Freescale Semiconductor
Release Notes

Document Number: MQXIMX6SXRN
Rev. 1.0, 01/2015

Freescale MQX™ RTOS 4.1.0 i.MX 6SoloX Release Notes

1 Introduction

These are the Release Notes for the i.MX 6SoloX standalone package for Freescale MQX™ RTOS 4.1.0. Freescale i.MX 6SoloX belongs to the i.MX 6 series processor family of 32-bit application processors, and provides one ARM® Cortex®-A9 core and one Cortex®-M4 core. The software is built based on the Freescale MQX RTOS version 4.1.0 and runs on the i.MX 6SoloX Cortex-M4 core. It includes the full set of RTOS services and a standard set of peripheral drivers. Prior installation of Freescale MQX RTOS 4.1.0 is not required to install this package.

For more information, see the *Freescale MQX™ RTOS 4.1.0 Release Notes* (MQXRN) and *Getting Started with Freescale MQX™ RTOS* (MQXGSRTOS).

Contents

1.1 Development tools

The i.MX 6SoloX release was tested with the following development tools:

- IAR Embedded Workbench® for ARM® Version 7.10.3 (Windows OS only)
- ARM-GCC V4.8 2014_Q1
- ARM Development Studio 5 (DS-5™) version 5.18.0
- Lauterbach TRACE32 PowerView for ARM version S.2012.04.000036477 (Windows OS only)

1.2 System requirements

The system requirements are defined by the development tool requirements. There are no special host system requirements for the Freescale MQX RTOS distribution.

- Minimum PC configuration:
    - As required by Development and Build Tools
- Recommended PC configuration:
    - 2GHz processor – 2 GB RAM – 2 GB free disk space
- Software requirements:
    - OS: Windows 7 or later, or Ubuntu 12.04 or later

1.3 Target requirements

The i.MX 6SoloX package was tested with the following hardware configuration:

- i.MX 6SX SABRE-SDB Rev. B with a PCIMX6X4EVM10AB processor
- i.MX 6SX SABRE-AI MCIMX6SXAICPU2 Rev. A with a PCIMX6X4AVM08AB processor
    - Extension board MCIMXABASEV1 Rev. E for peripherals.

2 Features

2.1 Key features

This package provides initial support for the i.MX 6SoloX System module and a standard set of features and example applications.

This section describes the major changes and new features implemented in this release.

- Cortex-M4 core clock: 227 MHz (default)
- AHB bus clock: 132 MHz (default)
- BSP Timer: SysTick
- Default console: ittyb

The package supports these features:

- PSP support for the i.MX 6SoloX System module
- BSP for the i.MX 6SX SABRE-SDB board and i.MX 6SX SABRE-AI board
- MCC for the communication between i.MX 6SoloX Cortex-A9 and Cortex-M4
- Standard set of I/O drivers supporting the i.MX 6SoloX peripherals, including:
    - ADC driver
    - FlexCAN driver
    - I2C interrupt and polled master/slave driver
    - GPIO driver
    - HWTIMER driver with EPIT
    - UART interrupt and polled driver
    - SPI master/slave driver
    - Sensor drivers
        - MAG3110 Magnetometer
        - MMA8451Q Accelerometer
- Example applications demonstrating MQX RTOS, MCC usage and low power functions.
- eCompass example demonstrating the sensor’s capabilities.
- RDC settings
    - MQX RTOS on Cortex M4 runs in RDC domain 1
    - Peripherals reserved for RDC domain 1 access only

        - ADC1
        - ADC2
        - UART2
        - I2C3
        - ECSPI4
        - ECSPI5
        - CAN1
        - CAN2
        - EPIT1
        - EPIT2
        - WDOG3
    - MQX code region is reserved for RDC domain 1 access only
        - Flash target
            - SDB: [0x78000000, 0x78040000)
            - AI: [0x68000000, 0x68040000)
        - RAM target: [0x00900000, 0x00920000)
    - Peripheral driver that supports RDC SEMA42 shared access
        - GPIO driver

2.2 Limitations

2.2.1 Boot loader

On the i.MX 6SoloX chip, Cortex M4 is an auxiliary core and cannot boot by itself, thus MQX RTOS must be loaded and booted by a loader executing on the main core or through a debugger script.

In this release, we provide two kinds of loaders:

- U-Boot for the board
- TRACE32 script (debugger, on Windows OS only)

In our examples, the target image is running out of the QSPI flash, and U-Boot can be used to kick off MQX RTOS.

