F&S Windows IoT First Steps

Documentation

Version 1.0
(17.06.2026)

© F&S Elektronik Systeme GmbH
Untere Waldplätze 23
D-70569 Stuttgart
Germany

Phone: +49(0)711-123722-0
Fax: +49(0)711-123722-99

About This Document

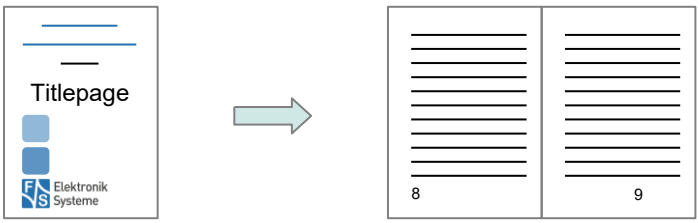
This document describes how to set up Windows 10/11 IoT Enterprise on your Single Board Computer or System on Module. This setup is crucial to ensure a seamless development experience with the F&S Board. The guide begins with a detailed walkthrough of the board setup process, including for example enabling remote access. Additionally, it covers the procedure for debugging a SBC/SoM using Visual Studio Remote Debugger on the Windows platform.

Remark

The version number on the title page of this document is the version of the document. It is not related to the version number of any software release. The latest version of this document can always be found at http://www.fs-net.de.

How to Print This Document

This document is designed to be printed double-sided (front and back) on A4 paper. If you want to read it with a PDF reader program, you should use a two-page layout where the title page is an extra single page. The settings are correct if the page numbers are at the outside of the pages, even pages on the left and odd pages on the right side. If it is reversed, then the title page is handled wrongly and is part of the first



double-page instead of a single page.

Typographical Conventions

We use different fonts and highlighting to emphasize the context of special terms:

`File names`

Menu entries

`Board input/output`

`Program code`

`PC input/output`

`Listings`

`Generic input/output`

`Variables`

History

Date	V	Platform	A,M,R	Chapter	Description	Au
2026-06-17	1.0	*	A	ALL	Create/Initialize Documentation	MR

V	Version
A,M,R	Added, Modified, Removed
Au	Author

Table of Contents

1 Foreword

As the support for Windows Embedded Compact 7/2013 comes to an end, F&S has proactively developed solutions to transition to Windows IoT on our boards. Our focus is on providing customers with a seamless and straightforward migration path to this modern platform.

Windows IoT offers enhanced security, improved performance, and a wide range of features suitable for diverse applications. The Long-Term Servicing Channel (LTSC) version guarantees extended support and stability, making it a reliable choice for long-term projects.

At F&S, we are committed to ensuring a successful transition for our customers. We have assembled a dedicated team of developers who meet daily to focus on the development and support of Windows IoT solutions. This ensures that we can address any challenges promptly and provide comprehensive support to our customers.

Our team is available to offer daily support for Windows IoT. If you require assistance, please reach out to us at support@fs-net.de.

For the latest updates and discussions about Windows IoT, we invite you to join our Windows IoT Forum. This platform provides the newest information and insights, allowing you to stay informed about the latest developments.

To access the latest software for Windows IoT, please register on our website. Registration grants you access to the most up-to-date software releases, ensuring your projects have the tools they need to succeed.

2 F&S Board Families and CPU Architectures

F&S offers a whole variety of Systems on Module (SOM) and Single Board Computers (SBC). There are different board families that are named NetDCU, PicoMOD, PicoCOM, armStone, QBliss, efus and PicoCore.

Family	Type	Size
NetDCU	Single Board Computer	80 mm x 100 mm
PicoMOD	System on Module	80 mm x 50 mm
PicoCOM	System on Module	40 mm x 50 mm
armStone	Single Board Computer	100 mm x 72 mm (PicoITX)
QBliss	System on Module	70 mm x 70 mm (Qseven)
efus	System on Module	62 mm x 47 mm
PicoCore	System on Module	40 mm x 35 mm

Table 1: F&S Board Families

Windows is not available on all of these platforms. Due to Microsoft licensing requirements and the NXP BSP package for Windows IoT, only specific CPU types are supported. The software, including drivers, therefore primarily comes from NXP and has been supplemented by our own F&S drivers.

OS	CPU	Platforms
Windows 10 IoT LTSC 2021	NXP i.MX8M-Plus	armStone8MP, PicoCore8MP, NetDCU8MP
Windows 11 IoT LTSC 2024	NXP i.MX93	PicoCOM93, PicoCore93, NetDCU93

Table 2: F&S Architectures

3 Getting started

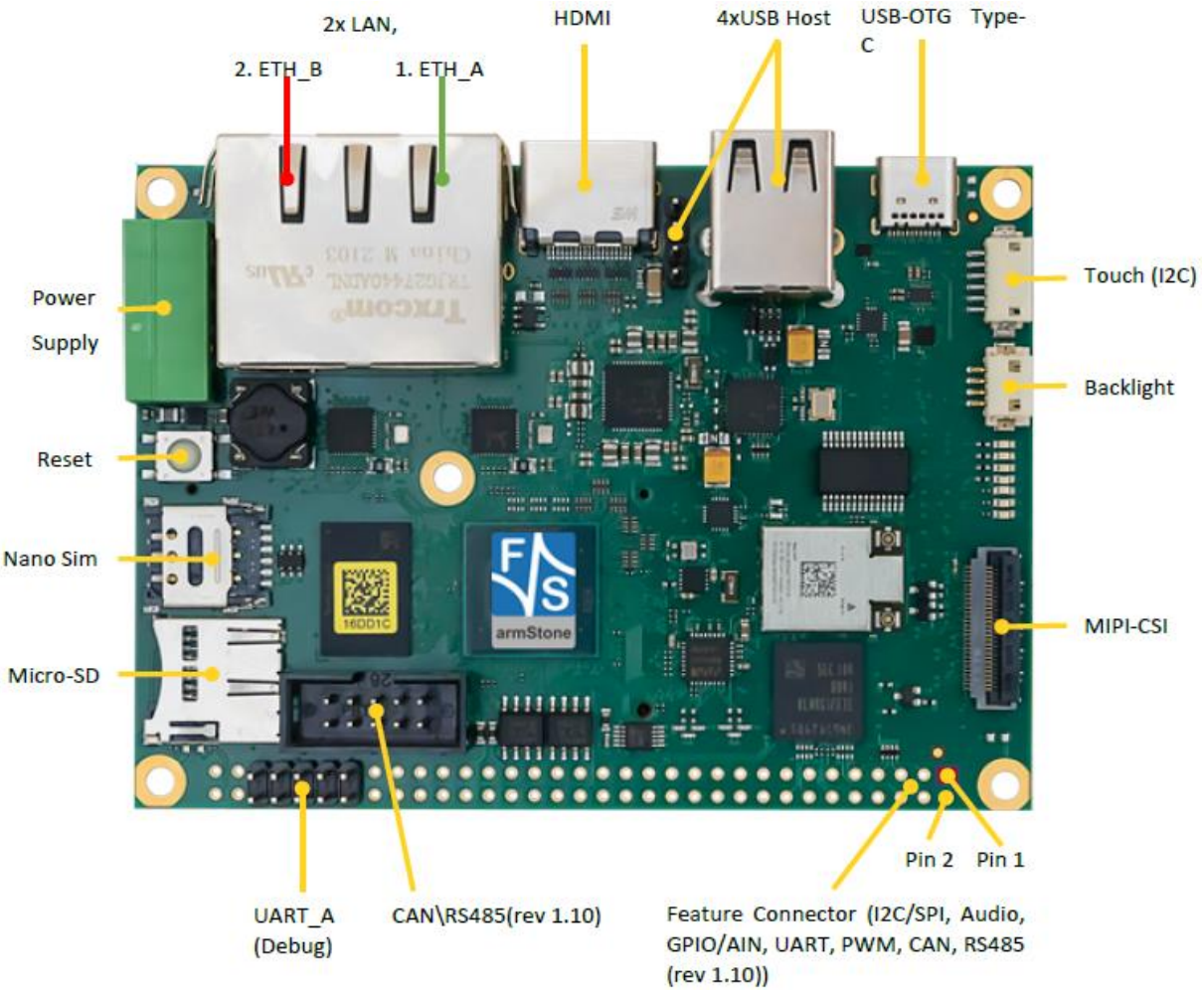
First, a step-by-step introduction describes how to set up the board and get started. All necessary steps required to begin working with the board are outlined. It is assumed that an F&S starter kit with the corresponding module is available.

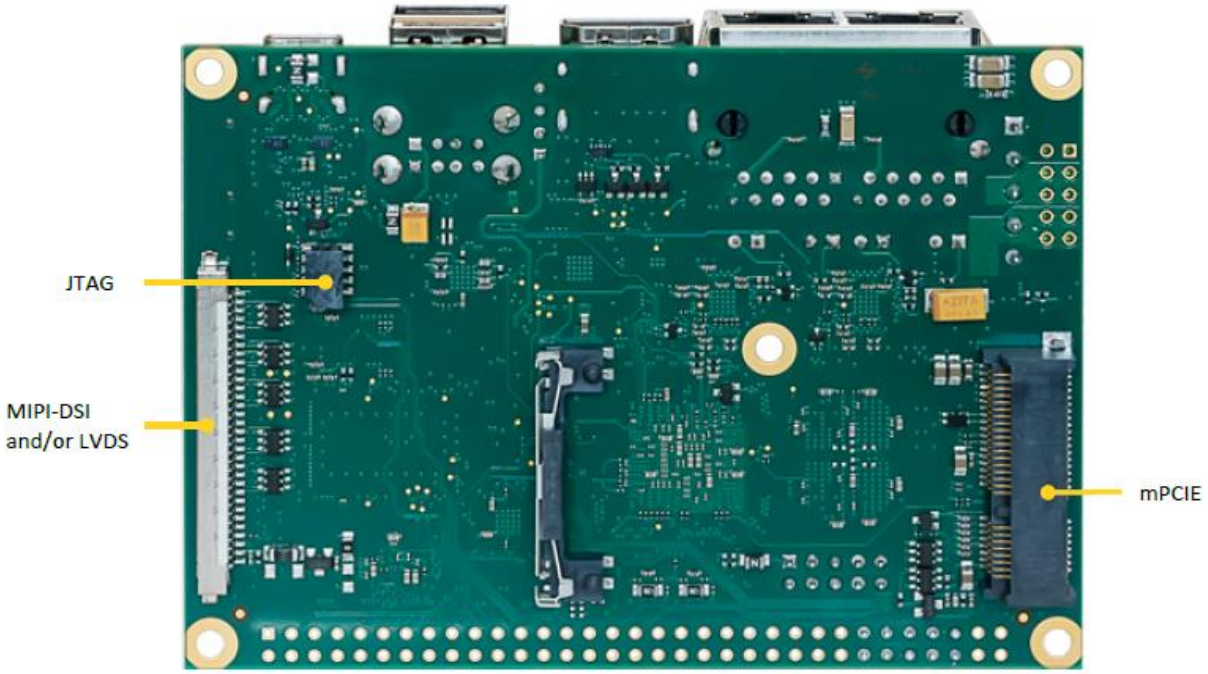
3.1 ArmStoneMX8MP

3.1.1 Interfaces

The Starterkit includes all components that are required for an initial setup. This includes:

