AN14205

i.MX 9 Family – Boot-Time Measurement Methodology

Rev. 1 — 31 January 2024 Application note

Document information

Information	Content
Keywords	AN14205, i.MX 9, GPIO, boot-time measurement
Abstract	This document presents an approach for measuring the boot time on the i.MX 9 platforms using the GPIO pins.

1 Introduction

This document presents an approach for measuring the boot time on the i.MX 9 platforms using the GPIO pins.

The main objectives of this document are:

- To modify the bootloader and the system image for measurement
- To set up the board and the external logic analyzer tool

1.1 Software environment

Linux BSP release 6.1.36_2.1.0 is used to perform the measurements. The `imx-image-full` Yocto image is used during the experiments. An Ubuntu PC is assumed.

1.2 Hardware setup and equipment

- Development kit: NXP i.MX 93 11x11 EVK LPDDR4
- Micro-SD card: SanDisk Ultra 32 GB micro SDHC I Class 10 is used for the current experiment.
- USB-Type C cable for the debug port.
- A logic analyzer with the following minimum requirements can be used to measure the time:
  – 4 channels
  – 10 MS/s

2 General description

This section describes the general procedure that must be performed to obtain a baseline measurement for a clean system (with no startup optimizations).

2.1 Choosing the GPIO pin for measurement

A general-purpose pin is used to generate a pulse signal at different booting phases. Ideally, the desired GPIO pin is chosen from the pins that are neither used in the bootloader nor used in the Linux. To determine, check the associated device tree.

However, if it is not possible, the peripheral module that uses the pin must be disabled in the next step. Furthermore, it is highly recommended to choose a pin from the expansion connector on the board.

2.2 Updating the device tree at the bootloader and Linux level

After choosing the desired pin, some modifications have to be made at the device-tree level.

First, you have to check the desired pin functionality to find out the macros associated with the GPIO functionality. The header files can be found in different locations, depending on the board used.

Second, if a pin is used by another peripheral module, you must disable the respective module. It is done by setting the status property as “disabled” in the configuration information for that peripheral module.

Third, the adequate pin muxing is defined in the "pinctrl_hog" section using the GPIO macro for the chosen pin and the pad configuration values (`SW_PAD_CTL_PAD_*`).

2.3 Adding the first pulse generator

The first measured period is between the board POR and the execution of the `board_init_f` function of the SPL part of the bootloader. To generate a pulse, the pin must be configured as the output. After this, the pin can be driven high for a short time and then driven low. It can be done without a delay, because only the rising edge of the pulse is necessary.

2.4 Adding the second pulse generator

The second measured period is between the execution of the `board_init_f` function of the SPL part of the bootloader and before the kernel image loading from the U-Boot console. The toggling can be done here using the U-Boot GPIO commands, which can be written in the board configuration file located in the `include/configs/` directory or directly when booting the U-Boot, in its menu.

2.5 Adding the third pulse generator

The third measured period is between the kernel image loading from the U-Boot console moment to the first process that starts during the kernel boot (`systemd`). The toggling is similar to the case of generating the first pulse.

2.6 Measuring the total time with the logic analyzer

After building and flashing both the bootloader and the Linux image to the board, the measurement stage can begin.

The boot time is measured by starting the recording mode on the logic analyzer software and applying the reset button on the board. The recording is stopped after the third rising edge on the chosen GPIO pin. The elapsed boot time is the time between the rising edge of the nRST signal and the third rising edge of the GPIO pin.

3 Examples

This section describes the examples.

3.1 i.MX 93

This section describes the examples for i.MX 93.

3.1.1 Choosing the pins

The chosen pin is the 29th pin on the J1001 expansion connector on the specified board. In the schematics for the baseboard, the pin is `EXP_GPIO_IO05`. When searching the specified pin in the reference manual, check the functionality of the `GPIO2_IO05` pad that is used.