To enable the TRACE32 debugger loader, the first step is to rebuild the example with the RAM target link file, and then follow the instructions in “Run with TRACE32 debugger” to load the program. See Section 3 “Installation Instructions” for details.

2.2.2 Debugger

Although IAR and DS5 are supported to build the MQX RTOS and examples, the debugger part of these IDE tools are not enabled.

Only TRACE32 is able to debug the programs built from ARM-GCC, IAR and DS5.

2.2.3 Flash downloader

i.MX 6SoloX flash downloader is not supported in the TRACE32 debugger, and only the RAM target program can be debugged at present.

2.3 Example applications

This package contains applications demonstrating the kernel and peripherals on the i.MX 6SoloX System. The applications can be found at the following locations:

- `<install_dir>/mqx/examples`: standard set of examples for kernel features and basic peripheral drivers
- `<install_dir>/mcc/examples`: standard set of examples for MCC features

2.4 Release contents

This section provides an overview of the release content.

Deliverable	Location
MQX PSP Source Code and Examples	<install_dir>/mqx/...
MQX PSP source code for i.MX 6SoloX CM4	.../mqx/source/psp/cortex_m
MQX PSP build projects	.../mqx/build/<tool>/psp_imx6sx_<board>_m4/…
MQX example applications	.../mqx/examples/...
MQX BSP Source Code	<install_dir>/mqx/...
MQX BSP source code	.../mqx/source/bsp/imx6sx_<board>_m4
MQX BSP build projects	../mqx/build/<tool>/bsp_imx6sx_<board>_m4/…
MQX MCC Source Code and Examples	<install_dir>/mcc/...
MCC source code	.../mcc/source
MCC build projects	.../mcc/build/<tool>/mcc_imx6sx_<board>_m4/…
MCC examples applications	.../mcc/examples/…
PC Host Tools and U-Boot	<install_dir>/tools
Toolchain Plug-ins	<install_dir>/tools/ds5
U-Boot for SDB board	<install_dir>/tools/u-boot-sdb.imx
U-Boot for SABRE-AI board	<install_dir>/tools/u-boot-ai.imx
Documentation	<install_dir>/doc
User Guides and Reference Manuals for MQX RTOS, IO Drivers, MCC, etc.	…/doc

3 Installation Instructions

3.1 Installation guide

For Windows: Run the i.MX 6SoloX MQX RTOS package installer and proceed according to instructions. This package does not require prior installation of Freescale MQX RTOS 4.1.0.

For Ubuntu: Unpack the i.MX 6SoloX MQX RTOS compressed package. This package does not require prior installation of Freescale MQX RTOS 4.1.0.

3.1.1 Build procedure

For build procedures, see *Getting Started with Freescale MQX™ RTOS* (MQXGSRTOS).

3.1.2 Default jumper settings

i.MX 6SX SABRE-SDB:

- J18 fitted
- SW10: 00000000
- SW11: 00111000
- SW12: 01000000

i.MX 6SX SABRE-AI:

- J1 1-2 fitted
- J2
    - 1-2 fitted without extension board
    - 2-3 fitted with extension board
- J3 2-3 fitted
- SW3: 00001100
- SW4: 01000010
- BOOT MODE: 0010

3.1.3 Board-specific build targets

- Flash (Debug and Release): These targets enable building applications suitable for booting MQX RTOS from the QSPI Flash memory, and they are the default link targets. The built program

should be loaded by U-Boot and can run on the Cortex-M4 core while Linux is running on the Cortex A9 core.

- RAM (Debug and Release): These targets are for MQX RTOS application debug only, and should not run at the same time as Linux to avoid memory conflicts. To build with RAM target, you have to modify the link file of the build project.

3.1.4 Board-specific build procedures

1. Flash target
    - Default target, see *Getting Started with Freescale MQX™ RTOS* (MQXGSRTOS)
2. RAM target
    - The instructions are the same as the Flash target except one additional step when building application: change the link file to ram.xxx. For example, change from `lib/imx6sx_sdb_m4.iar/debug/bsp/intflash.icf` to `lib/imx6sx_sdb_m4.iar/debug/bsp/ram.icf` in the red boxes in Figures 1 and 2 below.
    - For ARM GCC, edit `<install_dir>/build/common/make/global.mak`,
        - Find `GET_BSP_LINKER_FILE = $(firstword $(wildcard …))`.
        - Set `GET_BSP_LINKER_FILE=<RAM link file path>`.
        - Once it is finished, all applications built with GCC will change to the RAM target.

