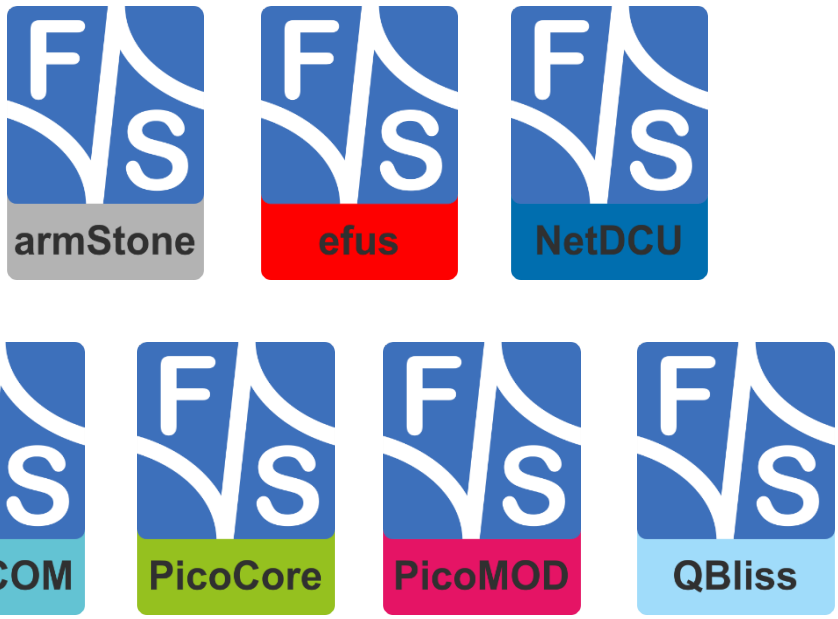
F&S i.MX93 Linux

First Steps

Version 1.1
(2024-09-9)



Elektronik Systeme

Untere Waldplätze 23
D-70569 Stuttgart
Germany

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

About This Document

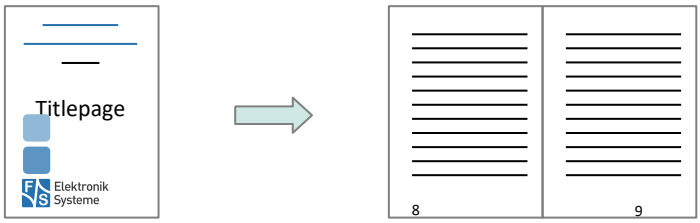
This document shows how to bring up F&S boards and modules under Linux, how to update firmware and how to use the system and the devices. It covers also compiling bootloader, Linux kernel and root filesystem as well as how to build your own applications for the device.

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
2024-03-19	1.0	fsimx93	A	ALL	Based on fsimx8ulp V1.0	AD
2024-09-09	1.1	fsimx93	A, M	2.2-2.4, 3.1-3.3, 4.11, 5.2.2	Add section Dockefile and changes because of new hardware revision	AD/ CS

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

Table of Contents

1 Introduction 1

1.1 F&S Board Families and CPU Architectures 1

1.2 Scope of This Document 2

2 Setting up the Board 2

2.1 PicoCoreMX93 - Starterkit 3

2.2 FS 93 OSM-SF – Starterkit 5

2.3 Serial Connection 6

2.4 Start Board 7

3 Software Installation 7

3.1 Build Release from Github 7

3.1.1 Download manifest repository 7

3.1.2 Dockerfile 7

3.1.3 Prepare build environment 8

3.1.4 Configure Yocto 8

3.1.5 Build Yocto 8

3.1.6 Deployed Images 9

3.2 Flash new Images 10

3.2.1 Flash via UUU 10

3.2.2 Flash via U-Boot 11

3.3 Enter UBoot 12

3.4 Set MAC Address 14

3.5 Restart Board 14

4 Using the Standard System and Devices 15

4.1 procfs 15

4.2 sysfs 16

4.3 USB Stick (Storage) 16

4.4 Qt6 17

4.5 Wayland/Weston 17

4.6 Ethernet 17

4.7 TFTP 17

4.8 SSH 17

4.9 Serial 18

4.10 I²C 19
4.11 GPIO 19
4.12 RTC 20
5 Next Steps 21
5.1 F&S Workshops 21
5.2 Further Information 21
5.2.1 Resources for PicoCoreMX93 22
5.2.2 Resources for FS 93 OSM-SF 22
6 Appendix 23
6.1 uuu.auto 23
6.2 List of Figures 24
6.3 List of Tables 24
6.4 Important Notice 25

1 Introduction

1.1 F&S Board Families and CPU Architectures

F&S offers a whole variety of Systems on Module (SOM) and Single Board Computers (SBC). For new projects there are different board families that are named armStone, efus, OSM, PicoCore, SMARC and SolderCore.