- Cables (Serial, power, USB …).
- Software (source, binaries, install scripts, examples).
- armStoneMX8MP module.
- Display

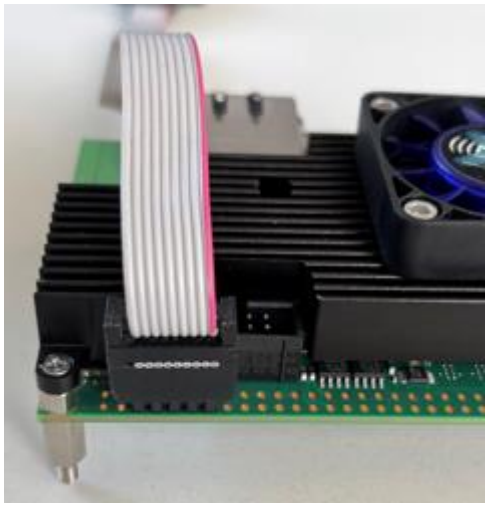




3.1.2 COM Port

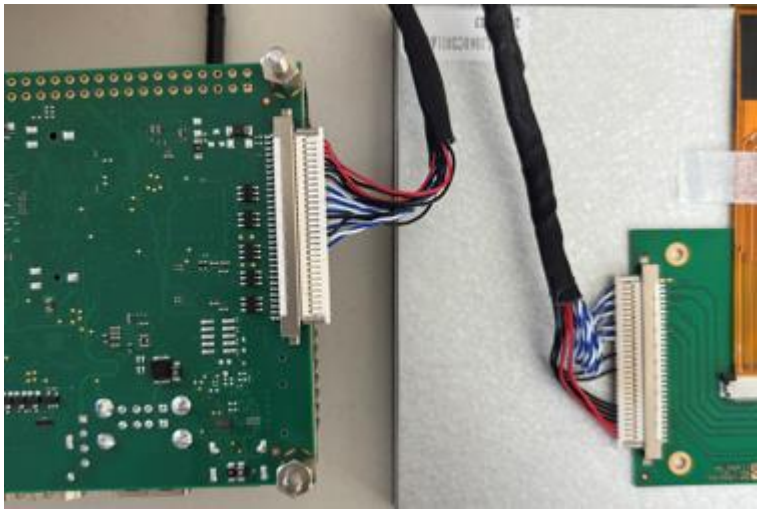
A serial connection is not strictly required, but it is very helpful to verify that the board boots correctly and to check which software versions are installed.

To do so, a serial connection between your PC and the board is required. Use the supplied null modem cable + adapter and connect the debug port of the armStone to the serial interface of your PC.



3.1.3 Display connection

The display cable is not symmetrical. Therefore, it must be connected as shown in the image.



Although not shown here, the other thin cable is used for the backlight and also needs to be connected.

3.1.4 Powering

Connect the board to a power supply. A supply voltage of +5 V is required.

Pin	Signal Name	I/O	Voltage	Description
1	N.C.	X		Not Connected
2	VDD_VBAT [2]	PWR	3.0 V	External RTC Supply Voltage (optional)
3	VCC_IN	PWR	5.0 V	Supply Voltage (board)
4	GND			
5	VDD_3V3	O	3.3 V	Feedback (if board is powered up)

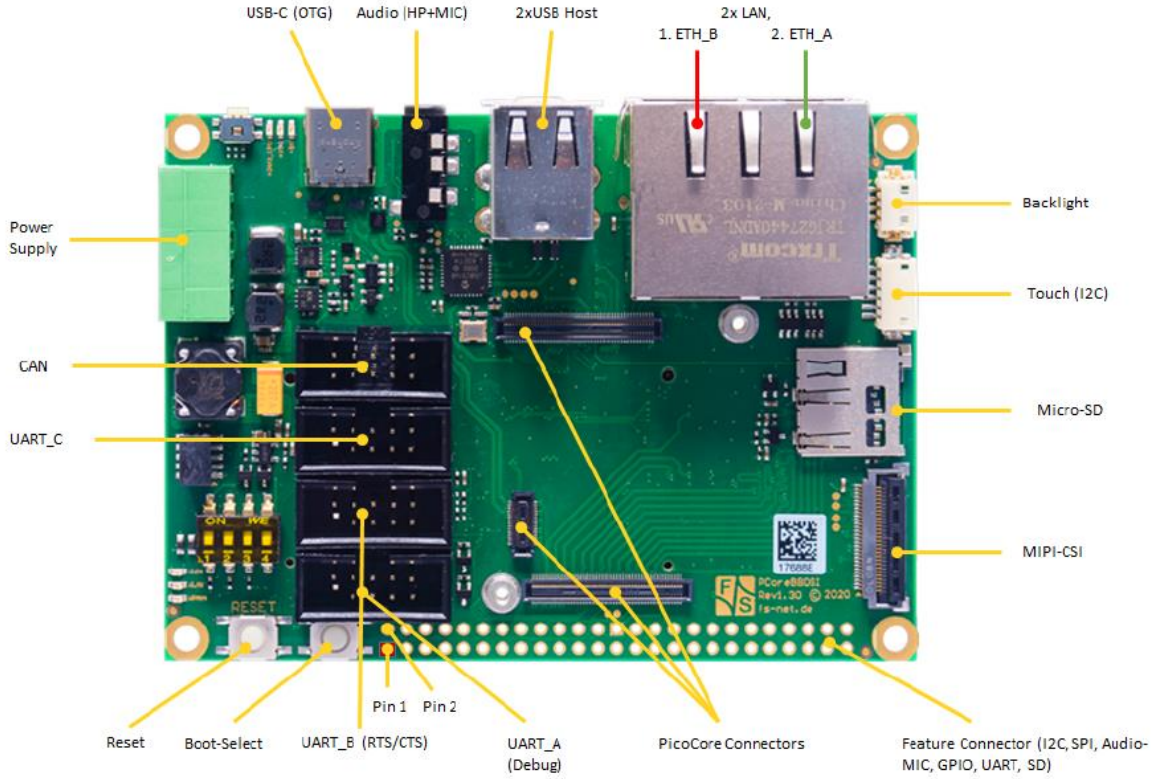
For this just connect the green plug to the green socket on the board.

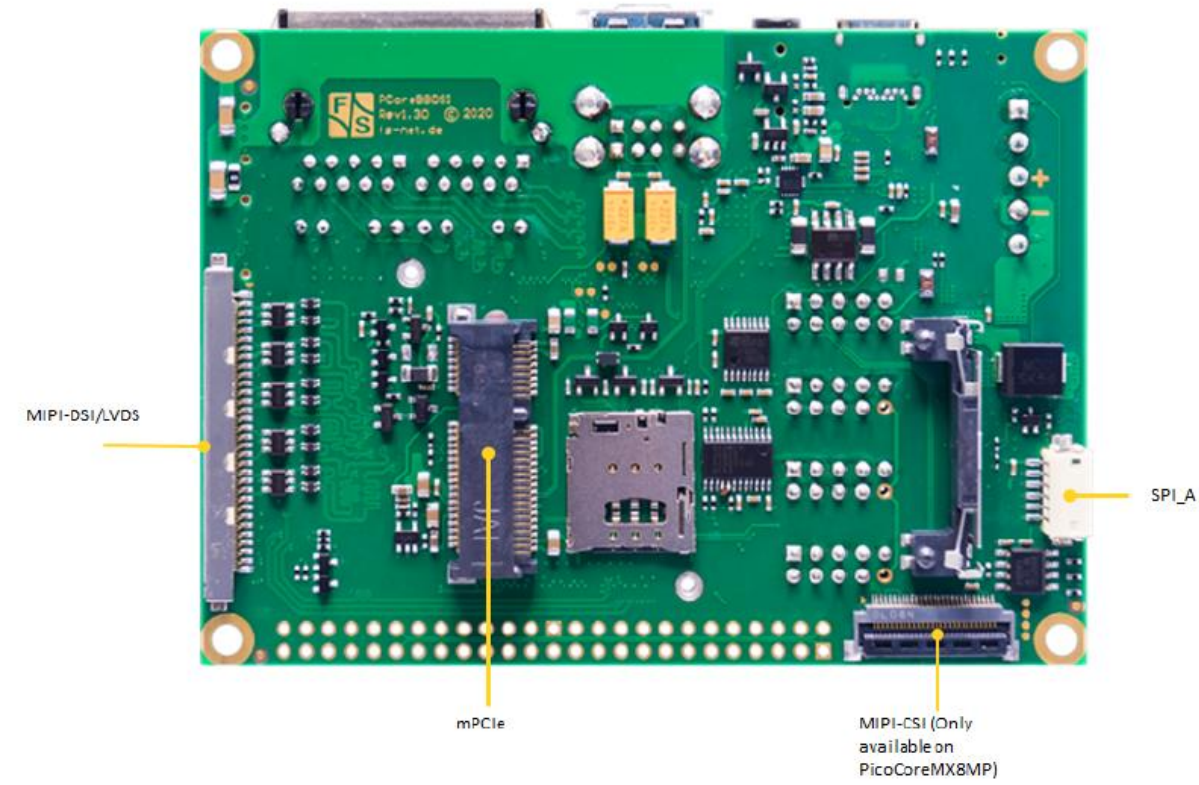
3.2 PicoCoreMX8MP / PicoCoreMX93

3.2.1 Interfaces

The Starterkit includes all components that are required for an initial setup. This includes:

- Cables (Serial, power, USB …).
- Software (source, binaries, install scripts, examples).
- picoCoreMX8MP module
- picoCoreMX8MP baseboard (PicoCoreBBDSI)
- Display