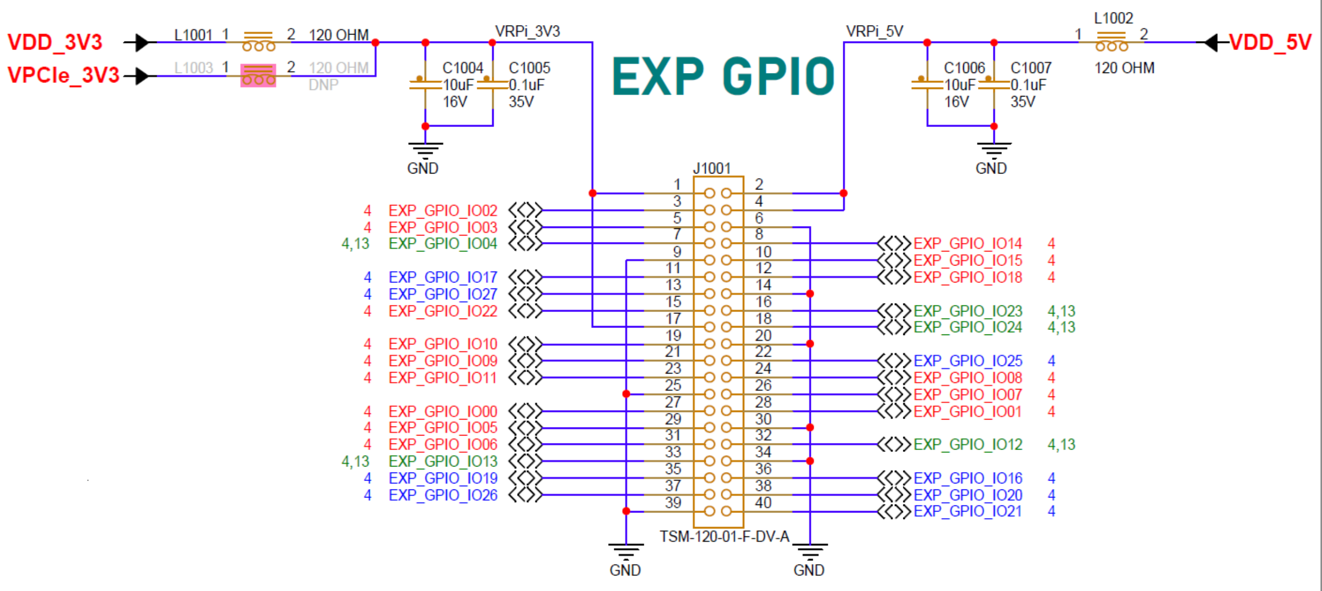


Figure 1. Schematic view of the expansion connector for the i.MX 93 EVK board

Table 1. Offset

Register	Offset
SW_MUX_CTL_PAD_GPIO_IO05	24h

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R / W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R / W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2. Diagram

Table 2. Fields

Field	Description
31-5 —	— Reserved
4 SION	Software Input On Field. Force the selected mux mode input path, no matter the `MUX_MODE` functionality. 0 - The input path is determined by the functionality. 1 - Force the input path of the `GPIO_IO05` pad.
3 —	— Reserved
2-0	MUX Mode Select Field. Select one of the eight `iomux` modes to be used for the `GPIO_IO05` pad. 000 - Select the mux mode: ALT0 mux port: GPIO2_IO05 of instance: gpio2.

Table 2. Fields*...continued*

Field	Description
MUX_MODE	001 - Select the mux mode: ALT1 mux port: TPM4_CH0 of instance: tpm4. 010 - Select the mux mode: ALT2 mux port: PDM_BIT_STREAM00 of instance: pdm. 011 - Select the mux mode: ALT3 mux port: MEDIAMIX_DISP_DATA01 of instance: mediamix. 100 - Select the mux mode: ALT4 mux port: LPSPI7_SIN of instance: lpspi7. 101 - Select the mux mode: ALT5 mux port: LPUART6_RX of instance: lpuart6. 110 - Select the mux mode: ALT6 mux port: LPI2C6_SCL of instance: lpi2c6. 111 - Select the mux mode: ALT7 mux port: FLEXIO1_FLEXIO05 of instance: flexio1.