Family	Type	Size
armStone	Single Board Computer	100 mm x 72 mm (PicoITX)
efus	System on Module	62 mm x 47 mm
OSM	BGA System on Module SGeT	30 mm x 30 mm
PicoCore	System on Module	40 mm x 35 mm
SMARC	System on Module SGeT	82 mm x 50 mm
SolderCore	BGA System on Modul	35 mm x 35 mm

Table 1: F&S Board Families

Linux is available for all these platforms. F&S combines releases for platforms with the same CPU – or rather SoC (System on Chip) – as so-called *architecture releases*. All the boards of the same architecture can use the same sources, and the binaries can be used on any board of this architecture. Please note the difference: *board families* are grouped by form factor, *architectures* are grouped by CPU type, i.e. they usually contain boards of different families.

Table 2 shows all the architectures that are currently supported by F&S.

Architecture	CPU	Platforms
fsimx6	NXP i.MX6	efusA9, QBlissA9, QBlissA9r2, armStoneA9, armStoneA9r2, PicoMODA9, NetDCUA9
fsimx6sx	NXP i.MX6-SoloX	efusA9X, PicoCOMA9X, PicoCoreMX6SX
fsimx6ul	NXP i.MX6-UltraLite	efusA7UL, PicoCOM1.2, PicoCoreMX6UL
fsimx7ulp	NXP i.MX7ULP	PicoCore
fsimx8mm	NXP i.MX8MM	PicoCore, OSM
fsimx8mp	NXP i.MX8MP	armStone, efus, PicoCore, SMARC
fsimx8ulp	NXP i.MX8ULP	OSM, PicoCore, SolderCore
fsimx93	NXP i.MX93	PicoCore

Table 2: F&S Architectures

1.2 Scope of This Document

This document describes the *fsimx93* architecture. That means all F&S boards and modules based on the NXP i.MX93 SOC. The steps in this document will help you get to know your board and do some basic operations in Linux, so that you can try out all the periphery and do some first tests and comparisons.

The additional document `LinuxOnFSBoards_eng.pdf` explains the more generic ideas and concepts of Linux on F&S boards and modules. So, after having become acquainted with the board, you should continue reading that Linux document to get a more in-depth knowledge of the board and software.

2 Setting up the Board

In this chapter we will show how to connect the board to the PC. For a first test of the board functions, we only need a serial connection between PC and board. So as a first step, we will introduce all the boards and Starterkits of the *fsimx93* architecture and show the location of all connectors, especially the debug port.

2.1 PicoCoreMX93 - Starterkit

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

- Cables (ethernet, serial, power, USB, …).
- Software (source, binaries, install scripts, examples).
- Starterkit carrier board that offers connectivity for most interfaces available in PicoCoreMX93.
- PicoCoreMX93 module.

For basic operation please make sure that the Serial Debug Port and power are connected correctly.

