FreeRTOS on FSiMX7ULP Boards

Manual on how to use/configuring the software

Version 1.2
(2020-03-31)

armStone efus NetDCU

PicoCOM PicoCore PicoMOD QBliss

About This Document

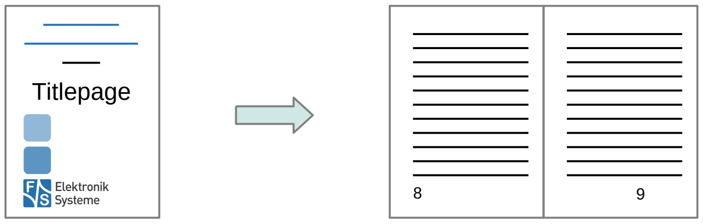
This document describes how to configure the Linux kernel, the device tree and the board to use it with FreeRTOS and its demo applications provided. The software is configured for PicoCoreMX7ULP from F&S under Linux/Buildroot.

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
2019-01-25	1.0	All	A	-	Derivate from MX6SX-Doku	PG
2020-03-18	1.1	All	A	- 1 8 7 9 8 8 8 1, 8 8	Update documentation Add Pin Assignment with corresponding cross-reference to examples Added note about boot process while using some examples Added custom board preparation manual Added List of Figures/Tables Removed wdog32, not supported for now Adjusted Sai_example output Added note for pf1550 example Added Pin assignment for GND and Voltage output Adjusted Output of freertos_tickless example	RS
2020-03-24	1.1	All	M	6	Explain two variants of how to add a custom board for fsimx7ulp	PJ
2020-03-31	1.2	All	M	3.3	Change naming of pre-compiled images in binaries directory from picocoremx7ulp/*-<v>.bin to picocoremx7ulp/*.img	PJ

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

Table of Contents

1 Pin Assignment

In the following subchapters you can find an overview which pins are used for each Board. The examples itself also contains the necessary pins.

1.1 PicoCoreMX7ULP

1.1.1 DAC

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
DAC0_OUT	J2_69	J7_21

1.1.2 GPIOS

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
Push-button	J3_29	J8_21
Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
Led	J2_48	J8_23

1.1.3 I2C_FLEXIO

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
I2C0_SCL	J2_63	J8_11
I2C0_SDA	J2_65	J8_10

1.1.4 I2C_LP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
I2C3_SCL	J2_49	J8_26
I2C3_SDA	J2_51	J8_24

1.1.5 PWM

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
PWM_OUT	J2_51	J8_24

1.1.6 SPI_FLEXIO

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
CLK	J2_65	J8_10
PCS	J3_31	J8_20
SOUT	J2_63	J8_11
SIN	J3_29	J8_21

1.1.7 SPI_LP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
CLK	J3_35	J8_18
PCS	J3_33	J8_19
SOUT	J3_37	J8_7
SIN	J3_39	J8_8

1.1.8 UART_FLEXIO

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
TX	J2_63	J8_11
RX	J2_65	J8_10

1.1.9 LPADC

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
ADC1_CH6A	J2_42	J8_27

1.1.10 ACMP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
ACMP External Input6	J2_65	J8_10

1.1.11 GPIO_LED

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
Led	J2_48	J8_23

1.1.12 Voltage output 3v3

Function	ULPBB Rev 1.3
3v3	J8_2

1.1.13 Voltage output 1v8

Function	ULPBB Rev 1.3
1v8	J8_1

1.1.14 GND

Function	ULPBB Rev 1.3
GND	J8_34

1.1.15 Push_Button

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
Push-button	J3_29	J8_21

2 Introduction

The F&S FreeRTOS_BSP-package is based on the MCUXpresso Software Development Kit (SDK) by NXP. It provides comprehensive software support for Kinetis and LPC Microcontrollers.

The MCUXpresso SDK includes a flexible set of peripheral drivers designed to speed up and simplify development of embedded applications. Along with the peripheral drivers, the MCUXpresso SDK provides an extensive and rich set of example applications covering everything from basic peripheral use case examples to full demo applications. The MCUXpresso SDK contains FreeRTOS and various other middleware to support rapid development.

3 Installation

This section describes the installation of the CST code-signing client files.