3.1.2 Updating the U-Boot device tree

To get the necessary files, the U-Boot sources must be downloaded:

```
$ git clone https://github.com/nxp-imx/uboot-imx
$ cd uboot-imx
$ git checkout lf-6.1.36-2.1.0
```

To update the device tree, the macro associated to the desired functionality must be identified. This information resides in the `arch/arm/dts/imx93-pinfunc.h` file. The macro that must be used in this context is `MX93_PAD_GPIO_IO05__GPIO2_IO05`. However, the usage of this macro is not enough for an adequate pin mux setup, because it requires a sixth value, which represents the pad configuration. It is added in the associated device tree.

```
#define MX93_PAD_GPIO_IO05__GPIO2_IO05             0x0024 0x01D4 0x0000 0x0 0x0
#define MX93_PAD_GPIO_IO05__TPM4_CH0               0x0024 0x01D4 0x0000 0x1 0x0
#define MX93_PAD_GPIO_IO05__PDM_BIT_STREAM00       0x0024 0x01D4 0x0438 0x2 0x0
#define MX93_PAD_GPIO_IO05__MEDIAMIX_DISP_DATA01   0x0024 0x01D4 0x0000 0x3 0x0
#define MX93_PAD_GPIO_IO05__LPSPI7_SIN             0x0024 0x01D4 0x0000 0x4 0x0
#define MX93_PAD_GPIO_IO05__LPUART6_RX             0x0024 0x01D4 0x0000 0x5 0x0
#define MX93_PAD_GPIO_IO05__LPI2C6_SCL             0x0024 0x01D4 0x03F0 0x16 0x1
#define MX93_PAD_GPIO_IO05__FLEXIO1_FLEXIO05       0x0024 0x01D4 0x0380 0x7 0x0
```

Looking at the associated DTS file for the i.MX 93 EVK board (`arch/arm/dts/imx93-11x11-evk.dts`), the pin is not used. Therefore, there is nothing to disable.

To use the pin with the GPIO functionality, the following pin muxing must be added in the `iomuxc` node:

```
pinctrl_hog: hoggrp {
   fsl,pins = <
    MX93_PAD_GPIO_IO05__GPIO2_IO05        0x59e
   >;
  };
```

Note: *The `0x59e` configuration is based on the following pad settings in the `SW_PAD_CTL_PAD_GPIO_IO23` register:*

- ***Drive strength field:*** *X4*
- ***Slew rate field:*** *Fast slew rate*
- ***Pull up field:*** *No pull up*
- ***Pull down field:*** *Pull down*
- ***Open drain field:*** *Disabled*
- ***Schmitt trigger field:*** *No Schmitt input*

3.1.3 Updating the Linux device tree

To get the necessary files, the Linux sources must be downloaded:

```
$ git clone https://github.com/nxp-imx/linux-imx
$ cd linux-imx
$ git checkout lf-6.1.36-2.1.0
```

To update the device tree, the macro associated with the desired functionality must be found out. This information resides in the `arch/arm64/boot/dts/freescale/imx93-pinfunc.h` file. The macro that must be used in this context is `MX93_PAD_GPIO_IO05__GPIO2_IO05`. However, the usage of this macro is not enough for an adequate pin mux setup, because it requires a sixth value which represents the pad configuration. It is added in the associated device tree.

```
#define MX93_PAD_GPIO_IO05__GPIO2_IO05            0x0024 0x01D4 0x0000 0x0 0x0
#define MX93_PAD_GPIO_IO05__TPM4_CH0              0x0024 0x01D4 0x0000 0x1 0x0
#define MX93_PAD_GPIO_IO05__PDM_BIT_STREAM00      0x0024 0x01D4 0x0438 0x2 0x0
#define MX93_PAD_GPIO_IO05__MEDIAMIX_DISP_DATA01  0x0024 0x01D4 0x0000 0x3 0x0
#define MX93_PAD_GPIO_IO05__LPSPI7_SIN            0x0024 0x01D4 0x0000 0x4 0x0
#define MX93_PAD_GPIO_IO05__LPUART6_RX            0x0024 0x01D4 0x0000 0x5 0x0
#define MX93_PAD_GPIO_IO05__LPI2C6_SCL            0x0024 0x01D4 0x03F0 0x16 0x1
#define MX93_PAD_GPIO_IO05__FLEXIO1_FLEXIO05      0x0024 0x01D4 0x0380 0x7 0x0
```

Looking at the associated DTS file for the i.MX 93 EVK board (`arch/arm64/boot/dts/freescale/imx93-11x11-evk.dts`), it can be seen that the pin is not used. Therefore, there is nothing to disable.