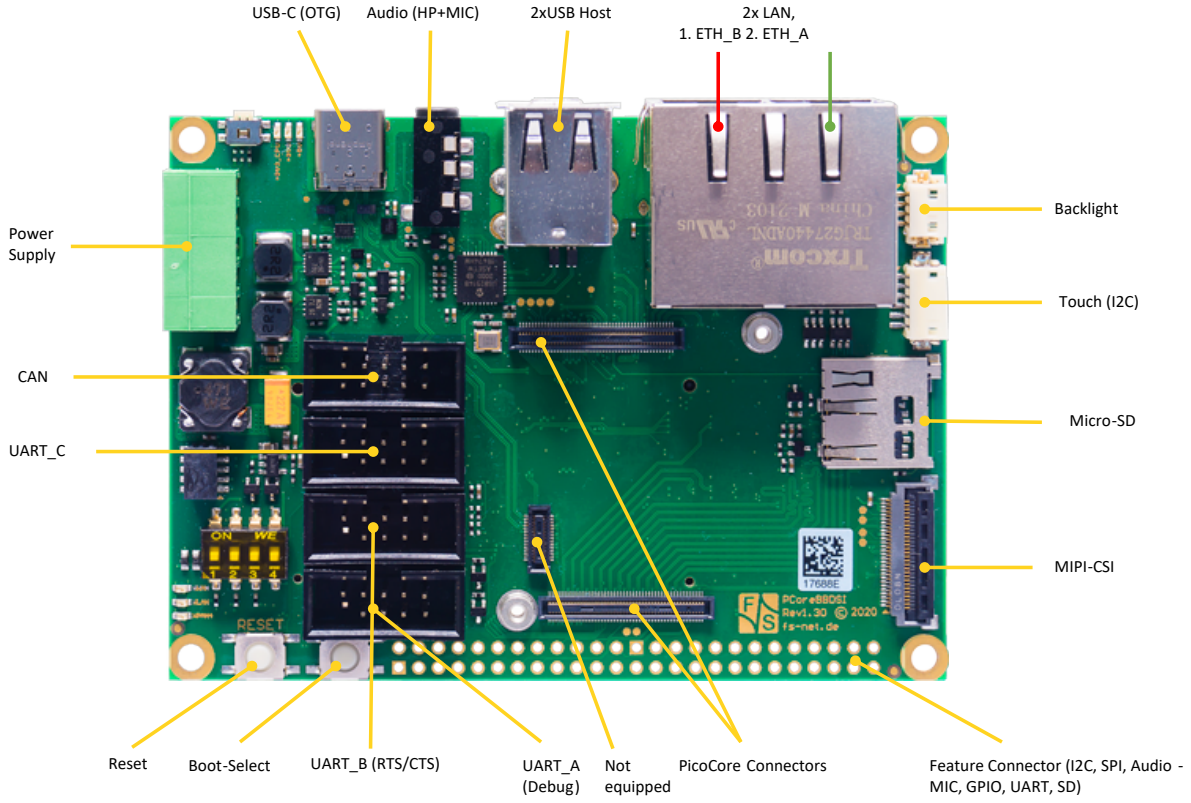


Figure 1: PicoCoreBBDSI Starterkit Top Side (PicCoreBBDSI – Rev. 1.30)

Figure 1 shows the connectors available on the top side of the PicoCoreMX93 SKIT carrier board.

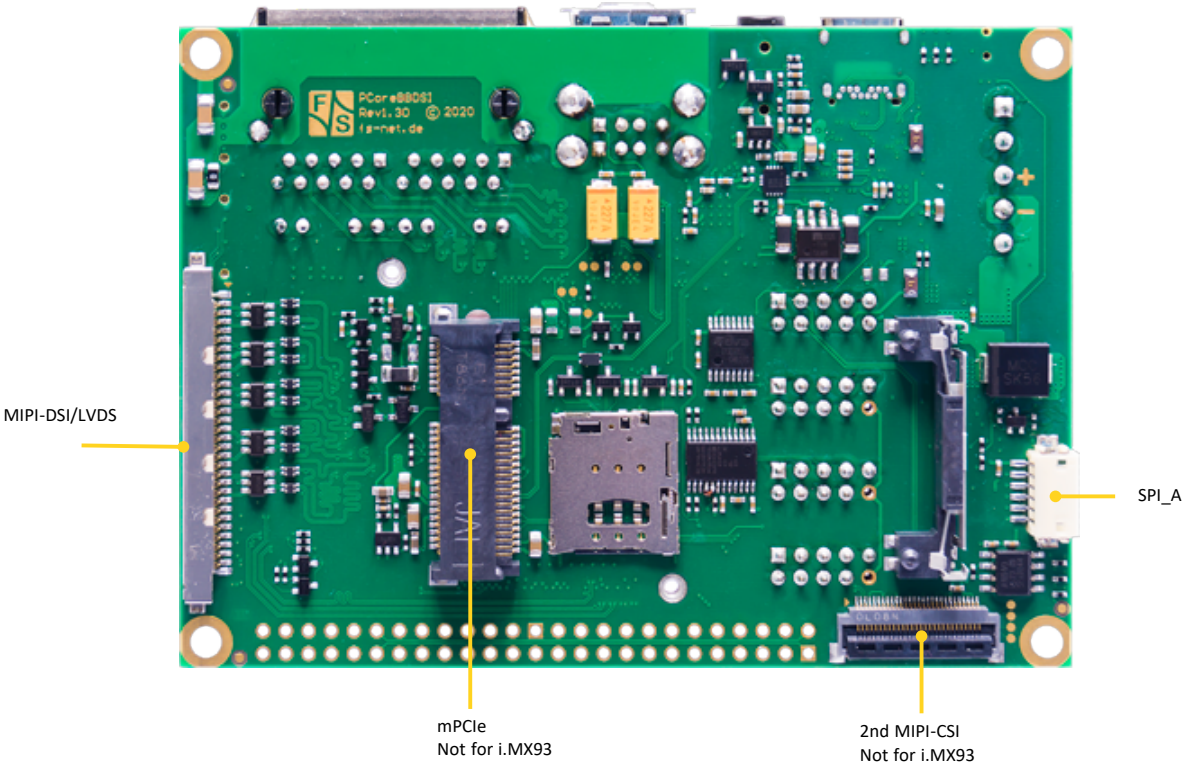


Figure 2: PicoCoreBBDSI Starterkit Bottom Side (PicCoreBBDSI – Rev. 1.30)

The connectors available from the bottom side of the PicoCoreMX93 SKIT can be seen in the Figure 2

Figure 3 shows the PicoCoreMX93 module.

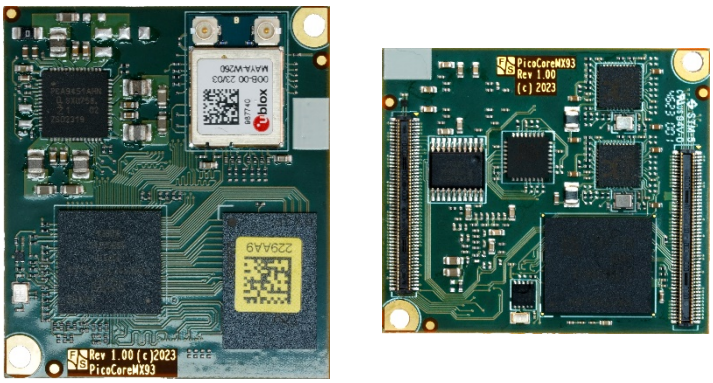


Figure 3: PicoCoreMX93 top and bottom view

2.2 FS 93 OSM-SF – Starterkit

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

- Cables (ethernet, serial, power, USB, …).
- Software (source, binaries, install scripts, examples).
- Starterkit carrier board that offers connectivity for most interfaces available in ADP-OSM-BB.
- FS-93-OSM-SF soldered on ADP-OSM-BB.
- ADP-OSM-BB is the carrier for all OSM-SF Modules.
  It provides compatible Interfaces to PicoCoreBBDSI.

