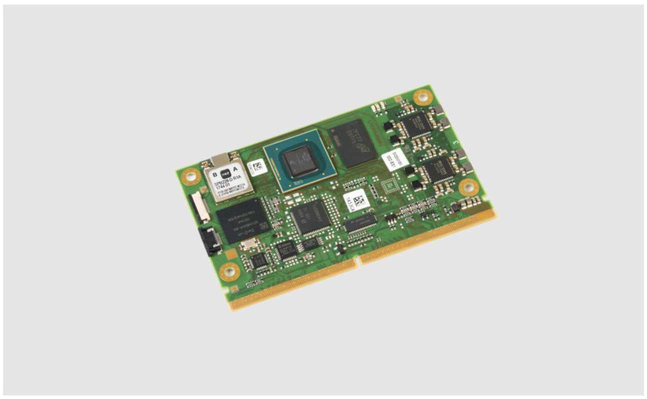
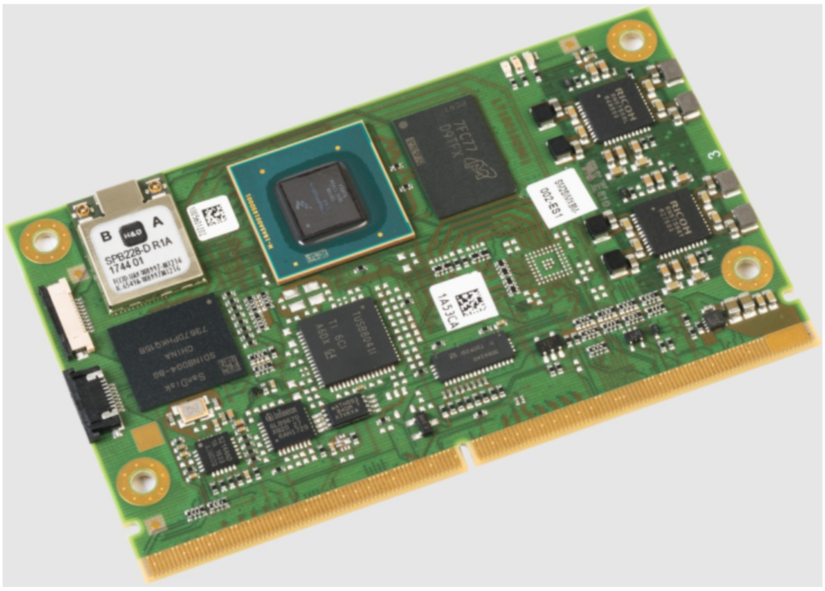


AVNET EMBEDDED MSC SM2S-IMX8M Debug UART Port ARM Based Computers on Module Instructions

AVNET EMBEDDED MSC SM2S-IMX8M Debug UART Port ARM Based Computers on Module Instructions



Contents

Preface

General Information

Scope

This document applies for all Avnet Embedded Computer-on-Modules based on NXP i.MX8- and i.MX9-series CPUs, such as e.g.

- SM2S-IMX8PLUS
- SM2S-IMX8M
- SM2S-IMX8MINI
- SM2S-IMX8NANO
- SM2S-IMX8 (QuadPlus/QuadMax)
- SM2S-IMX93
- OSM-SF-IMX91
- OSM-SF-IMX93
- OSM-MF-IMX8NANO
- OSM-MF-IMX8MINI

This list does not claim to be exhaustive, especially as new boards may be released for which the same procedures may apply, while this document may not always be updated immediately.

Revisions and Modifications

Revision	Date		Comment
1.0	25.05.2023	M. Koch	Initial version

How to change debug UART port

Introduction

Many ARM-based systems come with a Serial Console as the main means of access for debug and system bring-up purposes. Depending on customer requirements, it is a frequently occurring necessity in such systems that the default debug UART port needs to be switched to a different serial port. By debug UART port we mean the UART port which will provide all U-Boot input/output, kernel bootlog output and kernel shell, basically all UART communication one will see in a minimal Yocto image. Since the default UART port is not used by only one software component, but rather by many components, including u-boot, atf-firmware, optee-os and kernel itself, setting the default UART port can be challenging, and the overview can quickly be lost. This document will explain all steps to follow, to achieve this task. For easier explanation, this document was written for NXP i.MX8 MINI processor and mscldk, but can be applied to all i.MX8- and i.MX9-series processors and other build systems with very little effort.