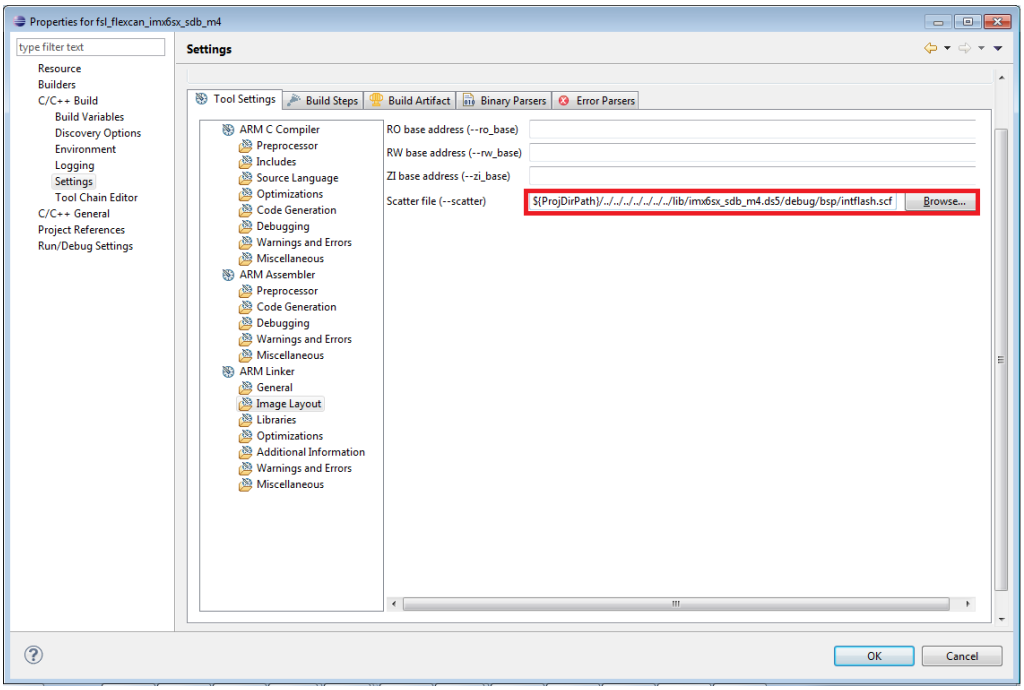

Figure 1 Scatter file in DS5 project properties

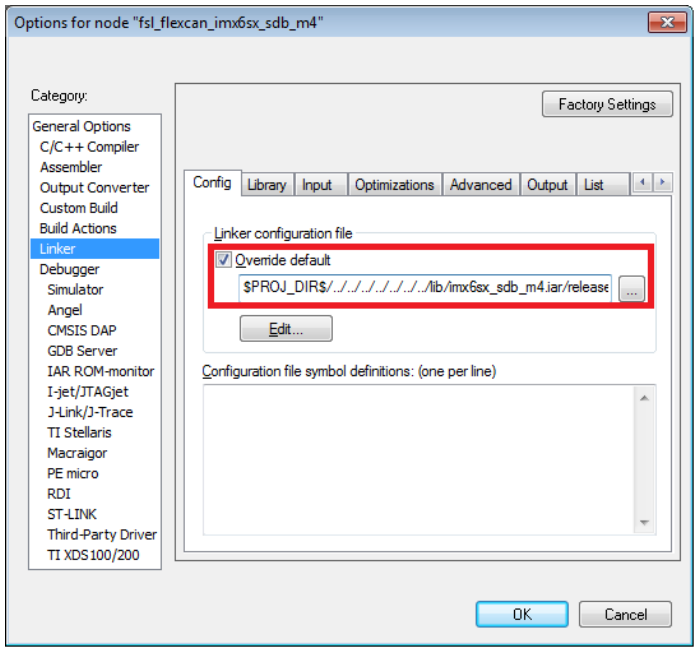

Figure 2 Linker configuration file in IAR project options

3.2 Running the examples

This section describes how to run the MQX RTOS application by itself without Linux running on A9. To run Linux and MQX RTOS at the same time on the i.MX 6SoloX SABRE-SDB or SABRE-AI board, load MQX RTOS with the U-Boot loader in the Linux package, using the same steps as in Section 3.2.1, “Run with U-Boot”.