For basic operation please make sure that the Serial Debug Port and power are connected correctly.

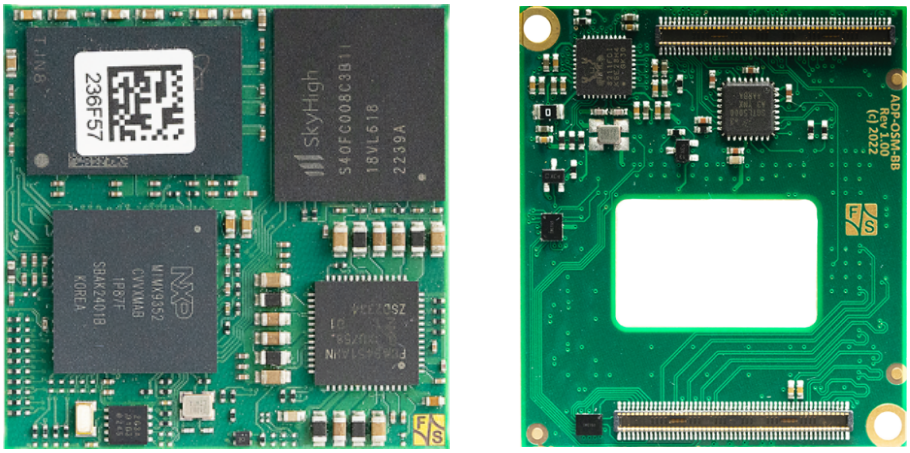


Figure 4: FS 93 OSM-SF and ADP-OSM-BB

2.3 Serial Connection

To work with the board, you need a serial connection with your PC. Use the provided Null- Modem cable and connect the debug port of the board (or Starterkit baseboard) with the serial port of a PC. Please refer to chapter 2.1 for the location of the COM ports. A serial port is mandatory on your PC, because we control the whole board via the serial port. If your PC does not provide a serial port, you must either use a USB-to-serial adapter or you need to install a PCIe extension card with a serial port.

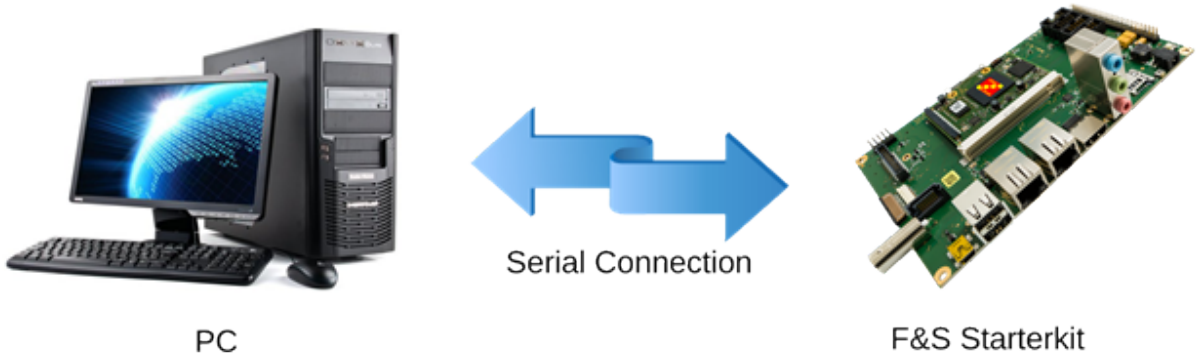


Figure 5: Serial connection from board to PC

For a first test, a Linux PC is not necessarily required. You can also use a Windows PC. But later for development, you definitely need a Linux PC, either native or as a Virtual Machine. With a Virtual Machine, you compile your software in Linux, but you can still have the serial connection done in Windows and use tools from Windows. This uses the best of both worlds.

On your PC, start a terminal program and open a serial connection to the board. Use 115200 baud, 1 start, 1 stop bit, no flow control. We recommend a terminal program that supports a 1:1 binary download and also supports ANSI Escape Sequences for color and text highlighting. Examples are:

- `TeraTerm` (Windows)
- `PuTTY` (Windows/Linux, does not support 1:1 download)

2.4 Start Board

Connect a power supply to the board. Please refer to section 2.1 for the location of the power supply pins. You need to supply +5V.

Now switch on the power supply. Quite immediately the terminal program should show boot messages from the booting Linux system. This will go on for a few seconds and then a login prompt should appear.

```
NXP i.MX Release Distro 6.1-mickledore fsimx93 ttyLP1

fsimx93 login:
```

Enter `root` to log in. In the default configuration, no password is required.