Preparing environment

Changing the default debug UART port will require modifying code in Yocto, and the easiest way to do it, is to use Yocto devtool to prepare the sources for us. It will be necessary to modify the following Yocto packages:

- u-boot-imx (virtual/bootloader)
- linux-imx (virtual/kernel)
- atf-imx
- optee-os (only if optee is used)

Sources should be prepared with devtool:

```
$ ./devtool modify u-boot-imx
$ ./devtool modify linux-imx
$ ./devtool modify atf-imx
$ ./devtool modify optee-os
```

All sources can be found in the “workspace” directory.

Modifying code

Modifying the Bootloader

In the bootloader some basic UART initialization will happen, so it will be necessary to modify muxing and the base address of the UART port. The second task of u-boot is to pass boot arguments to kernel and here it will be necessary to modify the console tty argument. UART initialization and muxing happens in the earliest stage of boot process in the SPL. The source code can be found in the board specific spl.c file.

Target file: workspace/sources/u-boot-imx/board/msc/sm2s_imx8mm/spl.c

Open the file and navigate to the function init_ser0():

```
static void init_ser0(void)
{
imx_iomux_v3_setup_multiple_pads(ser0_pads, ARRAY_SIZE(ser0_pads)); init_uart_clk(1);
}
```

The function activates the clock for **UART2(index** 1 for physical UART 2).

Now, if we want to use UART1 instead, we could define our own init_ser1 function:

```
static void init_ser1(void)
{
imx_iomux_v3_setup_multiple_pads(ser1_pads, ARRAY_SIZE(ser1_pads)); init_uart_clk(0);
}
```

Replace the function call of init_ser0 with init_ser1() in the board_early_init_f() function. Further we notice that ser1_pads structure is not defined. Here it will be necessary to be aware of the wiring of UART1. On imx8mm, UART1 can be muxed to uart1 pads, or to sai2 pads. Accordingly, ser1_pads might be defined as:

```
static iomux_v3_cfg_t const ser1_pads[] = {
IMX8MM_PAD_UART1_RXD_UART1_RX | DEFAULT_UART_PAD_CTRL,
IMX8MM_PAD_UART1_TXD_UART1_TX | DEFAULT_UART_PAD_CTRL, NULL
};
```

Or else, using sai2:

```
static iomux_v3_cfg_t const ser1_pads[] = {
IMX8MM_PAD_SAI2_RXFS_UART1_TX | DEFAULT_UART_PAD_CTRL, IMX8MM_PAD_SAI2_RXC_UART1_RX
| DEFAULT_UART_PAD_CTRL NULL };
```

Now the UART base address should be modified, the address is defined in boards header config file.

Target file: workspace/sources/u-boot-imx/include/configs/msc_sm2s_imx8mm.h

Modify the definition CONFIG_MXC_UART_BASE. For UART1 this would be:

- // #define CONFIG_MXC_UART_BASE
- UART2_BASE_ADDR
- #define CONFIG_MXC_UART_BASE
- UART1_BASE_ADDR

Finally, the console kernel argument should be modified. The value can be found in the same header file. Search for “console=ttymxc1…” and modify “ttymxc1” to “ttymxc0”. The index number correlates to UART index, and is always UART index minus 1. So for UART 2 we use ttymxc1, for UART3 use ttymxc2 etc..

