GT/s          (D)x1 / 2.5GT/s
DBG)PCI Bus:Dev:Func       (R)00:1E:00              (D)00:1E:00
DBG)PCI 00:1E:00 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:1E:00 DeviceID  (R)0xA127                (D)0xA127
DBG)PCI 00:1E:00 ClassCode (R)118000                (D)118000
DBG)PCI Bus:Dev:Func       (R)00:1F:00              (D)00:1F:00
DBG)PCI 00:1F:00 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:1F:00 DeviceID  (R)0xA150                (D)0xA150
DBG)PCI 00:1F:00 ClassCode (R)060100                (D)060100
DBG)PCI Bus:Dev:Func       (R)00:1F:02              (D)00:1F:02
DBG)PCI 00:1F:02 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:1F:02 DeviceID  (R)0xA121                (D)0xA121
DBG)PCI 00:1F:02 ClassCode (R)058000                (D)058000
DBG)PCI Bus:Dev:Func       (R)00:1F:03              (D)00:1F:03
DBG)PCI 00:1F:03 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:1F:03 DeviceID  (R)0xA170                (D)0xA170
DBG)PCI 00:1F:03 ClassCode (R)040300                (D)040300
DBG)PCI Bus:Dev:Func       (R)00:1F:04              (D)00:1F:04
DBG)PCI 00:1F:04 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:1F:04 DeviceID  (R)0xA123                (D)0xA123
DBG)PCI 00:1F:04 ClassCode (R)0C0500                (D)0C0500
DBG)PCI Bus:Dev:Func       (R)00:1F:06              (D)00:1F:06
DBG)PCI 00:1F:06 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 00:1F:06 DeviceID  (R)0x15B7                (D)0x15B7
DBG)PCI 00:1F:06 ClassCode (R)020000                (D)020000
DBG)PCI Bus:Dev:Func       (R)01:00:00              (D)01:00:00
DBG)PCI 01:00:00 VendorID  (R)0x8086                (D)0x8086
DBG)PCI 01:00:00 DeviceID  (R)0x1533                (D)0x1533
DBG)PCI 01:00:00 ClassCode (R)020000                (D)020000
DBG)PCI Bus:Dev:Func       (R)02:00:00              (D)02:00:00
DBG)PCI 02:00:00 VendorID  (R)0x13A8                (D)0x13A8
```

```
DBG)PCI 02:00:00 DeviceID  (R)0x0354                 (D)0x0354
DBG)PCI 02:00:00 ClassCode (R)070002                 (D)070002
DBG)HWCONF FPGA Version    (R)P108.0069 Release      (D)P108.0069 Release
DBG)HWCONF Dram Size       (R)32GB                   (D)32GB
DBG)SATA Port 0            (R)Not Connected (0.0GB)  (D)Not Connected (0.0GB
)
DBG)SATA Port 1            (R)LDLC           (31.6GB)(D)LDLC           (31
.6GB)
DBG)SATA Port 2            (R)Not Connected (0.0GB)  (D)Not Connected (0.0GB
)
DBG)SATA Port 3            (R)ST9160314AS    (160.0GB)(D)ST9160314AS    (1
60.0GB)
DBG)SMBUS0 Device 0        (R)0x38                   (D)0x38
DBG)SMBUS0 Device 1        (R)0x42                   (D)0x42
DBG)SMBUS0 Device 2        (R)0x70                   (D)0x70
DBG)SMBUS0 Device 3        (R)0xA0                   (D)0xA0
DBG)SMBUS0 Device 4        (R)0xAE                   (D)0xAE
DBG)SMBUS0 Device 5        (R)0xC0                   (D)0xC0
DBG)SMBUS0 Device 6        (R)0xC2                   (D)0xC2
DBG)SMBUS0 Device 7        (R)0xC4                   (D)0xC4
DBG)SMBUS0 Device 8        (R)0xC6                   (D)0xC6
DBG)SMBUS0 Device 9        (R)0xE0                   (D)0xE0
DBG)SMBUS0 Device 10       (R)0xE2                   (D)0xE2
DBG)SMBUS0 Device 11       (R)0xE4                   (D)0xE4
DBG)BIOS Checksum          (R)143647                 (D)143647
DBG)BIOS Version           (R)BBD7R110               (D)BBD7R110
DBG)ETH Port 0             (R)Link UP speed 1000 Mb/s(D)Link UP speed 1000 M
b/s
ERR)ETH Port 1             (R)Link UP speed 1000 Mb/s(D)Not Connected
ERR)USB Drive's Count      (R)2 Drives               (D)3 Drives
DBG)USB Kbd's Count        (R)1 Keyboard             (D)1 Keyboard
DBG)USB Mouse's Count      (R)1 Mouse                (D)1 Mouse
DBG)USB Hub's Count        (R)1 Hub                  (D)1 Hub
DBG)USB Point's Count      (R)0 Point                (D)0 Point
DBG)USB Ccid's Count       (R)0 SmartCard Reader     (D)0 SmartCard Reader
DBG)USB MassStorage(0)     (R)UFD 3.0 Silicon-Pow    (D)UFD 3.0 Silicon-Pow
DBG)USB MassStorage(1)     (R)KingstonDataTravele    (D)KingstonDataTravele
ERR)USB MassStorage(2)     (R)No Device              (D) USB DISK 3.0 PMAP
DBG)USB MassStorage(3)     (R)No Device              (D)No Device
DBG)USB Port 0             (R)Connected              (D)Connected
DBG)USB Port 1             (R)Not Connected          (D)Not Connected
DBG)USB Port 2             (R)Not Connected          (D)Not Connected
DBG)USB Port 3             (R)Not Connected          (D)Not Connected
DBG)USB Port 4             (R)Not Connected          (D)Not Connected
DBG)USB Port 5             (R)Not Connected          (D)Not Connected
DBG)USB Port 6             (R)Connected              (D)Connected
DBG)USB Port 7             (R)Not Connected          (D)Not Connected
FAILED