If everything went well, you could skip the next chapter and proceed with entering Linux commands.

3 Software Installation

When you get a Starterkit from F&S, the Linux system is usually pre-installed and boots to the Linux login prompt right away. In this case you can skip this chapter. But if you are switching over from a different operating system, if you are upgrading from a previous release, or if your board is empty for some other reason, the following sections describe how to install some standard software on your platform.

3.1 Build Release from Github

It is recommended to use the F&S Development Machine Fedora 40 to build a Yocto Image. F&S offers docker support for build current yocto BSP. Dockerfile for ubuntu 22.04 is added for the build process. The distribution is still supported for yocto scarthgap release.

Note:

Build process of bsp version fsimx93-Y2024.09-pre is not supported in fedora 40. Docker must be used by the customer.

3.1.1 Download manifest repository

```
git clone -b fsimx93-Y2024.09-pre \
    https://github.com/FSEmbedded/releases-fus.git
```

3.1.2 Dockerfile

```
git clone -b ubuntu-22.04 \
    https://github.com/FSEmbedded/docker-fus.git
```

Copy Dockerfile and run-docker.sh files to releases-fus directory.

3.1.3 Prepare build environment

Run run-docker.sh to build docker container and run it in interactive mode. Change to home directory and run setup-yocto to prepare your build environment. The script will read the repo manifest.xml file and syncs all repositories that are needed for Yocto.

```
cd release-fus
./setup-yocto <yocto-env>
```

< yocto-env > is the name of the directory where the repositories will be placed.

3.1.4 Configure Yocto

```
cd ~/<yocto-env>
MACHINE=fsimx93 DISTRO=<distro> source fus-setup-release.sh
```

Please note that this script needs to be sourced, which means it must be run in the current shell to be able to modify the existing environment. NXP uses the Yocto concept of distros to differentiate between the possible display solutions.

F&S handles this in a similar way, but has own layer, that define own distro names. However, they simply replace “fsl” with “fsimx93”. See following Table for a list of possible values.

<distro>	meaning
fus-imx-wayland	Use Wayland graphics
fus-imx-xwayland	Use Wayland with X11 support (no EGL on X11 applications)

Table 3: Yocto distros for fsimx93

3.1.5 Build Yocto

```
bitbake <image-name>
```

where <image-name> is one of the images listed below.

image name	target
core-image-minimal	A small image that is optimized for very small memory usage
imx-image-core	i.MX image with i.MX test applications to be used for Wayland backends.
imx-image-multimedia	i.MX image with a GUI without any Qt content.
imx-image-full	Builds an opensource Qt 6 image

Table 4: Yocto images for fsimx93

3.1.6 Deployed Images

Yocto creates a *"<image-name>.sysimg"* image which contains a partition table, boot-fs and root-fs partition. In addition, "imx-boot-fsimx93-pc.bin" contains the boot-firmware like U-Boot-SPL, ATF, U-Boot, M33 application, UPower Firmware and Secure Enclave Firmware.

You can find the images in:

```
~/<yocto-env>/yocto-fus/build-fsimx93-<distro>/
tmp/deploy/images/fsimx93/
```

3.2 Flash new Images

There are multiple ways to flash a new Image on a target board. The easiest way to flash a new release is while using the UUU-Tool (formally known as mfgtool). You can find the newest release on the following link:

https://github.com/nxp-imx/mfgtools/releases

Alternatively, the image can be written directly to the eMMC via U-Boot.

3.2.1 Flash via UUU

Place all needed files in a directory, where also uuu-tool is placed.

Connect the board via USB to the computer.

To load a new image, the board must be in fastboot mode. To access fastboot via UBoot, just run the following command:

```
=> fastboot 0
```

It is also possible to boot a board directly in serial download mode. Just set BMODE0=1.

To verify the connection with the computer run following in a shell:

```
./uuu -lsusb
```

The result should look like this:

```
Connected Known USB Devices
Path      Chip     Pro      Vid      Pid      BcdVersion
==========================================================
1:13              FB:      0x1FC9 0x0152    0x0221
```

When a valid connection is detected then run following command to flash new imx-boot-tagged and yocto image:

```
./uuu <uuu.auto>
```

<uuu.auto> is a script which is included in the appendix.

3.2.2 Flash via U-Boot

Write all Binary Files on a USB-Stick and plug the device into one of the USB ports of the Board. Enter U-Boot and run following commands:

```
=> usb start
=> load usb 0:1 $loadaddr imx-boot-fsimx93-pc.bin
=> mmc dev 0 1
=> mmc write $loadaddr 0 <bs-count>
=> load usb 0:1 $loadaddr <image-name>.wic
=> mmc dev 0
```

=> mmc write $loadaddr 0 <bs-count><bs-count> is the file size divided with the block-size of 512 Byte. Note that <bs-count> is a hexadecimal number.