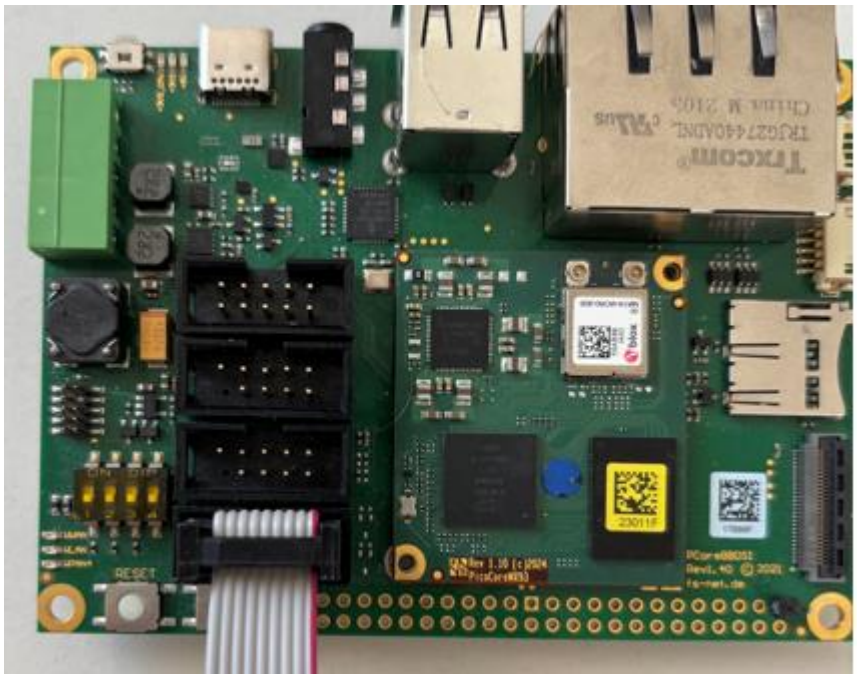




3.2.2 COM Port

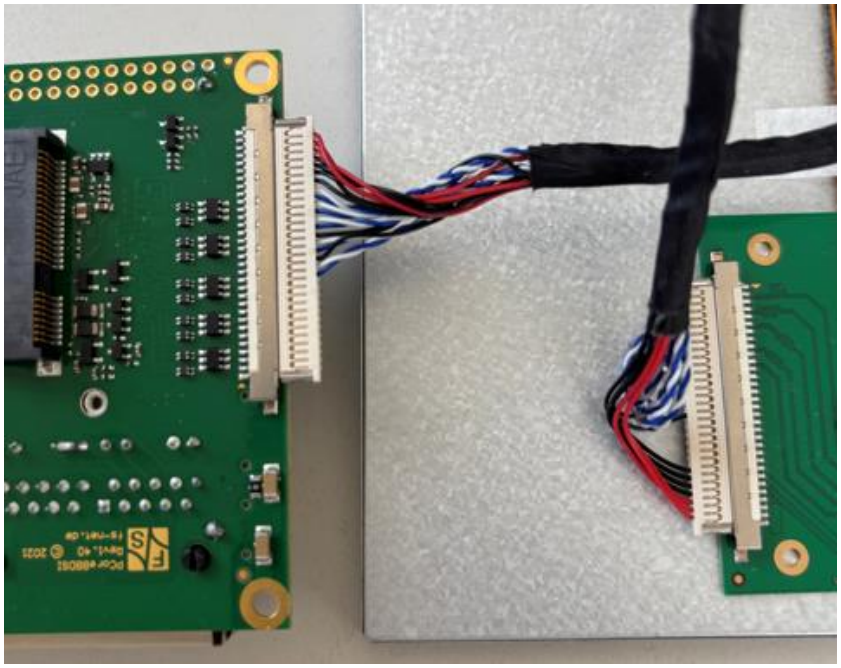
A serial connection is not strictly required, but it is very helpful to verify that the board boots correctly and to check which software versions are installed.

To do so, a serial connection between your PC and the board is required. Use the supplied null modem cable and connect the debug port of the starter kit baseboard to the serial interface of your PC.



3.2.3 Display connection

The display cable is not symmetrical. Therefore, it must be connected as shown in the image.



Although not shown here, the other thin cable is used for the backlight and also needs to be connected.

3.2.4 Powering

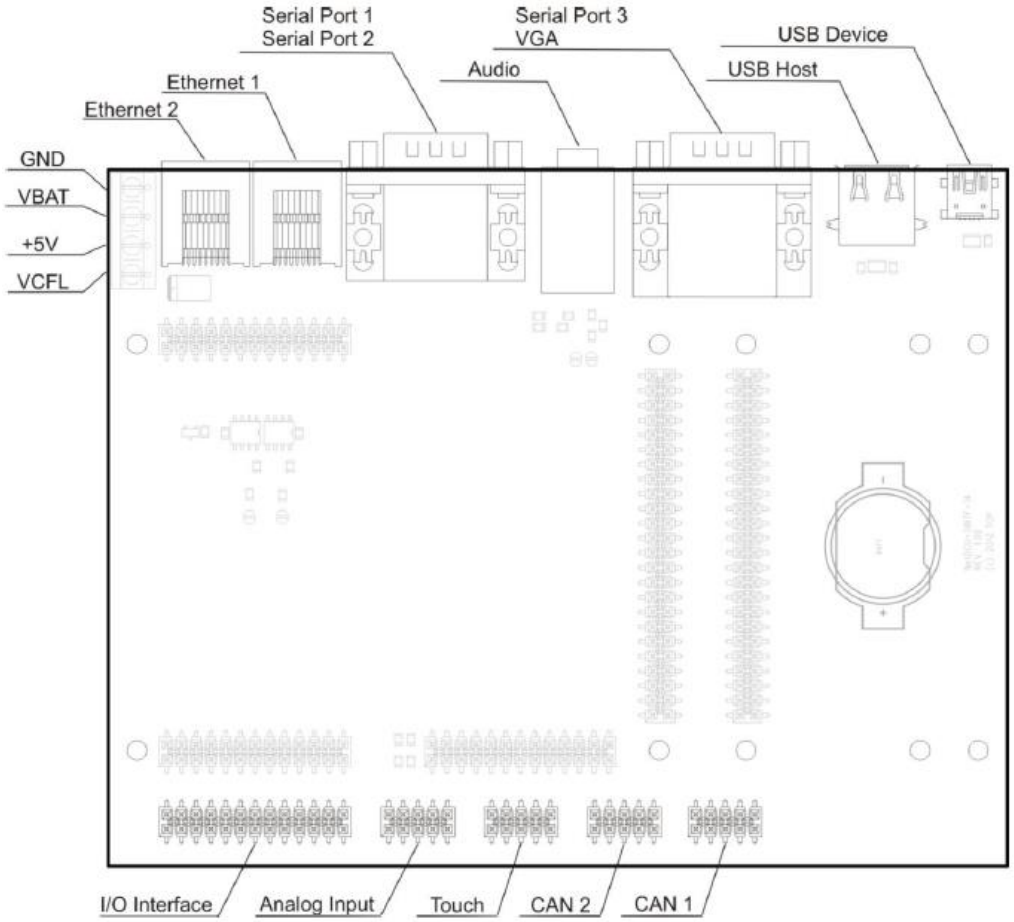
Connect the board to a power supply. A supply voltage of +5 V is required.

Pin	Signal Name	I/O	Voltage	Description
1	N.C.	X		Not Connected
2	VDD_VBAT [2]	PWR	3.0 V	External RTC Supply Voltage (optional)
3	VCC_IN	PWR	5.0 V	Supply Voltage (board)
4	GND			
5	VDD_3V3	O	3.3 V	Feedback (if board is powered up)

For this just connect the green plug to the green socket on the board.

3.3 NetDCUMX8MP / NetDCUMX93

3.3.1 Interfaces

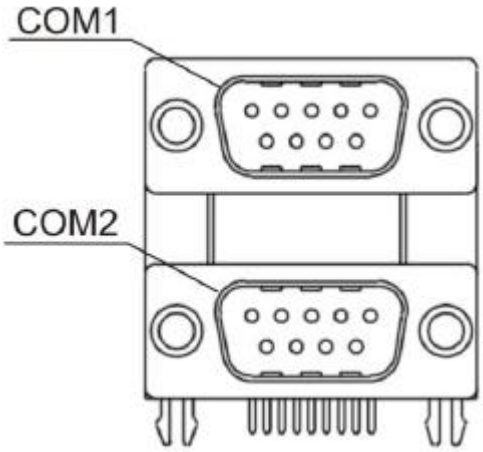


3.3.2 COM Port

A serial connection is not strictly required, but it is very helpful to verify that the board boots correctly and to check which software versions are installed.

To do so, a serial connection between your PC and the board is required. Use the supplied null modem cable and connect the debug port of the starter kit baseboard to the serial interface of your PC.

The starter kit provides a double 9-pin Sub-D connector. Connect the supplied RS232 cable to the COM1 port.

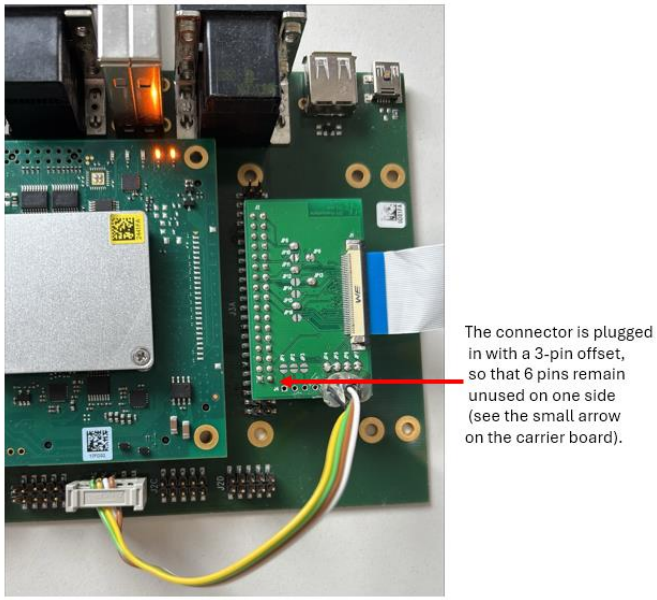


3.3.3 Display connection

In certain variants of the module, an LVDS display connector is available on the top side. Alternatively, for compatibility with older NetDCUs, an RGB connector is provided on the bottom side of the board (connector J3).

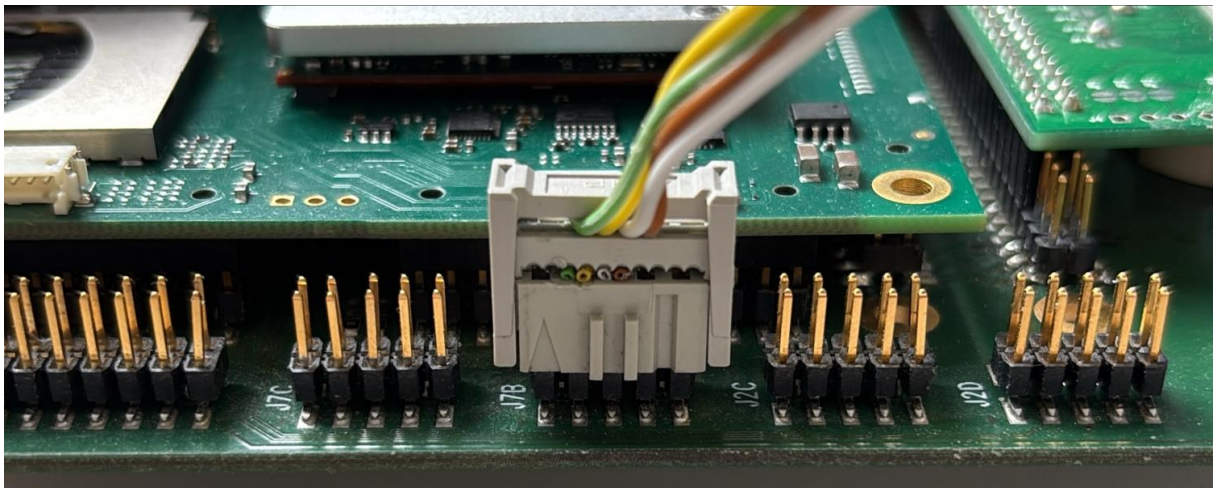
The software already includes several predefined display timings (see here). However, if you would like to integrate support for your own display in the future, please contact mailto:support@fs-net.de.