To use the pin with the GPIO functionality, the following pin muxing must be added in `iomuxc`:

Note: *If the chosen pad has another pin mux configuration, the respective pin muxing must be replaced with the following one to successfully generate the pulse in Linux.*

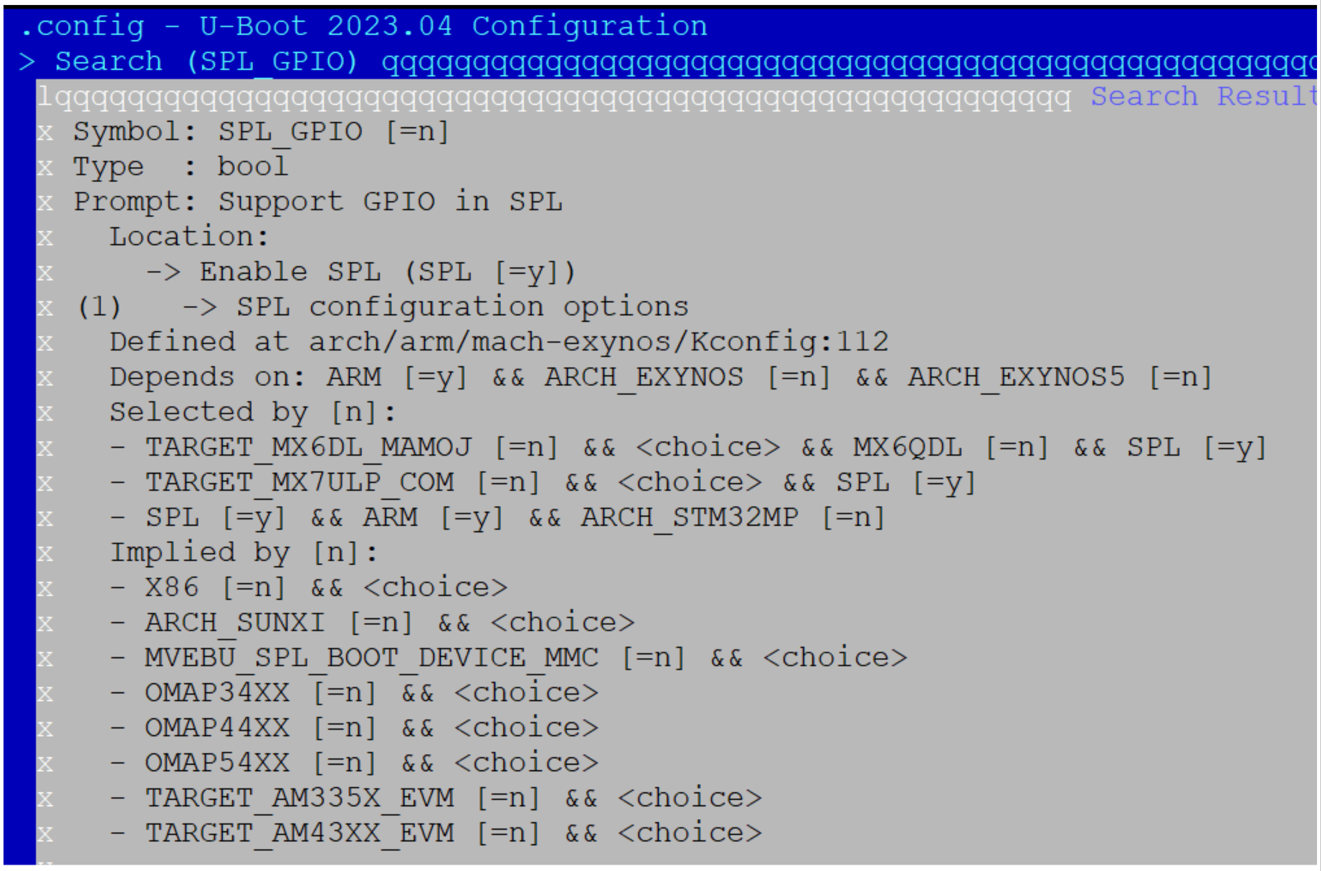
```
pinctrl_hog: hoggrp {
  fsl,pins = <
   MX93_PAD_GPIO_IO05__GPIO2_IO05  0x59e
  >;
 };
```

3.1.4 Adding the SPL GPIO support

To use the GPIO API in SPL, the SPL GPIO support must be enabled using `menuconfig`. Run the following commands to apply the default configuration and to open the configuration menu:

```
ARCH=arm CROSS_COMPILE=aarch64-linux-gnu- make imx93_11x11_evk_defconfig
ARCH=arm CROSS_COMPILE=aarch64-linux-gnu- make menuconfig
```