3.3 Enter UBoot

The PicoCoreMX93 does not support NBoot so the first-level bootloader is UBoot. To enter UBoot it requires the serial setup as explained in Chapter 2.1. After that we can turn on the power and the following messages should appear.

```
U-Boot SPL 2024.04-F+S-fsimx93-Y2024.09-prev2024.04-fs0.1-pre (Sep
03 2024 - 11:33:09 +0000)
SOC: 0xa1009300
LC: 0x2040010
PMIC: Over Drive Voltage Mode
DDR: 3200MTS
M33 prepare ok
Normal Boot
Trying to boot from BOOTROM
Boot Stage: Primary boot
image offset 0x0, pagesize 0x200, ivt offset 0x0
Load image from 0x4d400 by ROM_API


U-Boot 2024.04-F+S-fsimx93-Y2024.09-prev2024.04-fs0.1-pre(Sep 03
2024 - 11:33:09 +0000)
eset Status: POR
CPU:   NXP i.MX93(52) Rev1.1 A55 700 MHz
CPU:   Industrial temperature grade (0C to 105C) at 63


Model: F&S PicreMX93
DRAM:  1 GiB
Core:  235 devics, 28 uclasses, dicetree: separate
                                                    MMC:    FSL_SDHC:
0, FSL_SDHC: 1
Loadingnvironment from MMC... OK
Fail to setup video link
I    serial
Out:   serial
Err   serial


BuildInfo:
  - ELE firmware version 1.1.0-17eafa6
```

```
switch to partitions #0, OK
mmc0(part 0) is current device
NBoot:
flash target is MMC:0
Net:   eth1: ethernet@490000, eth0: ethernet@428a0000 [PRIME]
Fastbo: Normal
NormalBoot
Hit any key to stop autoboot:  0
=>
```

You must hit any key within 3 seconds to enter UBoot.

3.4 Set MAC Address

When we erased the U-Boot environment including the MAC address for the Ethernet chip. We must set it again now and save it permanently.

The MAC address is a unique identifier for a network device. Each network device has its own address that should be unique across the whole world. So, each network port on each board needs a unique MAC address.

A MAC address consists of twelve hexadecimal digits (0 to 9 and A to F), that are often grouped in pairs and separated by colons. The first six digits for F&S boards are always the same: 00:05:51, which is the official MAC address code for the F&S company. The remaining six digits can be found on the bar-code sticker directly on your board (see Figure 3).

Figure 6: Barcode sticker

The full MAC address for this example would be 00:05:51:07:93:4B. If your board supports two ethernet ports, you need two MAC addresses. The second one is the first one plus 1, i.e. 00:05:51:07:93:4C.

The following two commands will set the MAC addresses and store the current environment (including the newly set MAC addresses) in NAND flash. Of course you have to replace `xx:yy:zz` with the six hex digits from the bar-code sticker on your board (and xx:yy:vv with the six hex digits plus 1).

```
setenv ethaddr 00:05:51:xx:yy:zz
saveenv
```

Warning

If you do not set this unique address, a random address is generated by U-Boot.

3.5 Restart Board

Installation is complete. To check if everything was done correctly, restart the board. You can either enter U-Boot command, …

```
reset
```

… or press the reset button, or simply switch the power off and on again. Like in chapter 0, the terminal program should show boot messages from the booting Linux system. This will go on for a few seconds and then a login prompt should appear.

```
fsimx93 login:
```

4 Using the Standard System and Devices

By default, the root filesystem is mounted read-write. For read-only root filesystems you cannot create files unless you go to a directory like /tmp that is located in a RAM disk. This is to make the system as stable as possible. If the root filesystem is mounted read-only, it is usually no problem to just switch off the power. So, you have to be more careful on Yocto.

If you want to remount the filesystem in read-write mode, just say

```
mount -o remount,rw /
```

Note the slash `/` that denotes the mount point of the root directory. Now you can create files everywhere. But remember that written data is often buffered in RAM first and is not immediately stored on the media itself. If you simply switch off the power now, data that was still buffered in RAM may be lost. So, in this case it is important to shut down the system with ...

```
halt
```

... or restart with:

```
reboot
```

Or you can remount the root filesystem in read-only mode again after applying the changes:

```
mount -o remount,ro /
```

All these commands will force the system to write any buffered data to the media.

The `/dev` directory is also built on top of a RAM disk. This allows the kernel to create and remove device entries dynamically. For example, if a USB stick is plugged in, a new device entry `/dev/sda1` is automatically created. And when the USB stick is unplugged, the device entry is also automatically removed again.

4.1 procfs

Linux has a virtual filesystem called Procfs. It is mounted under `/proc` and provides information about the system in general and about each process that is currently active.

Get information about the CPU

```
cat /proc/cpuinfo
```

Shows the Linux version.

```
cat /proc/version
```

Shows the current memory usage.

```
cat /proc/meminfo
```

Lists the supported filesystems.

```
cat /proc/filesystems
```

4.2 sysfs

Sysfs is another virtual file system in Linux. It exports information about devices and drivers from the kernel device model to user space. Which means you can get information about current device settings and some drivers even allow configuring the device at runtime.