The kit includes an ET0070 RGB display and an ADP-EDT display adapter. The adapter is connected as follows:



To enable touch functionality for the display, the cable from the display adapter must be connected to the carrier board as shown in the image.

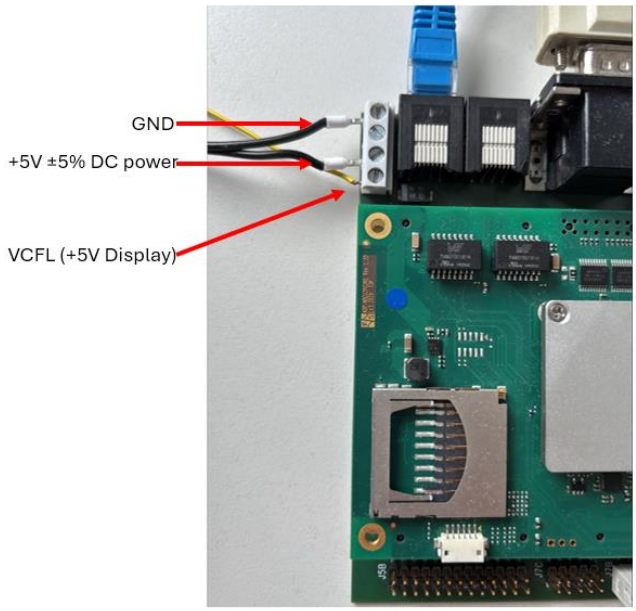
To do this, simply plug in the connector with the correct orientation (Pin 1 is marked by a triangle) as follows:



3.3.4 Powering

Connect the board to a power supply. A supply voltage of +5 V is required.

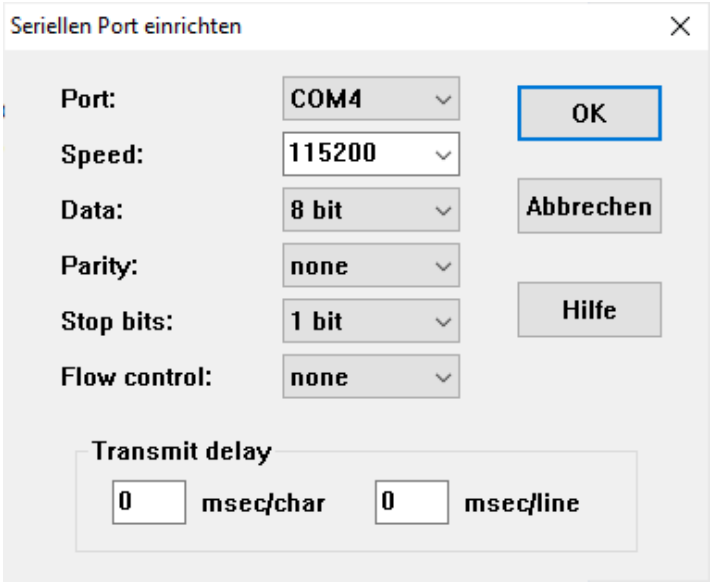
The display (yellow cable) also requires a +5 V supply. Make sure that both connections share the same ground (GND).



4 Serial output

4.1 Output Messages

U-Boot and UEFI output messages serially, version numbers or which drivers were loaded. These messages can be easily displayed using a terminal program like TeraTerm. To do this, simply establish a serial connection to the board as described in sections 3.1.2, 3.2.2, and 3.3.2. Then, open a new COM connection in the terminal program:



Note: COM-Port may be different on your computer.

4.2 Kernel Debugging

The kernel and drivers can be debugged using WinDbg, Microsoft's debugging tool, which should already be installed on every Windows PC.

Follow the steps below, to establish a connection:

- Start WinDbg on your PC
- Within WinDbg, go to the "File" tab in the menu bar
- Select "Kernel Debug..." from the "File" dropdown menu. This will open the "Kernel Debugging" dialog box
- Switch to the "COM" tab inside the "Kernel Debugging" dialog box
- Enter the correct COM port that you want to use for debugging the F&S Board. Make sure to specify the COM port associated with the F&S Board

Serial output

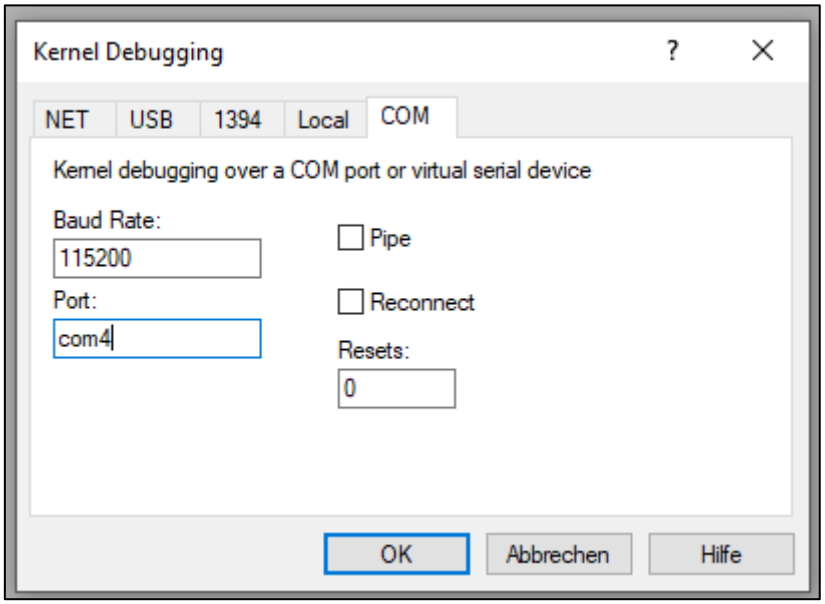
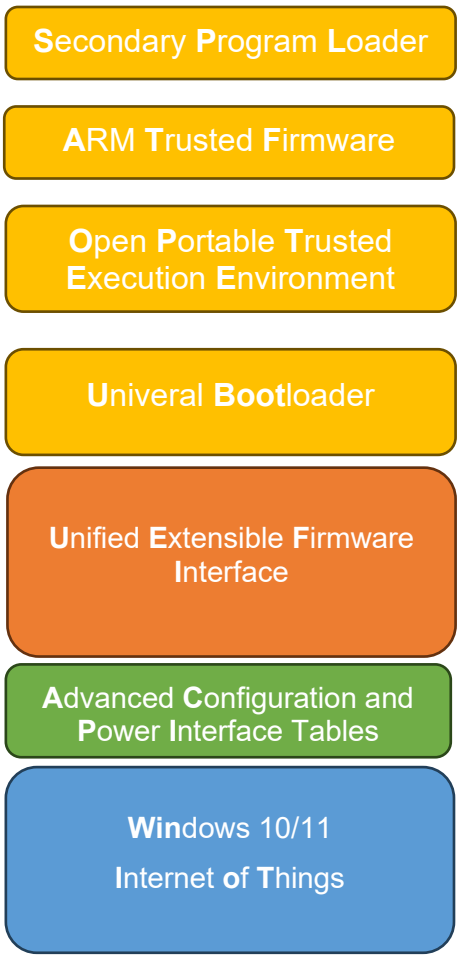


Figure 1: WinDbg configuration

Note: COM-Port may be different on your computer. Start kernel debugging only after the UEFI has fully loaded, as anything else may result in freezing (UEFI is fully loaded when a W-logo appears on the display).

5 Firmware

5.1 Boot process



5.2 Versioning

5.2.1 BugTracker

The Windows IoT project is actively maintained with the help of a bug tracker. This tool is essential for ensuring the smooth operation and continuous improvement of our IoT solutions. It allows you to view the current bugs and features being worked on.

To access this information, select the *Windows IoT* project from the dropdown menu in the top-left corner of the page. Then navigate to the *Roadmap* tab to get an overview.

To get to the MantisBT click here.

Firmware

click here.

5.2.2 Uboot

The version information can be obtained via a serial terminal as part of the debug output. A serial connection to the board is required for this.

Refer to section 4.1 of this document for instructions on how to establish the serial connection. The message appears as follows (can be various) :

```
U-Boot 2024.04-00016-g50fc4e06f-dirty (Feb 13 2026 - 12:54:48 +0100)
```

5.2.3 UEFI

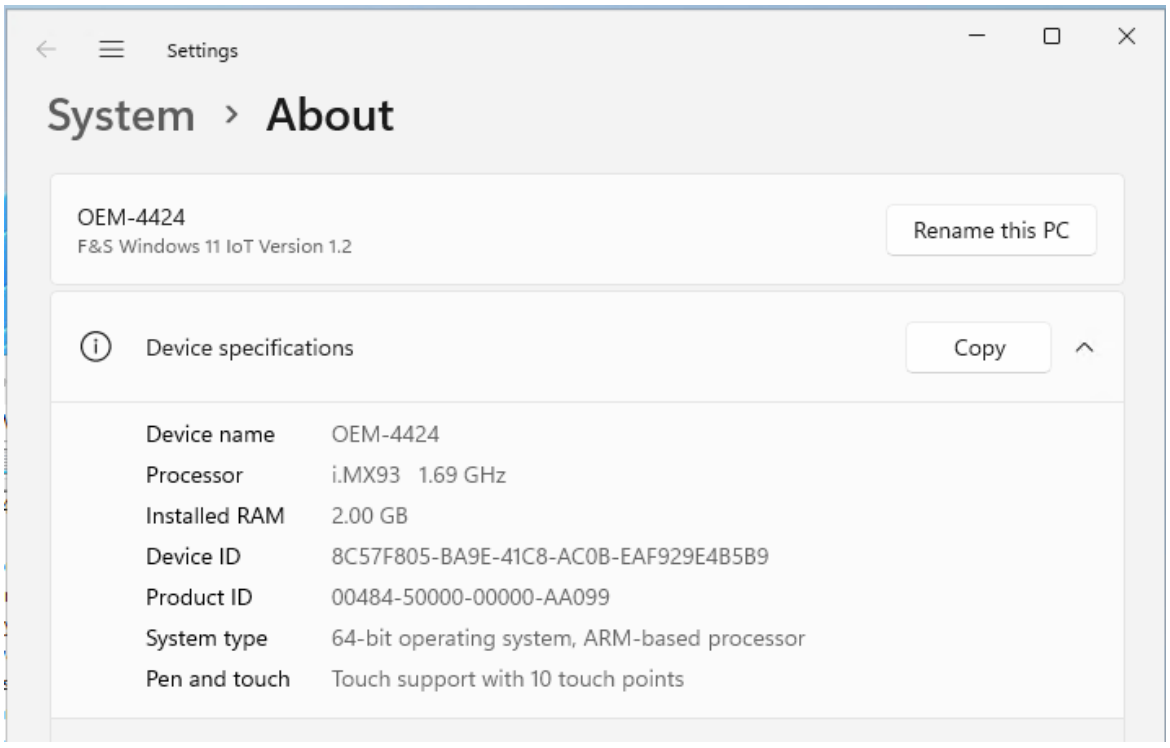
The version information can be obtained via a serial terminal as part of the debug output. A serial connection to the board is required for this.