Shell>
```

If the DEBUG LVL mode is selected, a scroll limit can be set in order to stop scrolling during the system test.
0 means no scroll limit, the maximum for the scroll limit is 30 lines.
The watchdog will be stopped during during the system test execution if the scroll limit is not 0.
At the next system test execution, use the Space bar (scroll to the limit) or the Enter key (scroll line by line) to make test results scrolling.
Three detail levels are available (for experts because not very useful for user): SYNTHETIC results mode, NORMAL info mode (default) and DETAILED mode.
The SYNTHETIC mode does not display errors and information results but returns the number of errors found by item and the total of failed items.

```
Shell> kdiag run system
PBIT "system" (fast,simple)
PCI...  HWCONF... SATA...  SMBUS...  BIOS...  ETH...  USB...
ERR)ETH Results : 1 ERRORS
ERR)USB Results : 2 ERRORS
ERR)TOTAL : 3 ERRORS
FAILED
Shell>
```

The NORMAL mode displays the system test results depending on the print level.

```
Shell> kdiag run system
PBIT "system" (fast,simple)
PCI...  HWCONF... SATA...  SMBUS...  BIOS...  ETH...  USB...
ERR)ETH Port 1             (R)Link UP speed 1000 Mb/s(D)Not Connected
ERR)USB Drive's Count      (R)2 Drives               (D)3 Drives
ERR)USB MassStorage(2)     (R)No Device              (D) USB DISK 3.0 PMAP
FAILED

Shell>
```

The DETAILED mode allows to print more details, mainly for the PCI devices. The classcode value is printed but also the ProgInterface number and the BaseClass and SubClass strings.
In the following example, enable USB OTG in BIOS Setup and run "**kdiag run system** ":

```
Shell> kdiag run system
PBIT "system" (fast,simple)
PCI...  HWCONF... SATA...  SMBUS...  BIOS...  ETH...  USB...
ERR)PCI Bus:Dev:Func       (R)00:14:02               (D)00:14:01
ERR)PCI 00:14:02 DeviceID  (R)0xA131                 (D)0xA130
ERR)PCI 00:14:02 ClassCode (R)118000                 (D)118000
ProgInterface          (R)0                      (D)0
BaseClass              (R)Data Acquisition & Sign(D)Data Acquisition & Sign
SubClass               (R)Other DAQ & SP controll(D)Other DAQ & SP controll
ERR)PCI Bus:Dev:Func       (R)00:16:00               (D)00:14:02
ERR)PCI 00:16:00 DeviceID  (R)0xA13A                 (D)0xA131
ERR)PCI 00:16:00 ClassCode (R)078000                 (D)078000
ProgInterface          (R)0                      (D)0
BaseClass              (R)Simple Communications C(D)Simple Communications C
SubClass               (R)Other communication dev(D)Other communication dev
ERR)PCI Bus:Dev:Func       (R)00:17:00               (D)00:16:00
ERR)PCI 00:17:00 DeviceID  (R)0xA102                 (D)0xA13A
ERR)PCI 00:17:00 ClassCode (R)010601                 (D)010601
ProgInterface          (R)1                      (D)1
BaseClass              (R)Mass Storage Controller(D)Mass Storage Controller
SubClass               (R)Serial ATA controller  (D)Serial ATA controller
ERR)PCI Bus:Dev:Func       (R)00:1D:00               (D)00:17:00
ERR)PCI 00:1D:00 DeviceID  (R)0xA118                 (D)0xA102
ERR)PCI 00:1D:00 ClassCode (R)060400                 (D)060400
ProgInterface          (R)0                      (D)0
BaseClass              (R)Bridge Device          (D)Bridge Device
SubClass               (R)PCI/PCI bridge         (D)PCI/PCI bridge
ERR)PCI 00:1D:00 LinkStat  (R)Link is UP             (D)LinkStat not availab
le
ERR)PCI 00:1D:00 Wdth/Spd  (R)x1 / 2.5GT/s           (D)Bandwidth not availa
ble
ERR)PCI Bus:Dev:Func       (R)00:1D:01               (D)00:1D:00
ERR)PCI 00:1D:01 DeviceID  (R)0xA119                 (D)0xA118
ERR)PCI Bus:Dev:Func       (R)00:1E:00               (D)00:1D:01
ERR)PCI 00:1E:00 DeviceID  (R)0xA127                 (D)0xA119
```