Devices that want to share information or want to accept configuration settings, create subdirectories under the `/sys` directory. Then virtual text files are used to pass the information. So, for example if a touch panel can accept some sensitivity configuration, it would create a file `sensitivity` there. By reading data from the file, you could query the current setting. And by writing a new value to the file, you could set a new sensitivity value.

For example, access the RTC subsystem:

```
cat /sys/class/rtc/rtc0/date
```

Show the CPU core temperature (in 1/1000 °C):

```
cat /sys/class/thermal/thermal_zone0/temp
```

4.3 USB Stick (Storage)

If a USB memory stick is inserted, it is available like a standard hard disk. Because there is usually no real hard disk connected, it is found as `/dev/sda`. If you have partitions on your USB stick, you must access them as `/dev/sda1`, `/dev/sda2` and so on.

You can mount and unmount all the partitions now. For example, to mount the first partition on directory `/mnt`, you must issue the following command:

```
mount /dev/sda1 /mnt
```

To unmount again, issue:

```
umount /mnt
```

Remarks

If a USB storage device contains more than one partition, a device entry will be created for each partition, for example also `/dev/sda2` and `/dev/sda3`. In fact, also a device entry `/dev/sda` without a number is available, that refers to the whole device, including all partitions. Some USB storage devices do not contain a partition table at all. Then only `/dev/sda` is created.

If more than one storage device is plugged in (for example via a USB hub), then the second device has the base name `sdb`, the third `sdc`, and so on.

4.4 Qt6

Qt is a cross-platform application framework that is widely used for developing applications, often with a graphical user interface (GUI). You can build Qt libraries with imx-image-full.

4.5 Wayland/Weston

Wayland is a GUI protocol that replaced X11. Wayland itself only manages the protocol and the communication to the clients. It needs an additional component, the so-called compositor. This compositor works like a Window Manager and presents the graphical output on the display.

The Yocto release uses weston as wayland compositor.

4.6 Ethernet

To activate the Ethernet port in Linux, you have to configure the network device first. For example, to use IP-Address 10.0.0.242, you can use the command

```
ip addr add 10.0.0.242/24 dev eth0
```

Then you can use network commands, e.g.

```
ping 10.0.0.121
```

There is also a DHCP client included. To receive an IP address via DHCP just call:

```
udhcpc
```

If your board supports more than one network interface, you can add option `-i` to specify the appropriate interface. For example, to request IP data for interface `eth1`:

```
udhcpc -i eth1
```

4.7 TFTP

There is a small program to download a file from a TFTP server. This can be rather useful to get some files to the board without having to use an SD card or a USB stick. For example, to load a file `song3.wav` from the TFTP server with IP address 10.0.0.121, just call

```
tftp -g -r song3.wav 10.0.0.121
```

4.8 SSH

Ls /In Yocto ssh is configured to allow root login and *empty-password* by default. Just connect via SSH by any host in the network with:

```
ssh <username>@<device-ip>
```

You can change these settings at:

```
vi /etc/ssh/sshd_config
```

4.9 Serial

On NXP CPUs, the devices are called `/dev/ttyLP<n>`, where `<n>` is a number starting with 0. One port is usually used as serial debug port where all console messages are sent to. This one port runs at 115200 bit/s. All other ports are at 9600 bit/s by default. Use the `stty` program to change this.

To access a serial port from the command line, you can use input and output redirection.

Show a string on port `ttyLP2`:

```
echo Hello > /dev/ttyLP2
```

Show characters that arrive on port `ttyLP2`:

```
cat < /dev/ttyLP2
```

Usually character input is line buffered, so you will only see information after sending Return.

Remark

The default setting for serial ports in Linux echo each character that arrives. So, if you connect one serial port to a second serial port with a Null-Modem cable, sending a single character will result in an endless loop. Each side will echo the character indefinitely. You can use `stty` to change this behavior.

4.10 I²C

Most devices on an I²C bus are accessed by a device driver in Linux. But you can also have access to additional devices from user space software. There are even command line tools to do this. The following examples are from a PicoCoreMX93.

Show the available I²C buses:

```
i2cdetect -l
i2c-0   i2c             44340000.i2c                    I2C adapter
i2c-1   i2c             44350000.i2c                    I2C adapter
i2c-4   i2c             426b0000.i2c                    I2C adapter
i2c-7   i2c             426e0000.i2c                    I2C adapter
```

Show the available devices on bus `i2c-0`:

```
i2cdetect -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: UU -- -- -- -- -- -- -- 58 -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

So, there are no devices on `i2c-0`. Some devices are handled by a Linux driver, so they cannot be accessed from user space (therefore marked as “used” with `UU`). Now you can read and write data with `i2cget` and `i2cset` or read whole data areas with `i2cdump`.

4.11 GPIO

You can setup and use GPIOs with the libgpiod and provided tools.

```
$ gpiodetect
```

```
gpiochip0 [43810000.gpio] (32 lines)
gpiochip1 [43820000.gpio] (32 lines)
gpiochip2 [43830000.gpio] (32 lines)
gpiochip3 [47400000.gpio] (32 lines)
gpiochip4 [1-0020] (16 lines)
```

Example

On PicoCoreMX93, use pin GPIO_IO19 (GPIO_J1_52, TOUCH_RESET) as output pin.