Refer to section 4.1 of this document for instructions on how to establish the serial connection. The message appears as follows (can be various):

```
UEFI firmware (version 2026-01-01 built at 16:43:56 on Jun 16 2026)
```

5.2.4 OS

In The F&S Windows version can be found in the system settings under **“About your PC.”** This section also provides all relevant information about the device, as well as the Windows specifications.



6 Tools

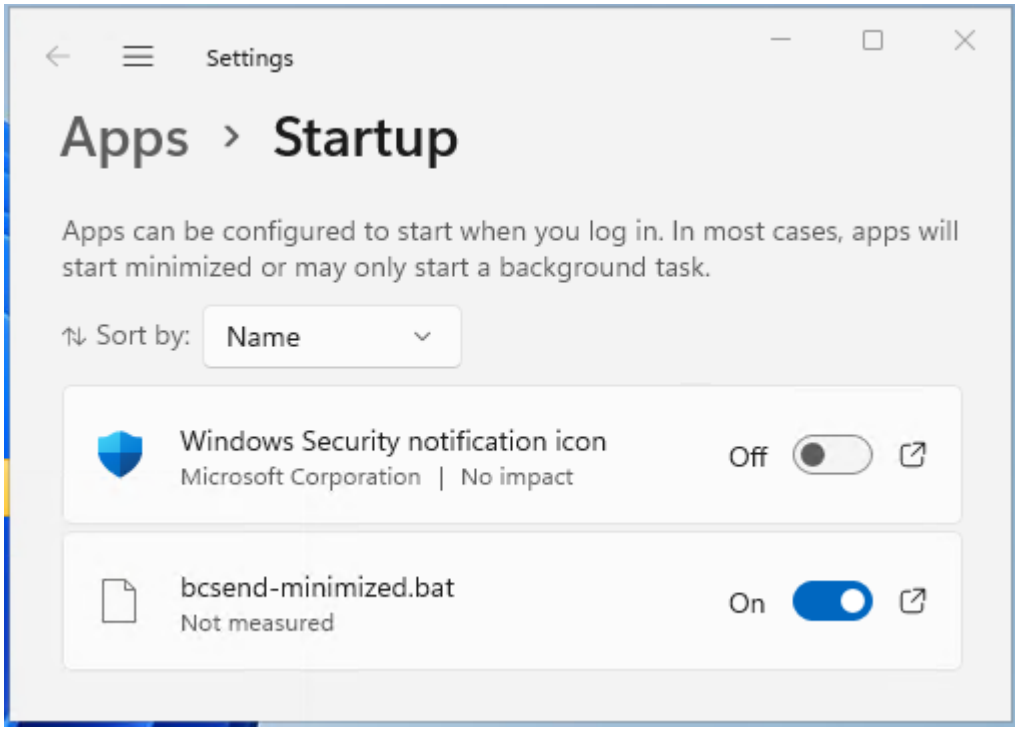
6.1 FSDeviceSpy

In the Windows IoT environment, F&S Boards usually are assigned an IP address via DHCP.

Identifying the current IP address of these boards can be challenging, especially without a connected display. But there's a light at the end of the tunnel...

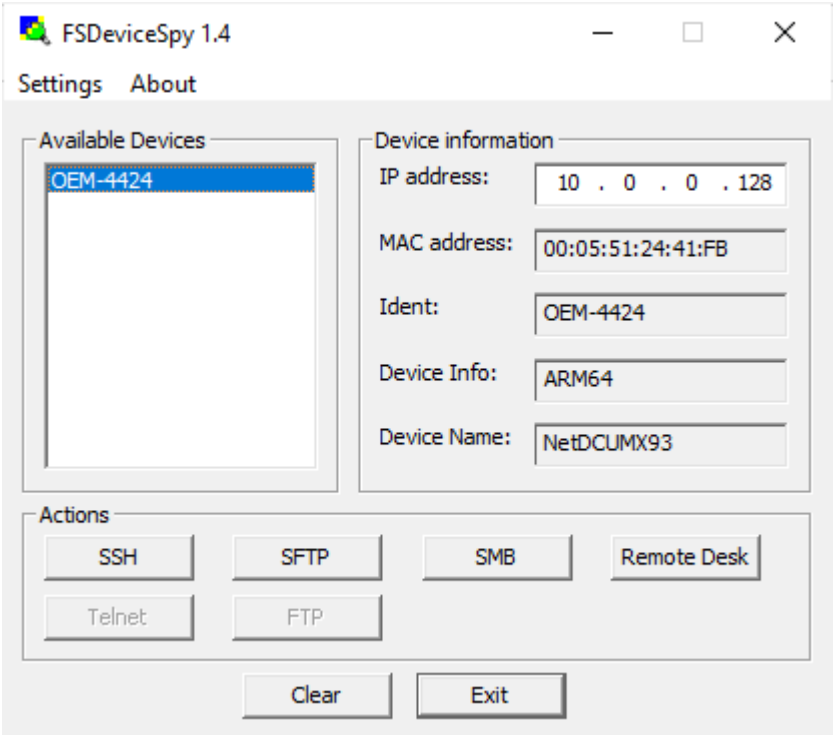
To address this, F&S has developed a new application '**BCSend**' that automatically sends broadcast messages from your WinIoT F&S Boards for some time after booting.

This application is implemented on every F&S Windows IoT board. If you don't need this application, you can easily turn it off in the settings:



Now, the question arises: who receives the broadcast? This is where the additional application, **FSDeviceSpy**, comes into play. FSDeviceSpy tool can receive these broadcast messages. Once a broadcast from an F&S Board (in this case OEM-4424) is received, users can establish various connections to the F&S Board, like SSH or Remote Desktop:

Tools



You can download this tool from our forum posts.

6.2 Interface Tools

Microsoft provides a wide range of tools and samples for Windows IoT, including the bus tools. These tools have proven to be extremely helpful in assessing the performance and reliability of the various interfaces on our F&S boards. Thanks to these tools, we can conduct various test scenarios to ensure that our hardware seamlessly interacts with Windows IoT.

For those of you who are working with or considering Windows IoT, it's great to know that these tools are free to use. You can download it from the following page.

In general, these tools utilize the Windows.Devices Namespace. Within this namespace, you'll find sub-namespaces with specific types that allow you to communicate with various peripheral devices directly from your application. The sub-namespaces are the following:

- I2C (I2cTransferResult, I2cSharingMode, ...)
- UART (CreateFileW, ReadFile, WriteFile ...)
- PWM (PwmPulsePolarity, PWMPin.Start ...)
- SPI (SpiMode, ISpiDeviceStatics ...)
- GPIO (GpioPinDriveMode, GpioPinValue ...)

CAN TestTool

However, it's important to note that, there's no official test tool provided by Windows for the CAN interface. In response to this, F&S has taken the initiative to develop its own CAN testing tool. This tool is structured in a manner similar to existing Microsoft test tools, ensuring familiarity for users.

This CAN test tool makes use of the official Input/Output Controls (IOCTL) from the NXP FlexCAN driver. For a more detailed understanding of these IOCTLs, you can refer to the project CAN header file.

You can download this tools from our forum post.

6.3 Visual Studio Remote Debugger

To debug a Visual Studio application on SBC/SoM, several actions must be performed. On SBC/SoM the remote tools (i.e. Remote Debugger) need to be installed and launched. Additionally, the project must be set up in visual studio so that a connection to the SBC can be established. Once configured, you can build your project and it will automatically deploy and execute the application on SBC/SoM.

Note: Using a User on SBC/SoM without a password, it is essential to allow a passwordless connection.

To establish a remote debugger connection from the SBC, the following steps must be performed:

- First of all you have to activate the network discovery & file & printer sharing. You can find this settings under
  Control Panel > Network and Internet > Network and Sharing Center > Advanced sharing settings

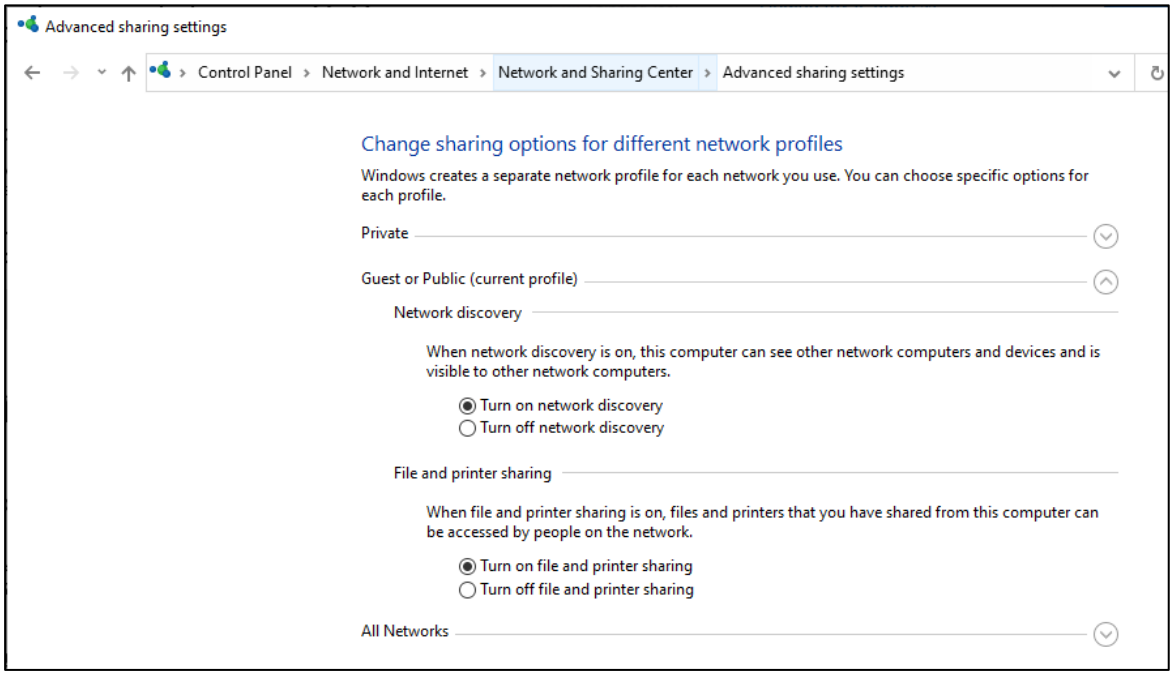


Figure 2: Network and file sharing configuration

- For the Visual Studio application, create a target directory on the SBC/SoM. In order for this target directory to be accessible from the developer PC, it needs to be shared, either for all users or for the specific (local) user.