```
ERR)PCI 00:1E:00 DeviceID  (R)0xA1B7                    (D)0xA129
ERR)PCI 00:1E:00 ClassCode (R)118000                    (D)118000
ProgInterface          (R)0                        (D)0
BaseClass              (R)Data Acquisition & Sign  (D)Data Acquisition & Sign
SubClass               (R)Other DAQ & SP controll  (D)Other DAQ & SP controll
ERR)PCI 00:1E:00 LinkStat  (R)LinkStat not available (D)Link is UP
ERR)PCI 00:1E:00 Wdth/Spd  (R)Bandwidth not available(D)x1 / 2.5GT/s
ERR)PCI Bus:Dev:Func       (R)00:1F:00              (D)00:1E:00
ERR)PCI 00:1F:00 DeviceID  (R)0xA150                (D)0xA127
ERR)PCI 00:1F:00 ClassCode (R)060100                (D)060100
ProgInterface          (R)0                        (D)0
BaseClass              (R)Bridge Device            (D)Bridge Device
SubClass               (R)PCI/ISA bridge           (D)PCI/ISA bridge
ERR)PCI Bus:Dev:Func       (R)00:1F:02              (D)00:1F:00
ERR)PCI 00:1F:02 DeviceID  (R)0xA121                (D)0xA150
ERR)PCI 00:1F:02 ClassCode (R)058000                (D)058000
ProgInterface          (R)0                        (D)0
BaseClass              (R)Memory Controller        (D)Memory Controller
SubClass               (R)Other memory controller(D)Other memory controller
ERR)PCI Bus:Dev:Func       (R)00:1F:03              (D)00:1F:02
ERR)PCI 00:1F:03 DeviceID  (R)0xA170                (D)0xA121
ERR)PCI 00:1F:03 ClassCode (R)040300                (D)040300
ProgInterface          (R)0                        (D)0
BaseClass              (R)Multimedia Device        (D)Multimedia Device
SubClass               (R)Mixed mode device        (D)Mixed mode device
ERR)PCI Bus:Dev:Func       (R)00:1F:04              (D)00:1F:03
ERR)PCI 00:1F:04 DeviceID  (R)0xA123                (D)0xA170
ERR)PCI 00:1F:04 ClassCode (R)0C0500                (D)0C0500
ProgInterface          (R)0                        (D)0
BaseClass              (R)Serial Bus Controllers   (D)Serial Bus Controllers
SubClass               (R)System Management Bus    (D)System Management Bus
ERR)PCI Bus:Dev:Func       (R)00:1F:06              (D)00:1F:04
ERR)PCI 00:1F:06 DeviceID  (R)0x15B7                (D)0xA123
ERR)PCI 00:1F:06 ClassCode (R)020000                (D)020000
ProgInterface          (R)0                        (D)0
BaseClass              (R)Network Controller       (D)Network Controller
SubClass               (R)Ethernet controller      (D)Ethernet controller
ERR)PCI Bus:Dev:Func       (R)01:00:00              (D)00:1F:06
ERR)PCI 01:00:00 DeviceID  (R)0x1533                (D)0x15B7
ERR)PCI Bus:Dev:Func       (R)02:00:00              (D)01:00:00
ERR)PCI 02:00:00 VendorID  (R)0x13A8                (D)0x8086
ERR)PCI 02:00:00 DeviceID  (R)0x0354                (D)0x1533
ERR)PCI 02:00:00 ClassCode (R)070002                (D)070002
ProgInterface          (R)2                        (D)2
BaseClass              (R)Simple Communications C(D)Simple Communications C
SubClass               (R)Serial controller        (D)Serial controller
ERR)PCI Bus:Dev:Func       (R)No Device             (D)02:00:00
ERR)PCI 00:00:00 VendorID  (R)No Device             (D)0x13A8
ERR)PCI 00:00:00 DeviceID  (R)No Device             (D)0x0354
ERR)PCI 00:00:00 ClassCode (R)No Device             (D)No Device
ProgInterface          (R)No Device                (D)No Device
BaseClass              (R)No Device                (D)No Device
SubClass               (R)No Device                (D)No Device
ERR)BIOS Checksum          (R)143647                (D)86557
ERR)ETH Port 1             (R)Link UP speed 1000 Mb/s(D)Not Connected
ERR)USB Drive's Count      (R)2 Drives              (D)3 Drives
ERR)USB MassStorage(2)     (R)No Device             (D) USB DISK 3.0 PMAP
FAILED
```