3.1 Installation of the GCC embedded toolchain

The examples are tested and can be built with the GCC embedded toolchain (gcc-arm-none-eabi-8-2019-q3-update), which can be found under https://launchpad.net/gcc-arm-embedded.

If the toolchain is not installed, you have to download the file and extract the content to your filesystem:

```
tar -xvjf gcc-arm-none-eabi-${version}.tar.bz2
```

where ${version} will be replaced by the corresponding version you've downloaded.

It is necessary to export the `ARMGCC_DIR` environment variable, if it´s not already exported:

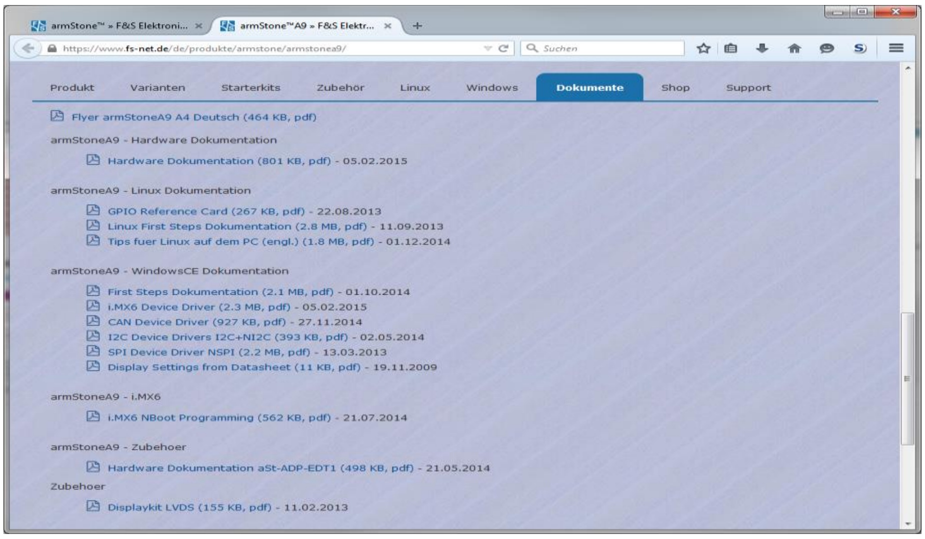
```
export ARMGCC_DIR=/usr/local/arm/gcc-arm-none-eabi-${version}
```

For a more convenient way you can add this to the rc file of your favorite shell (e.g. zshrc, bashrc, etc.)

3.2 Download Source Code

To download FreeRTOS source code, go to the F&S main website

http://www.fs-net.de


armStone™ » F&S Elektroni...
armStone™A9 » F&S Elektr...
https://www.fs-net.de/de/produkte/armstone/armstonea9/
Suchen
Produkt
Varianten
Starterkits
Zubehör
Linux
Windows
Dokumente
Shop
Support

Flyer armStoneA9 A4 Deutsch (464 KB, pdf)

armStoneA9 - Hardware Dokumentation

Hardware Dokumentation (801 KB, pdf) - 05.02.2015

armStoneA9 - Linux Dokumentation

GPIO Reference Card (267 KB, pdf) - 22.08.2013
Linux First Steps Dokumentation (2.8 MB, pdf) - 11.09.2013
Tips fuer Linux auf dem PC (engl.) (1.8 MB, pdf) - 01.12.2014

armStoneA9 - WindowsCE Dokumentation

First Steps Dokumentation (2.1 MB, pdf) - 01.10.2014
i.MX6 Device Driver (2.3 MB, pdf) - 05.02.2015
CAN Device Driver (927 KB, pdf) - 27.11.2014
I2C Device Drivers I2C+NI2C (393 KB, pdf) - 02.05.2014
SPI Device Driver NSPI (2.2 MB, pdf) - 13.03.2013
Display Settings from Datasheet (11 KB, pdf) - 19.11.2009