  For example:
  Create a directory named "debug" in the root directory (C:\) of the SBC/SoM. Afterward, access the properties of this directory. Proceed to the "Sharing" tab and click on the "Share" button. Select at least the local user of the SBC/SoM to share the directory with.

Now it should be possible to access this “debug” directory via File explorer from the PC.

After all important prerequisites have been met, we proceed with the actual installation and setup of the Remote Debugger Tool.

For this purpose, you will need the following file: VS2019_RemoteTools.exe. This file can be downloaded directly from Microsoft Visual Studio or from the download section on our website.

(Path: Downloads\VisualStudio\RemoteDebugger\VS2019_RemoteTools_ARM64)

Run the .exe file on the SBC/SoM and proceed with the installation.

After the application has been successfully installed run this “Remote Debugger” application. Confirm the Remote Debugger Configuration as follows:

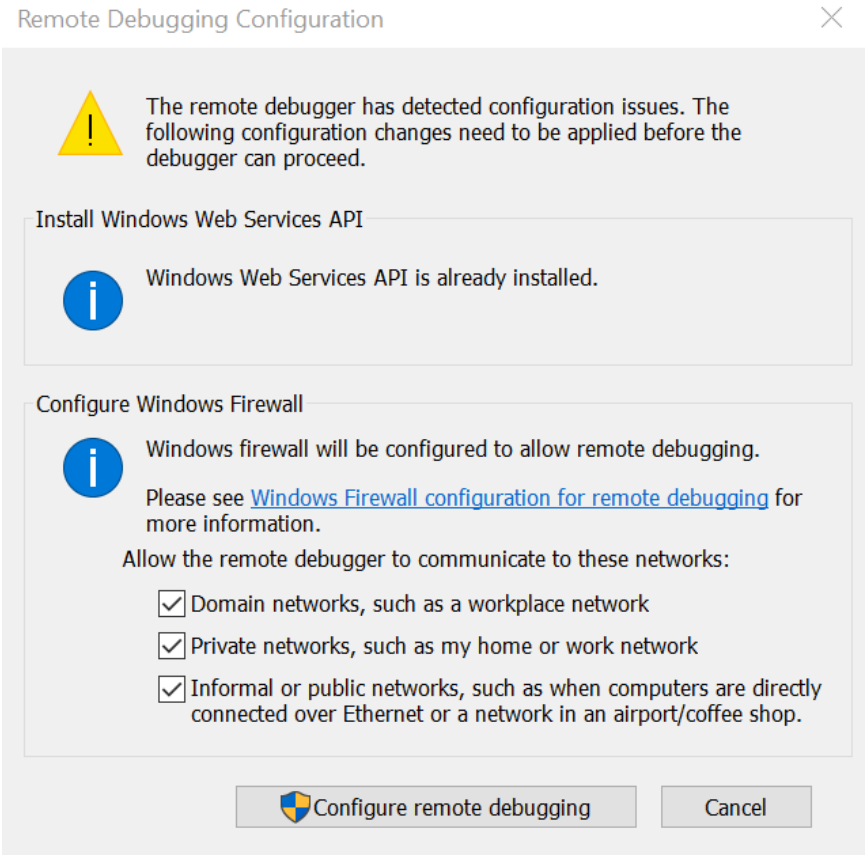


Figure 3: Remote Debugging Configuration

The Remote Debugger is a small dialog that displays notifications. We will keep this dialog open and now focus on the developer machine that wants to access it.

Tools

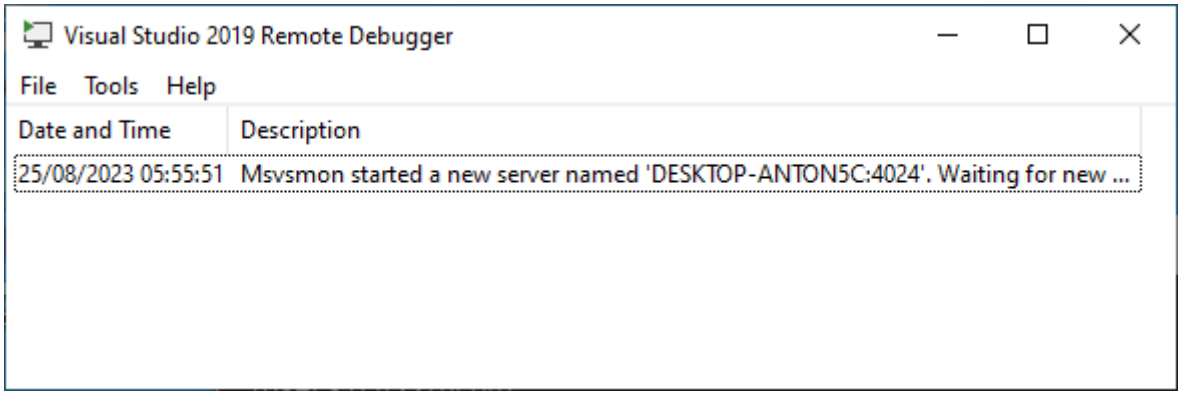


Figure 4: VisualStudio 2019 Remote Debugger

To establish a remote debugger connection from the developer PC, the following steps must be performed:

In the project settings of the C++ project, you have to navigate to the Debugger configuration. Under Debugging, the debugger to be launched is selected. Refer to the image below:

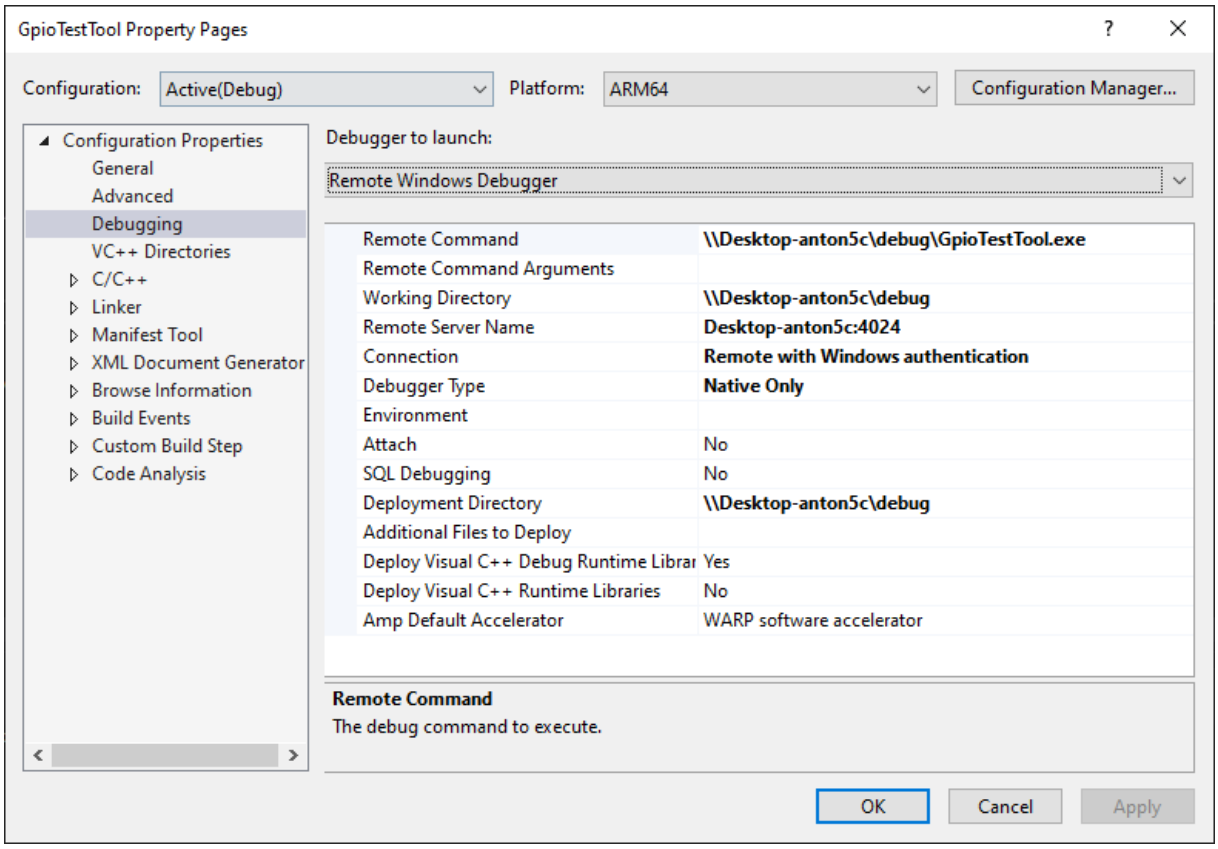


Figure 5: Project Configuration Properties Debugging

Described below are the key parameters required for configuring remote debugging settings:

Parameter	Description
Remote Command	Refers to the pathway leading to the executable file of your application located on the remote server (Single-Board Computer/System-on-Module - SBC/SoM).
Working Directory	Designates the location where the remote command is executed. Utilizing the browse function, you can directly select the previously created "debug" directory. This step is highly recommended because it ensures the accurate pathway. Additionally, this path can be employed for other parameters, such as the remote command or the deployment directory.
Remote Server Name	Connection details of the target machine (SBC/SoM). The information regarding the machine's name and port can be found in the Remote Debugger dialog see figure 9. In our case it is: DESKTOP-ANTON5C:4024
Debugger Type	Determines which type of debugger will be employed to remote debugging. Based on our platform, we choose the "native only" option.
Deployment Directory	Path on the remote server(SBC/SoM) where your application files or resources are copied before debugging commences. This step is essential to ensure that the remote application has all the required files to function properly and be debugged effectively.

After configuring the debugging settings, the final step is to set up the deployment of files. This can be achieved by accessing the Configuration Manager within the project solution (right-click on the project > Configuration Manager).