```
Shell>
```

The PCI devices list is shifted from the previous PCI list (done during the "kdiag learn system" command) due to the added PCI component for USB OTG controller.
The scrolling for displaying PCI list is not automatic (see scroll limit notes above), user have to press **<space>** key to move next screen.

Clearing System Status

The "Clear System Stats" selection provided in the "settings" menu is used to clear the system flags FAILED ONCE and LAST FAILED. For example:

```
Shell> kdiag run system
PBIT "system" (fast,simple)
PCI...  HWCONF... SATA...  SMBUS...  BIOS...  ETH...  USB...
ERR)PCI Bus:Dev:Func       (R)00:14:02              (D)00:14:01
ERR)PCI 00:14:02 DeviceID  (R)0xA131                (D)0xA130
ERR)PCI 00:14:02 ClassCode (R)118000                (D)118000
ProgInterface          (R)0                     (D)0
BaseClass              (R)Data Acquisition & Sign(D)Data Acquisition & Sign
SubClass               (R)Other DAQ & SP controll(D)Other DAQ & SP controll
ERR)PCI Bus:Dev:Func       (R)00:16:00              (D)00:14:02
ERR)PCI 00:16:00 DeviceID  (R)0xA13A                (D)0xA131
ERR)PCI 00:16:00 ClassCode (R)078000                (D)078000
ProgInterface          (R)0                     (D)0
BaseClass              (R)Simple Communications C(D)Simple Communications C
SubClass               (R)Other communication dev(D)Other communication dev
ERR)PCI Bus:Dev:Func       (R)00:17:00              (D)00:16:00
ERR)PCI 00:17:00 DeviceID  (R)0xA102                (D)0xA13A
ERR)PCI 00:17:00 ClassCode (R)010601                (D)010601
ProgInterface          (R)1                     (D)1
BaseClass              (R)Mass Storage Controller(D)Mass Storage Controller
SubClass               (R)Serial ATA controller  (D)Serial ATA controller
ERR)PCI Bus:Dev:Func       (R)00:1D:00              (D)00:17:00
ERR)PCI 00:1D:00 DeviceID  (R)0xA118                (D)0xA102
ERR)PCI 00:1D:00 ClassCode (R)060400                (D)060400
ProgInterface          (R)0                     (D)0
BaseClass              (R)Bridge Device          (D)Bridge Device
SubClass               (R)PCI/PCI bridge         (D)PCI/PCI bridge
ERR)PCI 00:1D:00 LinkStat  (R)Link is UP             (D)LinkStat not availab
le
ERR)PCI 00:1D:00 Wdth/Spd  (R)x1 / 2.5GT/s           (D)Bandwidth not availa
ble
ERR)PCI Bus:Dev:Func       (R)00:1D:01              (D)00:1D:00
ERR)PCI 00:1D:01 DeviceID  (R)0xA119                (D)0xA118
ERR)PCI Bus:Dev:Func       (R)00:1E:00              (D)00:1D:01
ERR)PCI 00:1E:00 DeviceID  (R)0xA127                (D)0xA119
ERR)PCI 00:1E:00 ClassCode (R)118000                (D)118000
ProgInterface          (R)0                     (D)0
BaseClass              (R)Data Acquisition & Sign(D)Data Acquisition & Sign
SubClass               (R)Other DAQ & SP controll(D)Other DAQ & SP controll
ERR)PCI 00:1E:00 LinkStat  (R)LinkStat not available (D)Link is UP
ERR)PCI 00:1E:00 Wdth/Spd  (R)Bandwidth not available(D)x1 / 2.5GT/s
ERR)PCI Bus:Dev:Func       (R)00:1F:00              (D)00:1E:00
ERR)PCI 00:1F:00 DeviceID  (R)0xA150                (D)0xA127
ERR)PCI 00:1F:00 ClassCode (R)060100                (D)060100
ProgInterface          (R)0                     (D)0
BaseClass              (R)Bridge Device          (D)Bridge Device
SubClass               (R)PCI/ISA bridge         (D)PCI/ISA bridge
ERR)PCI Bus:Dev:Func       (R)00:1F:02              (D)00:1F:00
ERR)PCI 00:1F:02 DeviceID  (R)0xA121                (D)0xA150
```