3.2.1 Run with U-Boot

There is a prebuilt U-Boot image available to kick off the MQX RTOS application. You can find it at `<install_dir>/tools/u-boot-<board>.imx`. This is the U-Boot loader for running MQX RTOS only.

The default boot configuration of the SDB board is boot from SD4, and on SABRE-AI board there’s only one SD slot which is used for boot. You need to prepare an SD card, and then create a single FAT partition with 2 MB offset from MBR (by Microsoft Windows® 7 “diskpart” tool, or Ubuntu “fdisk” tool), then write U-Boot at 1 KB offset (by Windows application ddcopy.exe located in <install_dir>/tools/ddcopy, or Ubuntu “dd” tool).

Here we only show how to do it on Microsoft Windows 7.

1. Open “command prompt”, and run “diskpart”.
    a. `list disk`: Shows all the available disks.
    b. `select disk [num]`: Choose the disk number that represents your SD card, such as 1.

c. `clean all`: Remove all the partitions from disk 1, which takes some time to complete.

d. `create partition primary offset=2048 size=512`: create the primary partition of 512 MB with the offset of 2048 KB.

e. Safely remove the SD card from the slot and plug it in again. Then you can format the partition to the FAT file system.

2. Then you can write U-Boot to the SD card, insert your SD card, and get the Windows partition number, such as "F:".Open "command prompt", and run "cmd".

   a. Change the directory to `<install_dir>/tools/ddcopy`.

   b. `ddcopy.exe infile=../u-boot-<board>.imx outdevice=F: seek=0x400 obs=512`

   c. Safely remove the SD card from the slot.

   You can now prepare the MQX RTOS application image to be loaded by U-Boot. For instructions on how to build applications to create a binary image, see the getting started guides of your toolchain. After the binary image is ready, you can copy it to the FAT system on the SD card.

3. Insert the SD card to the boot slot, and connect the "UART to USB" slot on the board with your PC through the USB cable. The Windows OS will install the USB driver automatically, and Ubuntu OS will find the serial devices as well.

   On Windows OS, open the device manager, find "USB serial Port" in "Ports (COM and LPT)". Assume that the ports are COM7 and COM8. The smaller numbered port (COM7) is for the debug message from Cortex A9 and the larger numbered port (COM8) is for Cortex M4.

   On Ubuntu OS, find the TTY device with name /dev/ttyUSB* to determine your debug port. Similar to Windows OS, the smaller number is for A9 and the bigger number is for M4.

   Open your favorite serial terminals forthe serial devices, set the speed to 115200 bps, data bits 8, no parity, and power on the board.

4. On the COM7 terminal, press any key within 3 seconds of booting, and U-Boot will enter command line mode, and then you can write your MQX application to QSPI flash.

   For i.MX 6SoloX SABRE-SDB board, U-Boot command like below:

   a. `fatload mmc 2:1 0x80800000 image.bin`: Load the flash image file from the SD card to DDR.

   b. `sf probe 1:0`: Load the SPI flash driver.

   c. You should get the message "`SF: Detected N25Q256 with page size 256 Bytes, erase size 4 KiB, total 32 MiB`".

   d. `sf erase 0x0 0x40000`: Erase the first 256 KB in the flash.

   e. `sf write 0x80800000 0x0 0x40000`: Burn the image from DDR to the flash.

f. `bootaux 0x78000000`: Start the M4 core at the flash head.

For i.MX 6SoloX SABRE-AI board, U-Boot command is minor different:

a. `fatload mmc 0:1 0x80800000 image.bin`: Load the flash image file from the SD card to DDR.

b. `sf probe 1:0`: Load the SPI flash driver.

c. You should get the message "`SF: Detected N25Q256 with page size 256 Bytes, erase size 4 KiB, total 32 MiB`".

d. `sf erase 0x0 0x40000`: Erase the first 256 KB in the flash.

e. `sf write 0x80800000 0x0 0x40000`: Burn the image from DDR to the flash.

f. `bootaux 0x68000000`: Start the M4 core at the flash head.

The MQX RTOS application message is displayed on the COM8 terminal. Because the image is on the flash, if you want to run the same image again, reboot the board and repeat Step 2 "`sf probe 1:0`"and Step 6 "`bootaux 0x78000000`" for SABRE-SDB board, or "`bootaux 0x68000000`" for SABRE-AI board and you can also add these commands to the "bootcmd" variable of U-Boot to run it automatically.

3.2.2 Running with the TRACE32 debugger (on Windows OS only)

To debug the program with TRACE32, a script file is used. Find it at `<install_dir>/tools/trace32/attach_imx6sx_m4.cmm`.