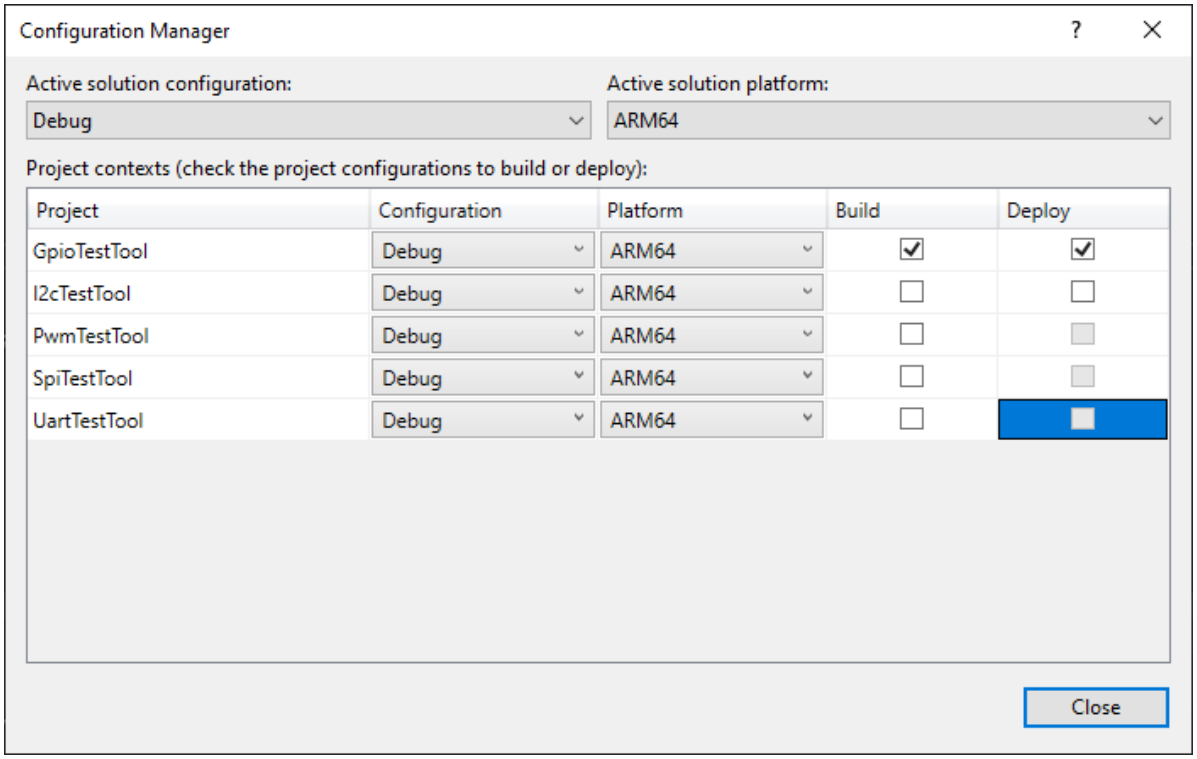


Figure 6: Project Configuration Manager deploy

It is important here to ensure the correct active Solution configuration and platform are selected. Also, for the specific project (in this case: GPIOTestTool), make sure to check both BUILD and DEPLOY.

With these steps completed, all settings for remote debugging should now be in place.

7 Driver

7.1 Display

The driver is particularly configurable over the UBoot environment.

7.1.1 Interface

First, you need to access the U-Boot. This is done by establishing a serial connection to the board using a terminal program like TeraTerm (see 3.1) and pressing a key while the board is rebooting. Once in U-Boot, you can set a display interface or panel using the command

setenv displayinterface [0-4]

setenv displaypanel [0-5]

After entering the command, you need to save the environment using the command

saveenv

Entry	Value	Description
Displayinterface	DWORD	0x0: HDMI 0x1: MIPI-DSI 0x2: LVDS0 0x3: LVDS1 (default) 0x4: LVDS Dual
Displaypanel	DWORD	0x0: FS 7” LVDS LCD Display (default) 0x1: FS 10” LVDS LCD Display 0x2: 10" LVDS Display G104STN01 0x3: 7" LVDS Display J070WVTC0211 0x4: 7" TFT (WVGA) EDT ET070080DH6 0x5: 10” TFT LCD EV101WXM-N80

7.2 Touch

7.2.1 Ilitek

The enable/disable driver registry key for the driver is:

[HKLM\SYSTEM\CurrentControlSet\Services\TouchIlitekKd}]

Driver

Entry	Value	Description
Start	DWORD	Startup type of the driver 0x0: boot 0x1: system 0x2: automatic 0x3: manual (default) 0x4: disabled

7.2.2 Goodix

The enable/disable driver registry key for the driver is:

[HKLM\SYSTEM\CurrentControlSet\Services\TouchGoodixKd}]

Entry	Value	Description
Start	DWORD	Startup type of the driver 0x0: boot 0x1: system 0x2: automatic 0x3: manual 0x4: disabled (default)

7.2.3 Focaltech

The enable/disable driver registry key for the driver is:

[HKLM\SYSTEM\CurrentControlSet\Services\TouchFocaltechKd}]

Entry	Value	Description
Start	DWORD	Startup type of the driver 0x0: boot 0x1: system 0x2: automatic 0x3: manual 0x4: disabled (default)

7.2.4 TSC2004

The enable/disable driver registry key for the driver is:

[HKLM\SYSTEM\CurrentControlSet\Services\TouchTsc2004Kd}]

Entry	Value	Description
Start	DWORD	Startup type of the driver 0x0: boot 0x1: system 0x2: automatic 0x3: manual 0x4: disabled (default)

The configuration registry key for the driver is:

[HKLM\SYSTEM\CurrentControlSet\Enum\ACPI\TSC0001\2&daba3ff&0\Device Parameters}]

Entry	Value	Description
SwitchXY	DWORD	Switch the X and Y-axes 0x0: no switch (default) 0x1: switch
InvertX	DWORD	Invert all X-coordinates 0x0: not invert 0x1: invert (default)
InvertY	DWORD	Invert all Y-coordinates 0x0: not invert (default) 0x1: invert
PanelVoltageStabilization	DWORD	Amount of time between touch screen drivers enable and voltage sampled/conversion started 0x0: 0µs 0x1: 100µs 0x2: 500µs 0x3: 1ms 0x4: 5ms (default) 0x5: 10ms 0x6: 50ms 0x7: 100ms

PrechargeTimeSelection	DWORD	These bits set the amount of time allowed for pre-charging any pin capacitance on the touch screen prior to sensing if a pen touch is happening 0x0: 20µs 0x1: 84µs 0x2: 276µs 0x3: 340µs 0x4: 1044ms (default) 0x5: 1108ms 0x6: 1300ms 0x7: 1364ms
SenseTimeSelection	DWORD	Sense time selection. These bits set the amount of time the TSC2004 waits to sense whether the screen is touched after converting a coordinate 0x0: 32µs 0x1: 96µs 0x2: 544µs 0x3: 608µs 0x4: 2080ms (default) 0x5: 2144ms 0x6: 2592ms 0x7: 2656ms
BatchDelaySelection	DWORD	These are the selection bits that specify the delay before a sample/conversion scan cycle is triggered. Scans per second = 1/BatchDelaySelection[ms] 0x0: 0ms 0x1: 1ms 0x2: 2ms 0x3: 4ms 0x4: 10ms (default) 0x5: 20ms 0x6: 40ms 0x7: 100ms

7.3 GPU

The registry key for enabling/disabling this driver:

[HKLM\SYSTEM\CurrentControlSet\Control\Class\{4d36e968-e325-11ce-bfc1-08002be10318}]

Entry	Value	Description
Load	DWORD	Enable/Disable Driver 0x0: Disable (default) 0x1: Enable

NOTICE: It is not recommended to enable the GPU, as even minor errors in the EDID structure may result in no display output. Please contact us in advance at support@fs-net.de before enabling this feature.

The registry key to configure display parameters / timings is:

[HKLM\SYSTEM\CurrentControlSet\Control\Class\{4d36e968-e325-11ce-bfc1-08002be10318}\000X]

(Notice: X must be replaced with the right number of the GPU Driver - can be various)

Entry	Value	Description
DriverDesc	SZ	Name of display driver “i.MX GPU Device” (default)
EnableMultiMon	DWORD	Under this entry you can enable/disable the support of multiple display interfaces. If enabled the first display is determined by the *Display0Interface* parameter, the second display is determined by the *Display1Interface* parameter and so on 0x0: single display 0x1: multiple displays (default)
Display<n>Interface	DWORD	Select display interface for the n display 0x1: DISP_INF_HDMI (default *Display1Interface)* 0x2: DISP_INF_DSI0 0x3: DISP_INF_DSI1 0x4: DISP_INF_LVDS0 0x5: DISP_INF_LVDS1 (default *Display0Interface*) 0x6: DISP_INF_DUAL0 0x7: DISP_INF_PARALLEL_LCD
Display<n>EDID	BINARY	The parameter contains EDID data encoded according to the EDID v1.3 structure (VESA). The first 128-

		byte data block is used, representing the basic EDID structure without any extensions. Resolution and timing parameters are extracted from the Standard Timing Information Descriptor 1 (offsets 54–71), including the pixel clock, horizontal and vertical active pixels, blanking pixels, synchronization pulse width, front porch, and the polarity of the VSYNC and HSYNC signals. During EDID loading from the registry, the EDID header (offsets 0–7) is verified, and the checksum (offset 127) must match. Other EDID data are ignored by the GPU driver. 00 FF FF FF FF FF FF 00 04 21 00 00 00 00 00 00 01 00 01 03 80 00 00 00 00 00 00 00 00 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 01 00 BC 1B 00 44 41 58 24 20 A0 04 C1 00 00 00 00 00 00 18 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AF (default *Display0EDID*)
Display<n>BusDataWidth	DWORD	Only related to LVDS Interface. This parameter determines the number of pixels mapped to the output signal. 0x12: 18 bpp 0x18: 24 bpp (default)
Display<n>BusMapping	DWORD	Only related to LVDS Interface. This parameter determines the pixel-mapping type in the output signal. 0x1: DISP_BUS_MAPPING_SPWG (default) 0x2: DISP_BUS_MAPPING_JEIDA
Display<n>NumLanes	DWORD	Only related to MIPI-DSI Interface. This parameter determines the number of DSI lanes. 0x1: One Lane 0x2: Two Lanes 0x3: Three Lanes 0x4: Four Lanes (default)
Display<n>ChannelId	DWORD	Only related to MIPI-DSI Interface. This parameter determines the virtual channel ID of the display. 0x0: Channel ID 0 (default)

7.4 UDPusher

The registry key for the driver is:

[HKLM\SYSTEM\CurrentControlSet\Enum\ACPI\UDP0001\2&daba3ff&0\Device Parameters}]

Entry	Value	Description
Load	DWORD	Enable/Disable Driver 0x0: Disable 0x1: Enable (default)
UDP_Left	DWORD	Under this entry, you can assign a HID keyboard key scancode that will be executed when the encoder is rotated to the left … 0x04: Keyboard a and A 0x05: Keyboard b and B … 0x50: Keyboard LeftArrow (default) …
UDP_Right	DWORD	Under this entry, you can assign a HID keyboard key scancode that will be executed when the encoder is rotated to the right … 0x04: Keyboard a and A 0x05: Keyboard b and B … 0x4F: Keyboard RightArrow (default) …
UDP_Push	DWORD	Under this entry, you can assign a HID keyboard key scancode that will be executed when the encoder is pushed … 0x04: Keyboard a and A 0x05: Keyboard b and B …

		0x28: Keyboard Return ENTER (default) …

Here you can find all possible keycode inputs.