```
ERR)PCI 00:1F:02 ClassCode (R)058000                (D)058000
ProgInterface          (R)0                     (D)0
BaseClass              (R)Memory Controller      (D)Memory Controller
SubClass               (R)Other memory controller(D)Other memory controller
ERR)PCI Bus:Dev:Func       (R)00:1F:03              (D)00:1F:02
ERR)PCI 00:1F:03 DeviceID  (R)0xA170                (D)0xA121
ERR)PCI 00:1F:03 ClassCode (R)040300                (D)040300
ProgInterface          (R)0                     (D)0
BaseClass              (R)Multimedia Device      (D)Multimedia Device
SubClass               (R)Mixed mode device      (D)Mixed mode device
ERR)PCI Bus:Dev:Func       (R)00:1F:04              (D)00:1F:03
ERR)PCI 00:1F:04 DeviceID  (R)0xA123                (D)0xA170
ERR)PCI 00:1F:04 ClassCode (R)0C0500                (D)0C0500
ProgInterface          (R)0                     (D)0
BaseClass              (R)Serial Bus Controllers (D)Serial Bus Controllers
SubClass               (R)System Management Bus  (D)System Management Bus
ERR)PCI Bus:Dev:Func       (R)00:1F:06              (D)00:1F:04
ERR)PCI 00:1F:06 DeviceID  (R)0x15B7                (D)0xA123
ERR)PCI 00:1F:06 ClassCode (R)020000                (D)020000
ProgInterface          (R)0                     (D)0
BaseClass              (R)Network Controller     (D)Network Controller
SubClass               (R)Ethernet controller    (D)Ethernet controller
ERR)PCI Bus:Dev:Func       (R)01:00:00              (D)00:1F:06
ERR)PCI 01:00:00 DeviceID  (R)0x1533                (D)0x15B7
ERR)PCI Bus:Dev:Func       (R)02:00:00              (D)01:00:00
ERR)PCI 02:00:00 VendorID  (R)0x13A8                (D)0x8086
ERR)PCI 02:00:00 DeviceID  (R)0x0354                (D)0x1533
ERR)PCI 02:00:00 ClassCode (R)070002                (D)070002
ProgInterface          (R)2                     (D)2
BaseClass              (R)Simple Communications C(D)Simple Communications C
SubClass               (R)Serial controller      (D)Serial controller
ERR)PCI Bus:Dev:Func       (R)No Device             (D)02:00:00
ERR)PCI 00:00:00 VendorID  (R)No Device             (D)0x13A8
ERR)PCI 00:00:00 DeviceID  (R)No Device             (D)0x0354
ERR)PCI 00:00:00 ClassCode (R)No Device             (D)No Device
ProgInterface          (R)No Device              (D)No Device
BaseClass              (R)No Device              (D)No Device
SubClass               (R)No Device              (D)No Device
ERR)BIOS Checksum          (R)143647                (D)86557
ERR)ETH Port 1             (R)Link UP speed 1000 Mb/s(D)Not Connected
ERR)USB Drive's Count      (R)2 Drives              (D)3 Drives
ERR)USB MassStorage(2)     (R)No Device             (D) USB DISK 3.0 PMAP
FAILED

Shell> kdiag edit system

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

--SystemEdit>>eth

--SystemEdit-ETH>>p

ETH Port 0             Link UP speed 1000 Mb/s
ETH Port 1             Link UP speed 1000 Mb/s [FAILED ONCE] [LAST FAILED]
--SystemEdit-ETH>>
```

```
-

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

--SystemEdit>>usb

--SystemEdit-USB>>p

UsbCounts
USB Drive's Count      2 Drives                [FAILED ONCE] [LAST FAILED]
USB Kbd's Count        1 Keyboard
USB Mouse's Count      1 Mouse
USB Hub's Count        1 Hub
USB Point's Count      0 Point
USB Ccid's Count       0 SmartCard Reader
MassNames
USB MassStorage(0)     UFD 3.0 Silicon-Pow
USB MassStorage(1)     KingstonDataTravele
USB MassStorage(2)     No Device               [FAILED ONCE] [LAST FAILED]
USB MassStorage(3)     No Device
UsbDevs
USB Port 0             Connected
USB Port 1             Not Connected
USB Port 2             Not Connected
USB Port 3             Not Connected
USB Port 4             Not Connected
USB Port 5             Not Connected
USB Port 6             Connected
USB Port 7             Not Connected
--SystemEdit-USB>>