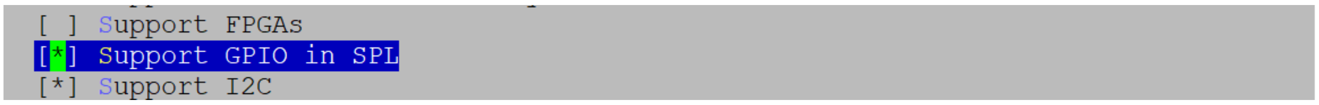
Use "/" for search, type `SPL_GPIO`, and press Enter.



```
.config - U-Boot 2023.04 Configuration
> Search (SPL_GPIO) qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
 lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq Search Result
 x Symbol: SPL_GPIO [=n]
 x Type  : bool
 x Prompt: Support GPIO in SPL
 x   Location:
 x     -> Enable SPL (SPL [=y])
 x (1)   -> SPL configuration options
 x   Defined at arch/arm/mach-exynos/Kconfig:112
 x   Depends on: ARM [=y] && ARCH_EXYNOS [=n] && ARCH_EXYNOS5 [=n]
 x   Selected by [n]:
 x   - TARGET_MX6DL_MAMOJ [=n] && <choice> && MX6QDL [=n] && SPL [=y]
 x   - TARGET_MX7ULP_COM [=n] && <choice> && SPL [=y]
 x   - SPL [=y] && ARM [=y] && ARCH_STM32MP [=n]
 x   Implied by [n]:
 x   - X86 [=n] && <choice>
 x   - ARCH_SUNXI [=n] && <choice>
 x   - MVEBU_SPL_BOOT_DEVICE_MMC [=n] && <choice>
 x   - OMAP34XX [=n] && <choice>
 x   - OMAP44XX [=n] && <choice>
 x   - OMAP54XX [=n] && <choice>
 x   - TARGET_AM335X_EVM [=n] && <choice>
 x   - TARGET_AM43XX_EVM [=n] && <choice>
```

Figure 3. SPL_GPIO configuration in menuconfig

To enable this configuration, press "1". This highlights the variable:



```
[ ] Support FPGAs
[*] Support GPIO in SPL
[*] Support I2C
```

Figure 4. Enable the support for GPIO in SPL

Press the space bar to enable it. A "*" should appear.

3.1.5 Adding the first pulse generator in board/freescale/imx93_evk/spl.c

Now the pin must be configured. A macro containing the pair of the GPIO group and GPIO pin must be declared in the file.

Note: *In the U-Boot software, the GPIO group numbering starts from zero, while the physical GPIO group numbering starts from one. Therefore, the associated software GPIO group of the physical GPIO group two is one.*

```
#define TIMED_GPIO IMX_GPIO_NR(1, 5)
```

Now the pulse generation code can be added into the `board_init_f` function, after the basic SPL initializations.

```
void board_init_f(ulong dummy)
{
    int ret;

    /* Clear the BSS. */
    memset(__bss_start, 0, __bss_end - __bss_start);

    timer_init();

    arch_cpu_init();

    board_early_init_f();

    spl_early_init();

    gpio_request(TIMED_GPIO, "timed_gpio");
    gpio_direction_output(TIMED_GPIO, 1);
    gpio_direction_output(TIMED_GPIO, 0);

    preloader_console_init();
```

3.1.6 Adding the second pulse generator in include/configs/imx93_evk.h

The commands responsible for toggling the pin are added in the environment variables for the bootloader, at the load image property. The pin is set before loading the image and reset afterward.

```
"loadimage=gpio set GPIO2_05; fatload mmc ${mmcdev}:${mmcpart} ${loadaddr}
${image}; gpio clear GPIO2_05\0" \
```

This variable can also be modified during U-Boot boot. Power up the board and stop it in U-Boot, by pressing any key. The `loadimage` variable can be modified using the `edit` command. After the modification, use the `saveenv` command to make this change permanent.

After modifing the U-Boot source files, the files must be rebuilt and written on the SD card. To build the U-Boot sources, see Section 4.5.12 and Section 4.5.13 from the *i.MX Linux User Guide* (document IMXLUG).