Modifying the ARM Trusted Firmware

Arm Trusted Firmware(imx-atf) has no own UART initialization routine, however it has a hardcoded UART base address, and relies on proper UART initialization from u-boot. A different base address configuration in u-boot and imx-atf will most likely leave the processor stuck in an exception handler, and there will be nothing visible on UART (the CPU seems to hang for no visible reason). When the UART port is changed, this change must also happen in imx-atf! Changing the UART port in imx-atf requires a base address modification. This value is set in the platform.mk file of the processor.

Target file: workspace/sources/imx-atf/plat/imx/imx8m/imx8mm/platform.mk

The correct UART base address can be found in the imx8 reference manual. In this example we change from UART2 to UART1 on imx8mm:

```
# IMX_BOOT_UART_BASE ?= 0x30890000
IMX_BOOT_UART_BASE ?= 0x30860000
```

Modifying optee-os

Optee OS is usually loaded when the CAAM module cryptography is used on the system. Optee runs on the same ARM Cortex-A53 cores, but in another completely independent instance as the kernel. Optee also requires UART access and in this case it means a base address modification.

Target file: workspace/sources/optee-os/core/arch/arm/plat-imx/conf.mk

This example will set UART base address from UART2 to UART1:

```
#CFG_UART_BASE ?= UART2_BASE
CFG_UART_BASE ?= UART1_BASE
```

Modifying the Kernel

Kernel will require only few device tree modifications, and only if the UART is not available in kernel yet. In many cases, no modification will be required at all. However, please verify if the tty instance you want to use exists! Check the tty presence with a shell command:

```
$ ls /dev/ttymxc*
```

If the ttymxc is already available, no modification is needed. For example, for UART2 this would be /dev/ttymxc1. As we already know, the index of tty is always the index of the physical UART minus 1. If the required tty is not available, please follow the usual UART integration procedure.

Build & Test

For a safe rebuild a full cleanup should be applied:

```
$ ./bitbake –c cleanall u-boot-imx linux-imx imx-atf optee-os
```

Build the image again with the default build command, for example:

```
$ ./bitbake msc-image-base
```

For testing purpose, a UART adapter should be connected to the old UART. There should be no more output on the old UART! The new UART should be fully functional, for that please verify the connection by using the u-boot UART shell and the linux console.

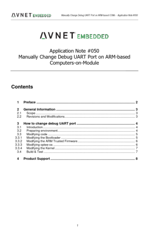
Product Support

Avnet Embedded engineers and technicians are committed to provide support to our customers whenever needed. Before contacting Technical Support of Avnet Embedded, please consult the respective pages on our website at
https://embedded.avnet.com/support/
for the latest documentation, drivers and software downloads.

If the information provided there does not solve your problem, please contact our Avnet Embedded Technical Support team as follows:

Email: support.boards@avnet.eu
Phone: +49 (0)8165 906-200

Documents / Resources

 Manually Change Debug UART Port on ARM-based COMs - Application Note #050 Application Note #050 Manually Change Debug UART Port on ARM-based Computers-on-Module **Contents** 1 **Preface** ..... 2 2 **General Information** ..... 3 2.1 Scope ..... 3 2.2 Revisions and Modifications ..... 3 3 **How to change debug UART port** ..... 4 3.1 Introduction ..... 4 3.2 Preparing environment ..... 4 3.3 Modifying code ..... 5 3.3.1 Modifying the Bootloader ..... 5 3.3.2 Modifying the ARM Trusted Firmware ..... 6 3.3.3 Modifying optee-os ..... 6 3.3.4 Modifying the Kernel ..... 7 3.4 Build & Test ..... 7 4 **Product Support** 8 1	**AVNET EMBEDDED MSC SM2S-IMX8M Debug UART Port ARM Based Computers on Module** [pdf] Instructions MSC SM2S-IMX8M, MSC SM2S-IMX8M Debug UART Port ARM Based Computers on Module, Debug UART Port ARM Based Computers on Module, UART Port ARM Based Computers on Module, Port ARM Based Computers on Module, ARM Based Computers on Mo dule, Based Computers on Module, Computers on Module, Module

References

- **Дома - platform.mk**
- **Support - Avnet Embedded**
- **User Manual**