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

--SystemEdit>>pci

--SystemEdit-PCI>>p

PCI 00:00:00 Bridge Device/Host/PCI bridge
PCI 00:02:00 Display Controller/VGA/8514 controller
PCI 00:08:00 Base System Peripherals/Other system peripheral
PCI 00:14:00 Serial Bus Controllers/USB
PCI 00:14:02 Data Acquisition & Signal Processing Co/Other DAQ & SP controll
ers [FAILED ONCE] [LAST FAILED]
PCI 00:16:00 Simple Communications Controllers/Other communication device [F
AILED ONCE] [LAST FAILED]
PCI 00:17:00 Mass Storage Controller/Serial ATA controller [FAILED ONCE] [LA
ST FAILED]
PCI 00:1D:00 Bridge Device/PCI/PCI bridge [FAILED ONCE] [LAST FAILED]
PCI 00:1D:01 Bridge Device/PCI/PCI bridge [FAILED ONCE] [LAST FAILED]
PCI 00:1E:00 Data Acquisition & Signal Processing Co/Other DAQ & SP controll
ers [FAILED ONCE] [LAST FAILED]
10. PCI 00:1F:00 Bridge Device/PCI/ISA bridge [FAILED ONCE] [LAST FAILED]
```

```
11. PCI 00:1F:02 Memory Controller/Other memory controller [FAILED ONCE] [LA
ST FAILED]
12. PCI 00:1F:03 Multimedia Device/Mixed mode device [FAILED ONCE] [LAST FAI
LED]
13. PCI 00:1F:04 Serial Bus Controllers/System Management Bus [FAILED ONCE]
[LAST FAILED]
14. PCI 00:1F:06 Network Controller/Ethernet controller [FAILED ONCE] [LAST
FAILED]
15. PCI 01:00:00 Network Controller/Ethernet controller [FAILED ONCE] [LAST
FAILED]
16. PCI 02:00:00 Simple Communications Controllers/Serial controller [FAILED
 ONCE] [LAST FAILED]
17. PCI No Device No Device/No Device [FAILED ONCE] [LAST FAILED]
18. PCI No Device No Device/No Device
19. PCI No Device No Device/No Device
20. PCI No Device No Device/No Device
21. PCI No Device No Device/No Device
22. PCI No Device No Device/No Device
23. PCI No Device No Device/No Device
24. PCI No Device No Device/No Device
25. PCI No Device No Device/No Device
26. PCI No Device No Device/No Device
27. PCI No Device No Device/No Device
28. PCI No Device No Device/No Device
29. PCI No Device No Device/No Device
30. PCI No Device No Device/No Device
31. PCI No Device No Device/No Device
32. PCI No Device No Device/No Device
33. PCI No Device No Device/No Device
34. PCI No Device No Device/No Device
35. PCI No Device No Device/No Device
36. PCI No Device No Device/No Device
37. PCI No Device No Device/No Device
38. PCI No Device No Device/No Device
39. PCI No Device No Device/No Device
40. PCI No Device No Device/No Device
41. PCI No Device No Device/No Device
42. PCI No Device No Device/No Device
43. PCI No Device No Device/No Device
44. PCI No Device No Device/No Device
45. PCI No Device No Device/No Device
46. PCI No Device No Device/No Device
47. PCI No Device No Device/No Device
48. PCI No Device No Device/No Device
49. PCI No Device No Device/No Device
--SystemEdit-PCI>>

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

--SystemEdit>>s

p : Set Print Level
Current Print Lvl is: ERRORS ONLY
d : Set Detail Level
Current Details Lvl is: DETAILED
```

```
c : Clear System Stats
--SystemEdit-generalSettings>>c

System stats have been cleared

p : Set Print Level
Current Print Lvl is: ERRORS ONLY
d : Set Detail Level
Current Details Lvl is: DETAILED
c : Clear System Stats
--SystemEdit-generalSettings>>

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

--SystemEdit>>q
Quit Edit menu

No modifications, quit without saving..
Shell> kdiag edit system

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

--SystemEdit>>eth

--SystemEdit-ETH>>p

ETH Port 0             Link UP speed 1000 Mb/s
ETH Port 1             Link UP speed 1000 Mb/s
--SystemEdit-ETH>>

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

--SystemEdit>>usb

--SystemEdit-USB>>p

UsbCounts
USB Drive's Count      2 Drives
USB Kbd's Count        1 Keyboard
USB Mouse's Count      1 Mouse
USB Hub's Count        1 Hub
USB Point's Count      0 Point
USB Ccid's Count       0 SmartCard Reader
MassNames
USB MassStorage(0)     UFD 3.0 Silicon-Pow
USB MassStorage(1)     KingstonDataTravele
USB MassStorage(2)     No Device
USB MassStorage(3)     No Device
UsbDevs
```

```
usbdevs
USB Port 0              Connected
USB Port 1              Not Connected
USB Port 2              Not Connected
USB Port 3              Not Connected
USB Port 4              Not Connected
USB Port 5              Not Connected
USB Port 6              Connected
USB Port 7              Not Connected
--SystemEdit-USB>>