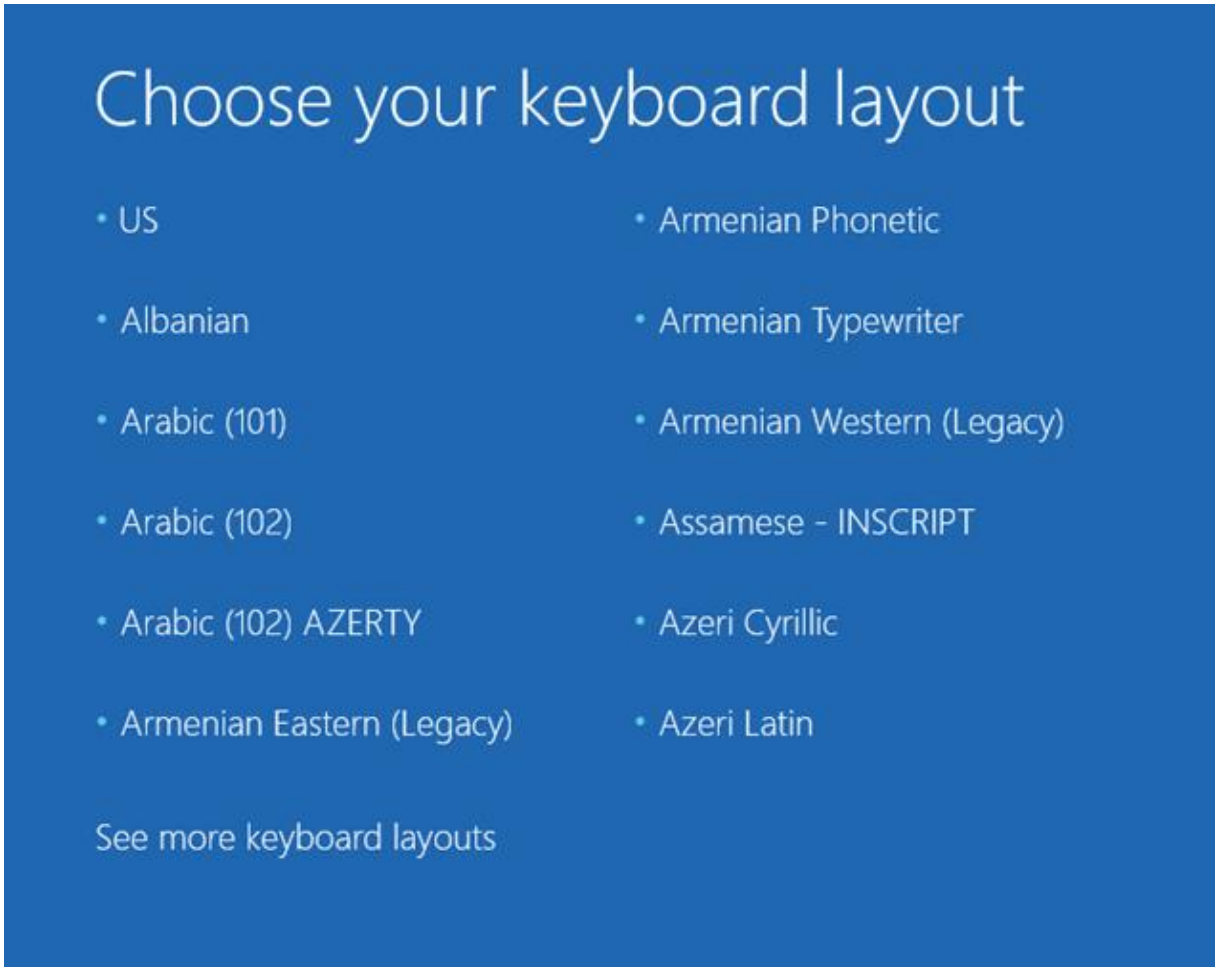
8 Troubleshoot

8.1 Preparing Automatic Repair

If the "Preparing Automatic Repair" screen appears when the board restarts, there is no need to worry. This message appears when the board has not been shut down properly (e.g., due to a crash). This problem can usually be resolved by restarting the board.

8.2 Choose your Language

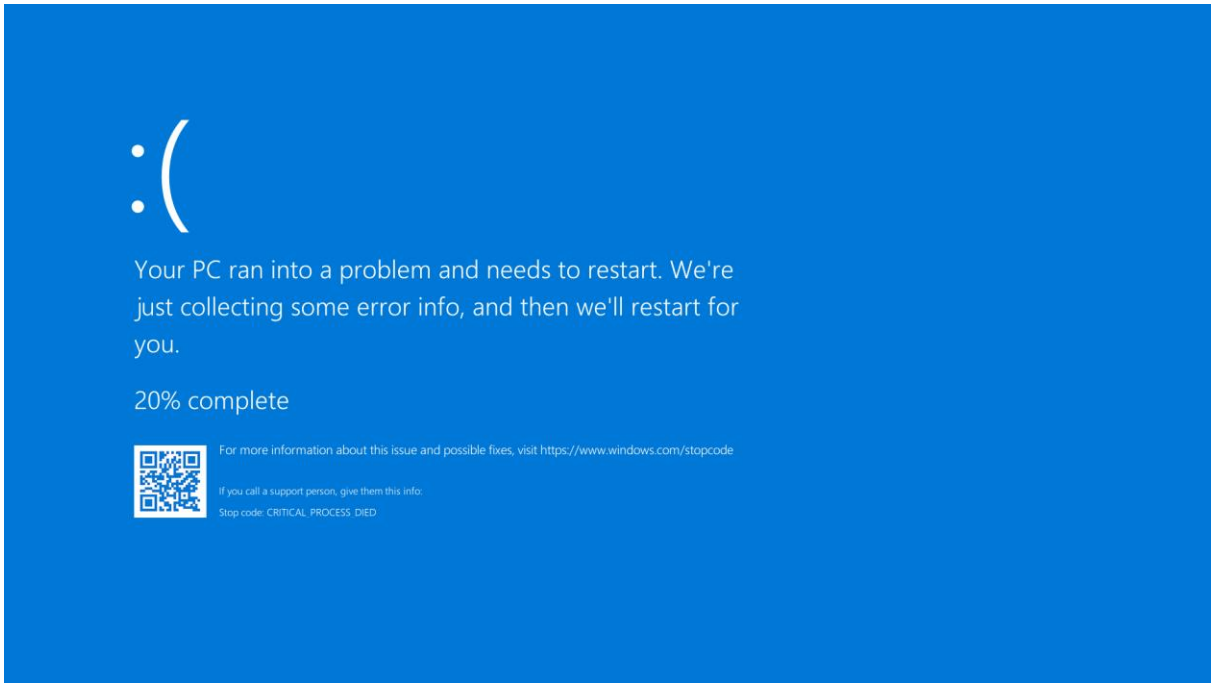
If, as described in section 8.1, the Automatic Repair screen appears and you do not restart the board, you will eventually reach the "Choose your Keyboard Layout" screen. At this point, USB devices will not work because no selection is needed. After restarting the board, it should start up as usual.



8.3 Blue Screen

A screen that no one likes to see but everyone has encountered: the good old Windows Blue Screen. No need to worry! According to the internet, 99% of Blue Screens can be attributed to drivers. Ideally, the Blue Screen will also display a message in the bottom left corner indicating which driver is involved.

Take note of this information and send us an email at support@fs-net.de. A restart should usually get the board up and running again.



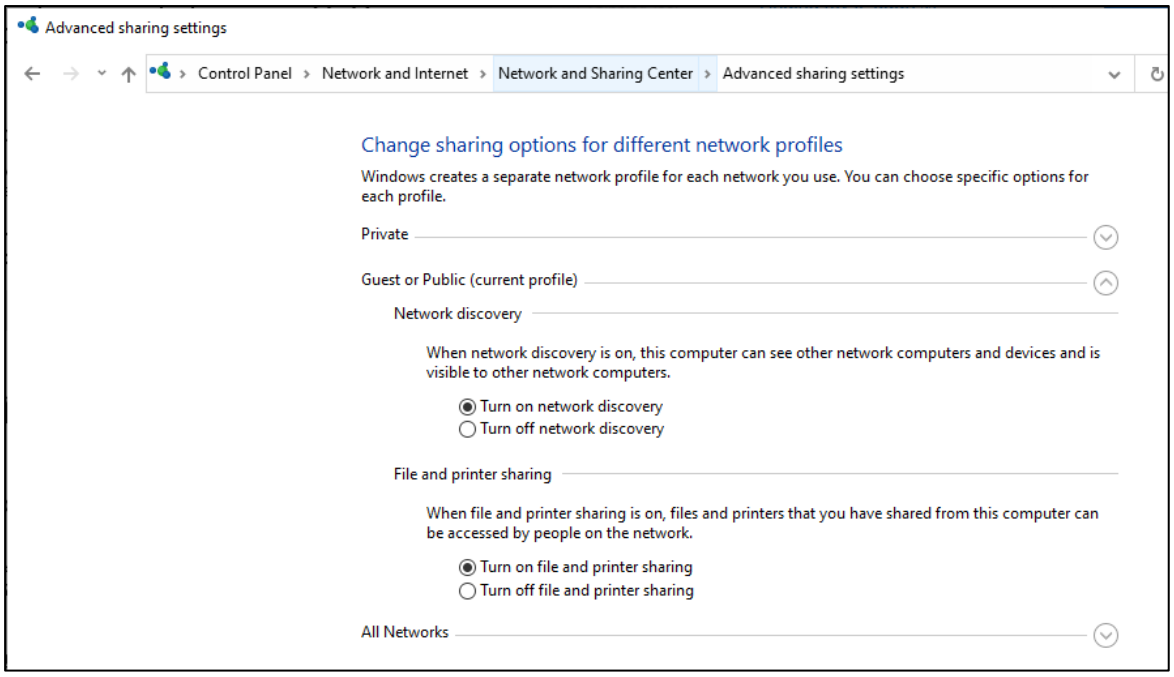
Since the latest Windows update in 2026, the familiar blue screen has been replaced. If a problem occurs now, the following screen is displayed:

9 Further information

Our internet forum is also a valuable source of information. Explore it for insights about Windows IoT, tools for many interfaces, development environments and more. If you have any questions or specific issues, please visit also our forum.

10 Appendix

List of Figures



Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. F&S Elektronik Systeme assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained in this documentation.

F&S Elektronik Systeme reserves the right to make changes in its products or product specifications or product documentation with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

F&S Elektronik Systeme makes no warranty or guarantee regarding the suitability of its products for any particular purpose, nor does F&S Elektronik Systeme assume any liability arising out of the documentation or use of any product and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

Products are not designed, intended, or authorised for use as components in systems intended for applications intended to support or sustain life, or for any other application in which the failure of the product from F&S Elektronik Systeme could create a situation where personal injury or death may occur. Should the Buyer purchase or use a F&S Elektronik Systeme product for any such unintended or unauthorised application, the Buyer shall indemnify and hold F&S Elektronik Systeme and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorised use, even if such claim alleges that F&S Elektronik Systeme was negligent regarding the design or manufacture of said product.