armStoneA9 - i.MX6

i.MX6 NBoot Programming (562 KB, pdf) - 21.07.2014

armStoneA9 - Zubehoer

Hardware Dokumentation aSt-ADP-EDT1 (498 KB, pdf) - 21.05.2014

Zubehoer

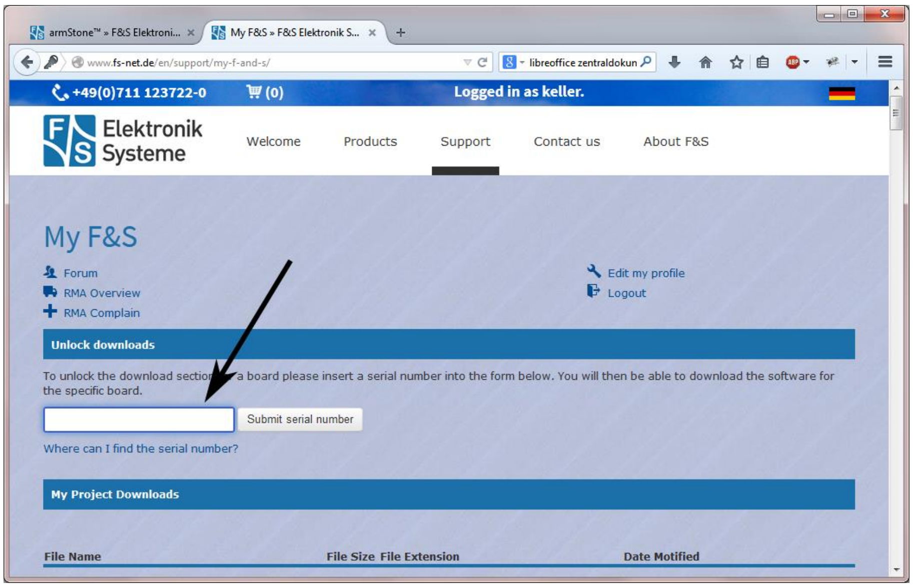
Displaykit LVDS (155 KB, pdf) - 11.02.2013

Figure 1: Register with F&S website

First you have to register with the website. Click on *Login* right at the top of the window and on the text “I am not registered, yet. Register now” (*Figure 1*).

In the screen appearing now, fill in all fields and then click on *Register*. You are now registered and can use the personal features of the website, for example the Support Forum and downloading software.

After logging in, you are at your personal page, called “My F&S”. You can always reach this place by selecting *Support → My F&S* from the top menu. Here you can find all software downloads that are available for you. In the top sections there are private downloads for you or your company (may be empty) and in the bottom section you will find generic downloads for all registered customers.



To unlock the download section [illegible] a board please insert a serial number into the form below. You will then be able to download the software for the specific board.

Figure 2: Unlock software with the serial number

To get access to the software of a specific board, you have to enter the serial number of one of these boards (see *Figure 2*). Click on “Where can I find the serial number” to get pictures of examples where to find this number on your product. Enter the number in the white field and press *Submit serial number*. This enables the software section for this board type for you. You will find Linux, Windows CE, and all other software and tools available for this platform like `DCUTerm` or `NetDCUUsbLoader`.

First click on the type of your board, e.g. PicoCoreMX7ULP, then on Linux. Now click on FreeRTOS. This will bring up a list of all our FreeRTOS releases. Old releases up to 2019 had <x>.<y> as version identifier, new releases use V<year>.<month>. We will abbreviate this as <v> from now on. Select the newest version, for example *freertos-fsimx7ulp-V2020.03*. This will finally show two archives that can be downloaded.

When you look at our Linux releases, you will find a list of all our releases and a README text. There are usually two files related to a release.

`freertos-fsimx7ulp-<v>.tar.bz2`	This is the main release itself containing all sources, the binary images, the documentation and the toolchain.

3.3 Release Content

These tar archives are compressed with bzip2. To see the files, you first have to unpack the archives

```
tar xvf freertos-<arch>-<v>.tar.bz2
```

This will create a directory `<arch>-<v>` that contains all the files of the release. They often use a common naming scheme:

`<package>-<platform>-<v>.<extension>`

With the following meaning:

`<package>`	The name of the package (e.g. `freertos-sdk`). If it is a source package, we also add the version number of the original package that our release is based on, for example `freertos-sdk-2.5.1`.
`<platform>`	The name of a board, if the package is only valid on one board (e.g. `PicoCoreMX7ULP`); or the name of an architecture, if the package is valid on different boards of the same architecture (e.g. `fsimx7ulp`), or the string `f+s` or `fus` if the package is architecture independent.
`<v>`	Release version, consisting of a letter `V` for version and the year and month of the release (e.g. `V2020.03`).
`<extension>`	The extension of the package (e.g. `.bin`, `.tar.bz2`, etc.).