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

--SystemEdit>>pci

--SystemEdit-PCI>>p

PCI 00:00:00 Bridge Device/Host/PCI bridge
PCI 00:02:00 Display Controller/VGA/8514 controller
PCI 00:08:00 Base System Peripherals/Other system peripheral
PCI 00:14:00 Serial Bus Controllers/USB
PCI 00:14:02 Data Acquisition & Signal Processing Co/Other DAQ & SP controll
ers
PCI 00:16:00 Simple Communications Controllers/Other communication device
PCI 00:17:00 Mass Storage Controller/Serial ATA controller
PCI 00:1D:00 Bridge Device/PCI/PCI bridge
PCI 00:1D:01 Bridge Device/PCI/PCI bridge
PCI 00:1E:00 Data Acquisition & Signal Processing Co/Other DAQ & SP controll
ers
10. PCI 00:1F:00 Bridge Device/PCI/ISA bridge
11. PCI 00:1F:02 Memory Controller/Other memory controller
12. PCI 00:1F:03 Multimedia Device/Mixed mode device
13. PCI 00:1F:04 Serial Bus Controllers/System Management Bus
14. PCI 00:1F:06 Network Controller/Ethernet controller
15. PCI 01:00:00 Network Controller/Ethernet controller
16. PCI 02:00:00 Simple Communications Controllers/Serial controller
17. PCI No Device No Device/No Device
18. PCI No Device No Device/No Device
19. PCI No Device No Device/No Device
20. PCI No Device No Device/No Device
21. PCI No Device No Device/No Device
22. PCI No Device No Device/No Device
23. PCI No Device No Device/No Device
24. PCI No Device No Device/No Device
25. PCI No Device No Device/No Device
26. PCI No Device No Device/No Device
27. PCI No Device No Device/No Device
28. PCI No Device No Device/No Device
29. PCI No Device No Device/No Device
30. PCI No Device No Device/No Device
31. PCI No Device No Device/No Device
32. PCI No Device No Device/No Device
33. PCI No Device No Device/No Device
34. PCI No Device No Device/No Device
35. PCI No Device No Device/No Device
```

```
36. PCI No Device No Device/No Device
37. PCI No Device No Device/No Device
38. PCI No Device No Device/No Device
39. PCI No Device No Device/No Device
40. PCI No Device No Device/No Device
41. PCI No Device No Device/No Device
42. PCI No Device No Device/No Device
43. PCI No Device No Device/No Device
44. PCI No Device No Device/No Device
45. PCI No Device No Device/No Device
46. PCI No Device No Device/No Device
47. PCI No Device No Device/No Device
48. PCI No Device No Device/No Device
49. PCI No Device No Device/No Device
--SystemEdit-PCI>>

           <<  KONTRON SYSTEM PBIT : EDIT MODE  >>

Edit by feature, choose: PCI HWCONF SATA ETH USB SMBUS BIOS
Other available cmds: `s' for settings, `?' for help, `q' to quit edit mode