```
#Set Output to 1
gpioset -c gpiochip0 19=1
```

Now the pin should have a high level (about 3.3V or 1V8) which you can measure with a voltmeter.
To set pin low again type:

```
# Set Output to 0
gpioset -c gpiochip0 19=0
```

To set the Pin as input, just run:

```
gpioget -c gpiochip0 19
```

4.12 RTC

Setting date:

```
date "2015-11-29 22:55"
```

Save current date to RTC:

```
hwclock -w
```

The time will automatically be loaded from the RTC at the next boot.

Note

PicoCoreMX93 has an internal RTC. Make sure VBAT is connected to the module. Otherwise, the RTC cannot keep time.

5 Next Steps

This document only showed a very basic usage of the board and the Linux system. The next logical step is the generic Linux documentation `LinuxOnFSBoards_eng.pdf`. It will show you the ideas and concepts behind the F&S Linux environment and how you can work efficiently with these boards.

5.1 F&S Workshops

F&S also offers several workshops. Especially if you are new to working with embedded boards or even new to Linux, we recommend visiting the workshop “Linux on F&S Modules”. Working with an embedded system is quite different to working with a desktop Linux. This workshop will show you a basic introduction to Linux, how to use NBoot, U-Boot and Linux on an F&S board, how to compile the system software, how to download files to the board, and how to write your own programs. The workshop lasts four hours and takes place in Stuttgart at the F&S company building. It may save you many hours of reading, trying, and even frustration.

Additional workshops are available for working with Buildroot, Asymmetric Multiprocessing, Secure Boot, Working with GIT. Please look at our website for any additional offers.

5.2 Further Information

Many additional resources of information are available on the F&S website.

Document	Description
AdvicesForLinuxOnPC.pdf	Explains how to install server software and tools on a Linux development PC that is used with F&S Linux boards.
PicoCoreBBDSI_eng.pdf	Hardware documentation: there are separate documents for each board and for the Starterkit baseboards. F&S also offers Eagle layout files for some of our Starterkits.
LinuxOnFSBoards.pdf	Shows how to use the bootloaders, Linux system and peripherals on F&S boards and modules

Table 5: Important documents, available on the F&S website

Important documents, available on the F&S website

We do not include all these documents in the release to make sure that you always get the newest version when you start. The following sections give direct links to important places like documentation and add-ons.

A good source for information is also our internet forum. If you have any questions or specific problems, please feel free to go to: https://forum.fs-net.de/.

5.2.1 Resources for PicoCoreMX93

Hardware documentation for PicoCoreMX93 module itself, Starterkit baseboard, including schematics:

https://www.fs-net.de/de/embedded-module/computer-on-module-picocore/picocoremx93-mit-nxp-imx93-cpu/#panel-6

Available accessories, adapters, and extensions:

https://www.fs-net.de/en/embedded-modules/computer-on-module-picocore/picocoremx93-with-nxp-imx93-cpu/#panel-4

5.2.2 Resources for FS 93 OSM-SF

Hardware documentation for PicoCoreMX8ULP SoM, Starterkit baseboard, including schematics:

https://www.fs-net.de/de/embedded-module/open-standard-module-osm/fs-osm-sf-mx93/#panel-6

Available accessories, adapters, and extensions:

https://www.fs-net.de/de/embedded-module/open-standard-module-osm/fs-osm-sf-mx93/#panel-4

6 Appendix

6.1 uuu.auto

```
uuu_version 1.2.39

# This command will be run when ROM support stream mode
# i.MX8QXP, i.MX8QM
SDPS: boot -f flash-<osm/pc>.bin

FB: ucmd setenv fastboot_dev mmc
FB: ucmd setenv emmc_dev 0
FB: ucmd setenv mmcdev ${emmc_dev}
FB: ucmd mmc dev ${emmc_dev}
FB: flash bootloader flash-<osm/pc>.bin

FB: flash -raw2sparse all fus-image-std.sysimg

FB: ucmd if env exists emmc_ack; then ; else setenv emmc_ack 0;
fi;
FB: ucmd mmc partconf ${emmc_dev} ${emmc_ack} 1 0

FB: done
```

6.2 List of Figures

Figure 1: PicoCoreBBDSI Starterkit Top Side (PicCoreBBDSI – Rev. 1.30) 3
Figure 2: PicoCoreBBDSI Starterkit Bottom Side (PicCoreBBDSI – Rev. 1.30) 4
Figure 3: PicoCoreMX93 top and bottom view 4
Figure 4: Serial connection from board to PC 6
Figure 5: Barcode sticker 14

6.3 List of Tables

Table 1: F&S Board Families 1
Table 2: F&S Architectures 2
Table 3: Yocto distros for fsimx93 8
Table 4: Yocto images for fsimx93 8
Table 5: Important documents, available on the F&S website 21

6.4 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.