After building the U-Boot image, write it on the SD card:

```
$ sudo dd if=flash.bin of=/dev/sd<x> bs=1k seek=32 conv=fsync
```

Note: *Check your card reader partition and replace `sd<x>` with your corresponding partition.*

3.1.7 Adding the third pulse generator in init/main.c

In this case, the third pulse is generated similarly to the first one. To use the GPIO API, the GPIO library and the macro that defines the GPIO pin number must be included. Add the following lines at the beginning of the file:

```
#include <linux/gpio.h>
#define IMX_GPIO_NR(port, index)     ((((port)-1)*32)+((index)&31))
```

The macro containing the pair of the GPIO group and GPIO pin must be declared in the file.

```
#define TIMED_GPIO IMX_GPIO_NR(2, 5)
```

Now, add the third pulse in the `run_init_process()` function. In this function, the `systemd` process is started. Add the pulse at the beginning of the function:

```
static int run_init_process(const char *init_filename)
```

```
{
      const char *const *p;

argv_init[0] = init_filename;

gpio_request(TIMED_GPIO, "timed_gpio");
gpio_direction_output(TIMED_GPIO, 1);
gpio_direction_output(TIMED_GPIO, 0);
```

To build the kernel sources, see Section 4.5.12 in the *i.MX Linux User Guide* (IMXLUG).

To write the kernel image on the SD card, use the following commands:

```
$ sudo mount /dev/sd<x>1 /mnt
$ cp Image /mnt
$ umount /mnt
```

3.1.8 Measuring the total time with the logic analyzer

The measuring stand is set up by connecting the logic analyzer to the board. The signal used as a starting reference is `JTAG_RESET`, which can be found on the JTAG connector at pin 10.

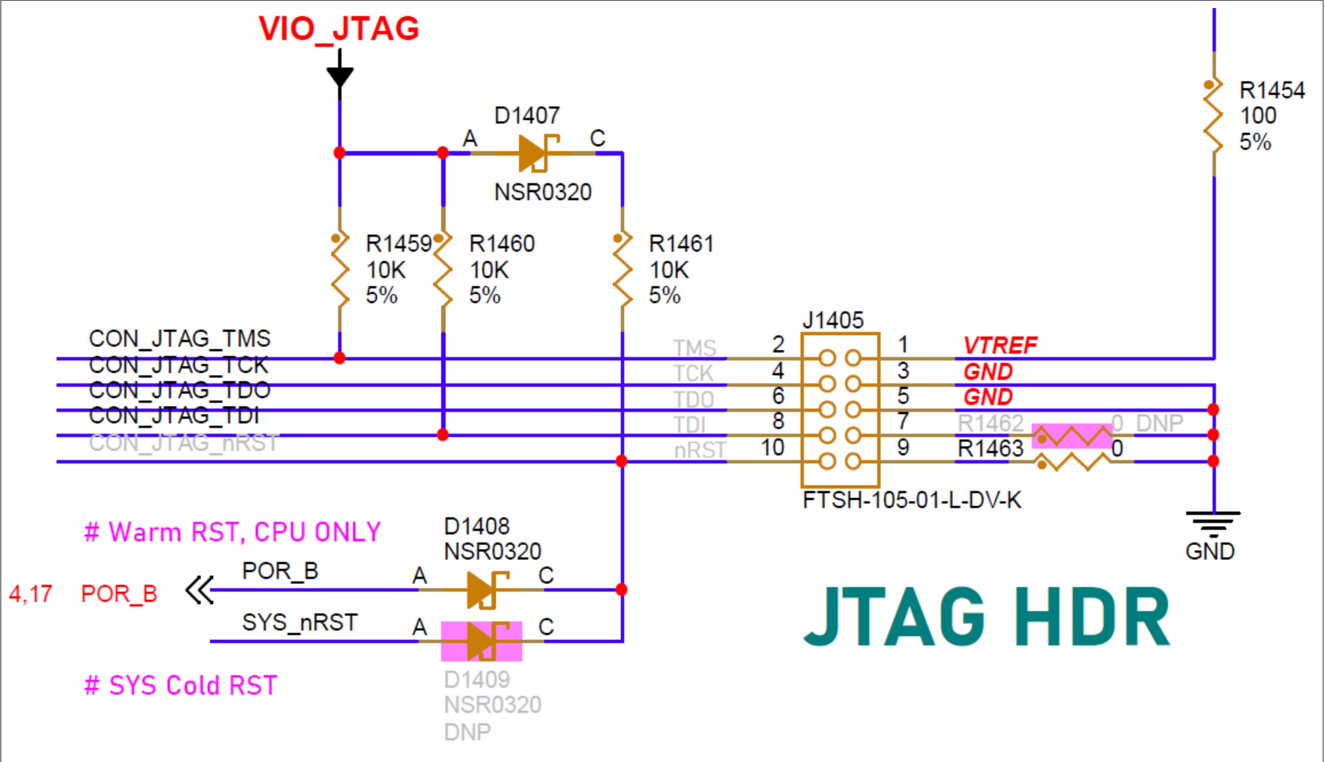


Figure 5. Schematic view of the JTAG connector available on the i.MX 93 EVK board