--SystemEdit>>q
Quit Edit menu

No modifications, quit without saving..
Shell>
```

Note that the edit menu must be entered again to verify the “clear” action.

Document symbols and acronyms

Symbols

The following symbols are used in Kontron documentation.

DANGER	DANGER indicates a hazardous situation which, if not avoided, will result in death or serious injury.
WARNING	WARNING indicates a hazardous situation which, if not avoided, could result in death or serious injury.
CAUTION	CAUTION indicates a hazardous situation which, if not avoided, may result in minor or moderate injury.
NOTICE	NOTICE indicates a property damage message.

	**Electric Shock!** This symbol and title warn of hazards due to electrical shocks (> 60 V) when touching products or parts of them. Failure to observe the precautions indicated and/or prescribed by the law may endanger your life/health and/or result in damage to your material. Please also refer to the "High-Voltage Safety Instructions" portion below in this section.

	**ESD Sensitive Device!** This symbol and title inform that the electronic boards and their components are sensitive to static electricity. Care must therefore be taken during all handling operations and inspections of this product in order to ensure product integrity at all times.
	**HOT Surface!** Do NOT touch! Allow to cool before servicing.

	This symbol indicates general information about the product and the documentation. This symbol also indicates detailed information about the specific product configuration.

	This symbol precedes helpful hints and tips for daily use.

Acronyms

ACPI	Advanced Configuration and Power Interface
AI	Artificial Intelligence
AIC	Add-in Card (e.g. PCI Express)
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
BIOS	Basic Input/Output System
BMC	Baseboard Management Controller
BSP	Board Support Package
CBIT	Continuous Built-In Test
CE	Community European (EU mark)
CLI	Command-Line Interface
COMe	COM-express
CPU	Central Processing Unit
CRMS	Communications Rack Mount Servers
CSA	Canadian Standards Association
DC	Direct Current
DDR4	Double Data Rate Fourth Generation
DHCP	Dynamic Host Configuration Protocol
DIMM	Dual Inline Memory Module
DRAM	Dynamic Random Access Memory
DTS	Digital Thermal Sensor
DU	Distributed Unit

ECC	Error Checking and Correcting
EEPROM	Electrically Erasable Programmable Read-Only Memory
EFI	Extensible Firmware Interface
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
ESD	Electrostatic Discharge
ETSI	European Telecommunications Standards Institute
ETSI	European Telecommunications Standards Institute
eUSB	Embedded Universal Serial Bus
FCC	Federal Communications Commission
FH/FL	Full Height/Full Length
FPGA	Field Programmable Gate Array
FRAU	Field Replaceable Unit
FRU	Field Replaceable Unit
Gb, Gbit	Gigabit
GB, Gbyte	Gigabyte – 1024 MB
GbE	Gigabit Ethernet
GND	Ground
GPI	General Purpose Input
GPIO	General Purpose Input/Output
GPO	General Purpose Output
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HDD	Hard Disk Drive
Hz	Hertz – 1 cycle/second
I/O	Input/Output
I^2C	Inter-Integrated Circuit Bus
iBMC	Integrated Baseboard Management Controller
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IMU	Inertial Measurement Unit
IOL	IPMI over LAN
IPMB	Intelligent Platform Management Bus
IPMI	Intelligent Platform Management Interface
IRQ	Interrupt Request Line
KB, Kbyte	Kilobyte – 1024 bytes
KCS	Keyboard Controller Style
KEAPI	Kontron Embedded Application Programming Interface
KVM	Keyboard, Video, Mouse
LAN	Local Area Network
LED	Light-Emitting Diode
LP	Low Profile
LPC	Low Pin Count
LVDS	Low Voltage Differential SCSI
MAT	Maximum Ambient Temperature
MB, Mbyte	Megabyte – 1024 KB
MCU	Microcontroller Unit

MCU	Microcontroller Unit
MEC	Multi-Access Edge Computing
MXM	Mobile PCI Express Module
NCSI	Network Communications Services Interface
NEBS	Network Equipment-Building System
NIC	Network Interface Card, or Network Interface Controller, or Network Interface Controller port
NMI	Non-Maskable interrupt
NOS	Network Operating System
NVMe	Non-Volatile Memory Express
OCXO	Oven-Controlled Crystal Oscillator
OS	Operating System
OTP	Over-Temperature Protection
OVP	Over-Voltage Protection
PBIT	Power On Built-In Test
PCH	Platform Controller Hub
PCI	Peripheral Component Interconnect
PCIe	Peripheral Component Interconnect Express
PECI	Platform Environment Control Interface
PIRQ	PCI Interrupt Request Line
PMbus	Power Management Bus
PMM	POST Memory Manager
PnP	Plug and Play
POC	Proof of Concept
POST	Power-On Self Test
PSU	Power Supply Unit
PTP	Precision Time Protocol
PXE	Preboot eXecution Environment
QM	Quality Managed
RAID	Redundant Array of Independent Disks
RAN	Radio Access Network
RAS	Reliability, Availability, and Serviceability
RDIMM	Registered Dual In-Line Memory Module
RDP	Remote Desktop
RMM	Remote Management Module
RoHS	Restriction of Hazardous Substances
SAS	Serial Attached SCSI (Small Computer System Interface)
SATA	Serial Advanced Technology Attachment
SCSI	Small Computer Systems Interface
SDRAM	Synchronous Dynamic RAM
SEL	System Event Log
SFP+	Small Form-factor Pluggable that supports data rates up to 10.0 Gbps
SMBus	System Management Bus
SMS	Server Management Software
SNMP	Simple Network Management Protocol
SOC	System on a Chip
SOL	Serial over LAN
SSD	Solid State Drive

SSH	Secure Shell
THOL	Tested Hardware and Operating System List
TPM	Trusted Platform Module
TUV	Technischer Uberwachungs-Verein (A safety testing laboratory with headquarters in Germany)
UART	Universal Asynchronous Receiver Transmitter
UEFI	Unified Extensible Firmware Interface
UL	Underwriter's Laboratory
USB	Universal Serial Bus
UV	Under-Voltage
V	Volt
VA	Volt-Ampere (volts multiplied by amps)
Vac	Volts Alternating Current
Vdc	Volts Direct Current
VDE	Verband Deutscher Electrotechniker (German Institute of Electrical Engineers)
VGA	Video Graphics Array
VPD	Vital Product Data
vRAN	Virtualized Radio Access Network
VSB	Voltage Standby
W	Watt
WEEE	Waste Electrical and Electronic Equipment
Ω	Ohm