Make sure that TRACE32 ICD (In-Circuit-Debugger) for ARM is installed, and your Lauterbach debugger device supports Cortex-M4 debugging.

If your TRACE32 version does not support Cortex-M4 debugging, copy all the files under `<install_dir>/tools/trace32/patches` to the TRACE32 install directory, such as `C:/T32`, and overwrite the original ones.

Build an MQX RTOS application of the RAM target, and change the default ELF load path in `attach_imx6sx_m4.cmm`.

data.load.elf "<your ELF path>" /verify

For "Task Awareness Debug" support, please also care about the "TASK.CONFIG" and "MENU.ReProgram" options, which load the TAD functions into TRACE32. The required files "mqx.t32" and "mqx.men" are located at `<install_dir>/tools/trace32`.

Connect the TRACE32 debugger device to your PC and the board (through JTAG). Run the TRACE32 ICD ARM debugger, and load `attach_imx6sx_m4.cmm` by choosing **File** -> **Run Batchfile**.

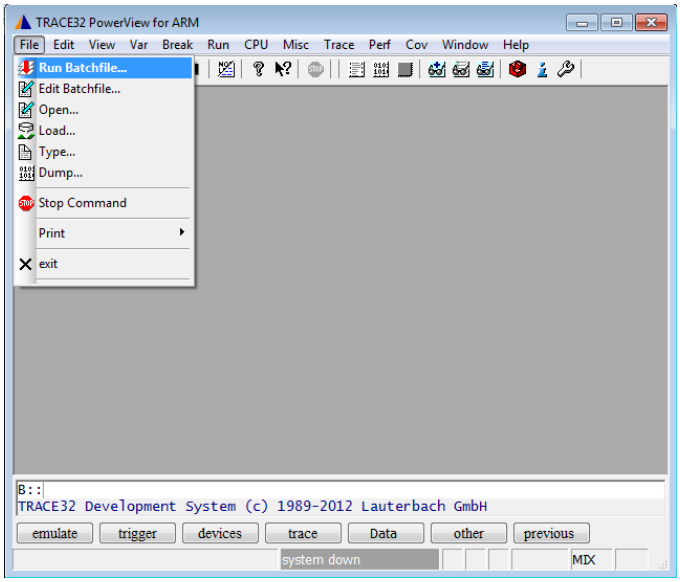

Figure 3 Running the script file

Now you can run (GO) and debug the program with the single step (Step, Over, Next, Return) or break points.

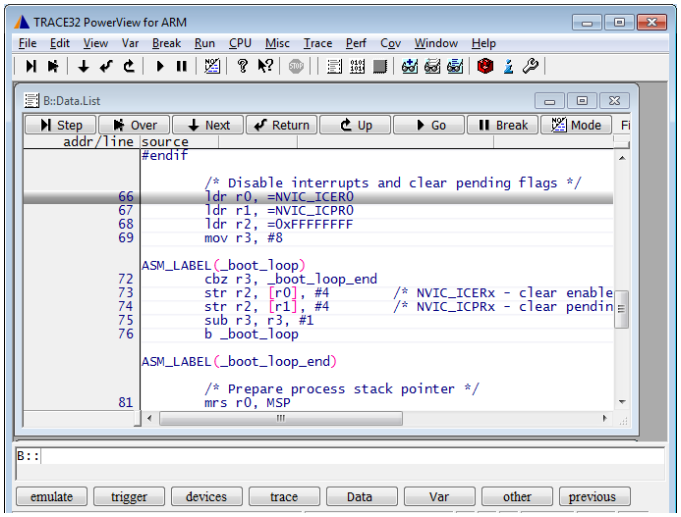

Figure 4 Starting debugging

For more information about the TRACE32 debugger, see `icd_tutorial.pdf` in your installation directory of TRACE32, such as. `C:/T32/pdf`.

4 Known Issues

- To get TCM fully used, _mem_extend() is called in BSP with MQX MEM allocator (highly recommended). Since LWMEM cannot support extending function, it can only manage half of the TCM as data area. This might cause out-of-memory issue (at linkage or runtime) with LWMEM configuration.
- In SPI master and slave example, it will reuse SD2 (WIFI) socket pins for connection, and this will conflict with Linux. Make sure to only run the SPI master/slave example without Linux.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
www.freescale.com/support

Document Number: MQXIMX6SXRN
Rev. 1.0
01/2015