To capture the rising edges of the chosen pin, the second probe of the analyzer must be connected to the 29th pin on the J1001 expansion connector.

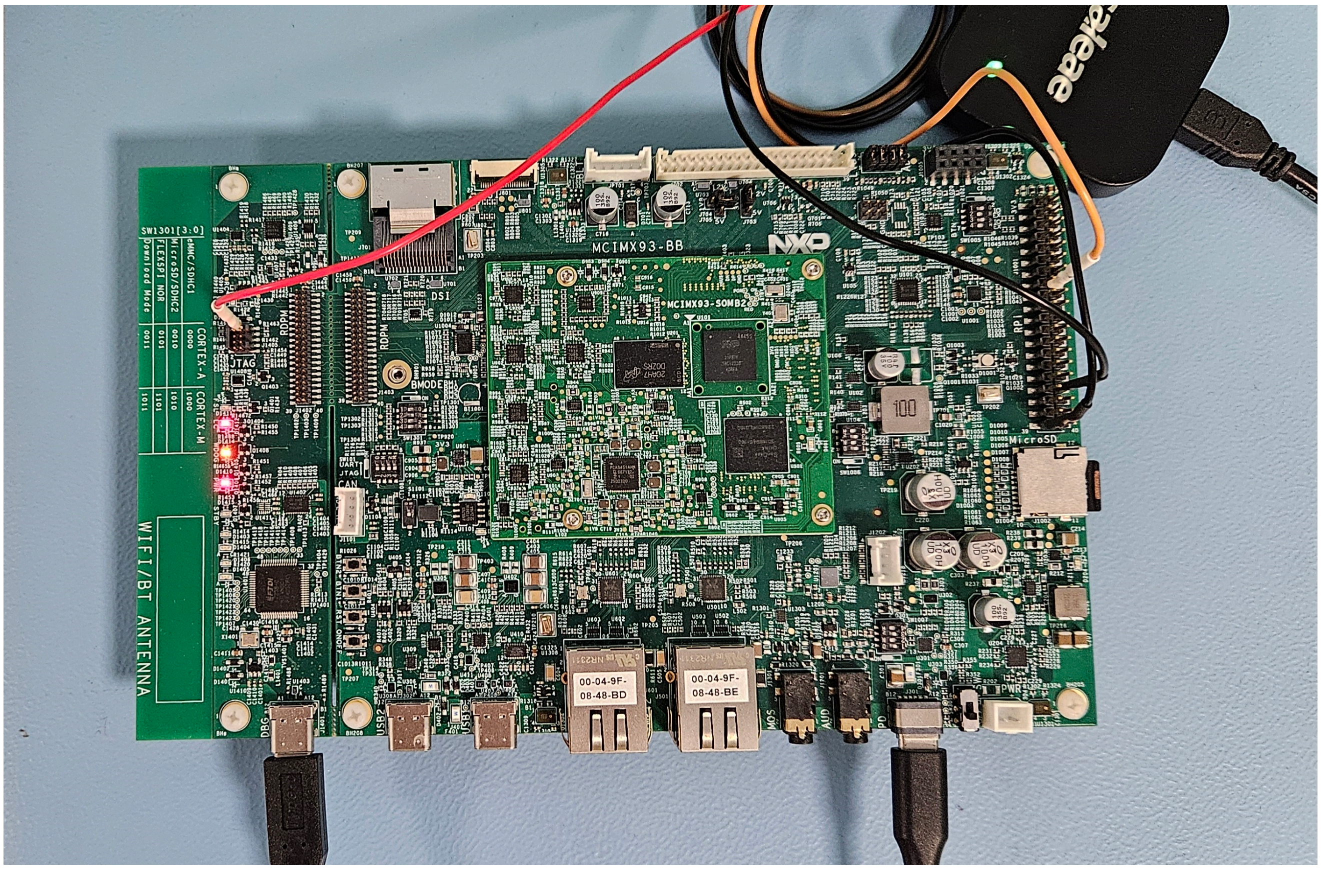
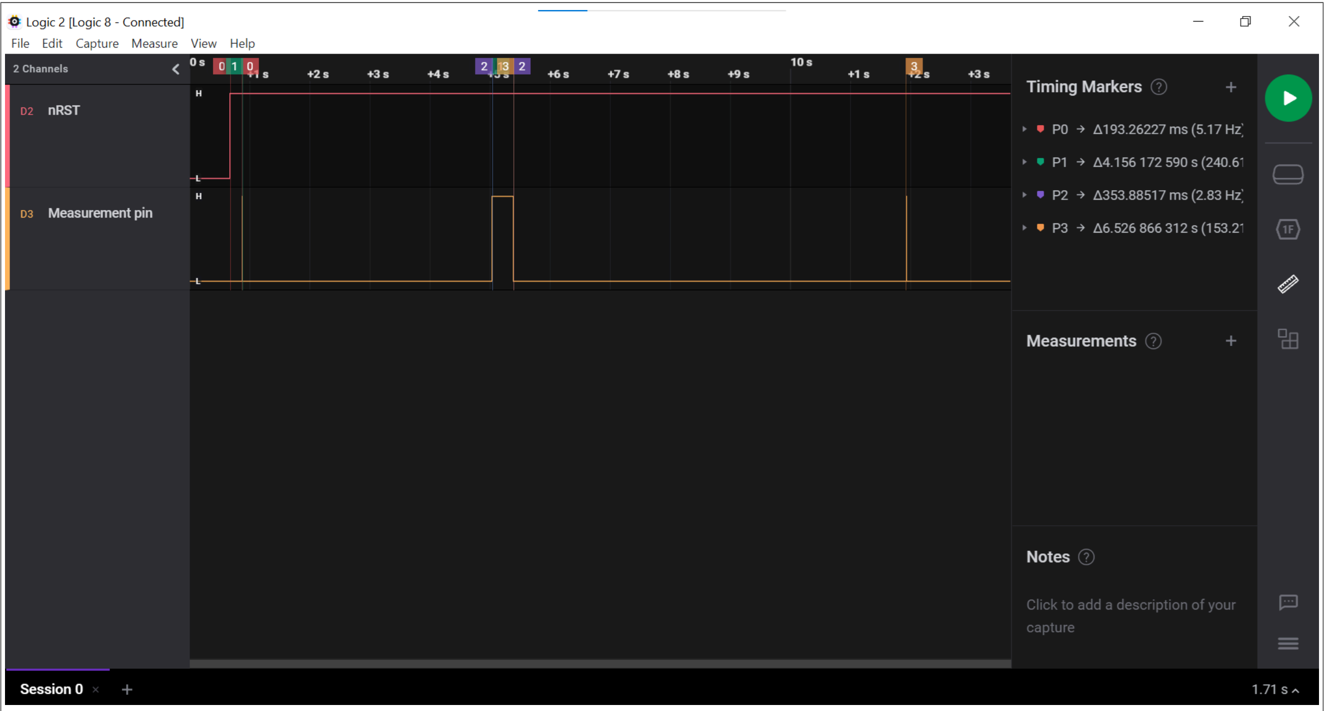


Figure 6. Measurement setup on the i.MX 93 EVK Board

After checking that both probes are referenced to the board’s ground, the logic analyzer software is started, where the probe parameters and the time frame can be set up.

A rising edge on the `JTAG_RESET` pin, followed by three pulses on the chosen pin, should be seen. At this moment, the recording is stopped and measurement flags are placed to find out the time for each phase. In this configuration, the boot time of the board is calculated from the sum of the elapsed time for each stage.



P0 → Δ193.26227 ms (5.17 Hz)
P1 → Δ4.156 172 590 s (240.61
P2 → Δ353.88517 ms (2.83 Hz)
P3 → Δ6.526 866 312 s (153.21

Figure 7. Measurements on the boot time for the i.MX 93 EVK board

4 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

5 Revision history

Table 3 summarizes the revisions to this document.

Table 3. Revision history

Document ID	Release date	Description
AN14205 v.1	31 January 2024	Initial public release

Contents

Date of release: 31 January 2024
Document identifier: AN14205