The following table lists the files that you get after unpacking the release archive. To avoid having a too excessive list, we use the wildcard `*` in some entries to refer to a whole group of similar file names that only differ in the name of the board or module.

Directory/File	Description
`/`	**Top directory**
`Readme-freertos-f+s.txt`	Release information (FreeRTOS)
`setup-freertos`	Script to unpack FreeRTOS source packages to a build directory
`binaries/`	**Images to be used with the board directly**
`picocoremx7ulp/*.img`	Precompiled examples for PicoCoreMX7ULP
`sources/`	**Source packages**
`freertos-sdk-2.5.1-fsimx7ulp-V2020.03.tar.bz2`	FreeRTOS source
`toolchain/`	**Cross-compilation toolchain**
`gcc-arm-none-eabi-8-2019-q3-update-linux.tar.bz2`	ARM toolchain to use with `<arch>`
`doc/`	**Documentation**
`FreeRTOS_on_FSiMX7ULP_Boards_eng.pdf`	Manual on how to use/configuring the software

Table 1: Content of the created release directory

3.4 Unpacking the Source Code

The source code packages are located in the `sources` subdirectory of the release archive. We will assume that you want to create a separate build directory where you extract the source code and build all the software.

We have prepared a shell script called `setup-freertos` that does this installation automatically. Just call it when you are in the top directory of the release and give the name of the build directory as argument.

```
cd <release-dir>
./setup-freertos <build-dir>
```

Add option `--dry-run` if you want to check first what this command will do. Then only a list of actions will be output but no actual changes will take place. For further information simply call

```
./setup-freertos --help
```

If you prefer to do the installation by hand, well, the script more or less executes the following commands, just with some more checks and directory switching.

```
mkdir <build-dir>
tar xf freertos-sdk-2.5.1-fsimx7ulp-<v>.tar.bz2
```

3.5 Description of the FreeRTOS directory structure

The following table describes the directory structure of the

/	Top Directory
bin	After you have run the make command the output binaries or images can be found here in their specific $boardname-directory.
build	After you have run the make command, this directory contains the .bin, .elf, .hex, .map and object files for each example.
CMakeFiles	Contains Cmake-specific files. Normally you don't have to change anything in here.
CMSIS	Contains the Cortex Microcontroller Software Interface Standard (CMSIS) library.
devices	Contains socket specific files and drivers.
doc	Contains the original documentation by NXP.
examples/	Contains the SoC and board specific Cortex-M4 examples. The first level distinguishes between the different SoC-architectures. At the second level you will find the SoC specific examples. For the MX7ULP-examples the board specific examples are located directly in the directory of each example. The examples are structured as follows:
demo_apps	Here you can find the applications which highlight certain key features of the ARM Cortex-M4 Core combined with FreeRTOS and bare metal. Because most of the examples use special onboard sensors, they did not get ported.
driver_examples	You can find simple applications here which are intended to show peripheral drivers working in the bare metal environment.
multicore_examples	Here you can find examples, which demonstrate the multicore communication via RPMsg and erpc.
rtos_examples	These examples show the usage of different FreeRTOS-specific functions.
mmcau_examples	This example demonstrates the Memory-Mapped Cryptographic Acceleration Unit (MMCAU).
middleware	Contains application independent code like the dma manager or rpmsg lite.
not_tested	Contains examples that have not been tested yet. This can have different reasons like missing sensors or hardware on the EVK.

	Some of them will be ported in the future. If you are interested in porting one of these examples please contact F&S Electronic Systeme. Please refer to the readme.txt located at not_tested/<soc>/ for further information.
rtos	Contains the operating system freeRTOS.
tools	Contains different tools needed for the building process.

Table 2: description of the directory structure

4 Configuration for Cortex-M4 usage

4.1 Boot modes

There are two boot modes available for the imx7ulp.

In **single boot-mode** the Cortex-M4 boots up first, starts the Cortex-A7 and waits until the A7 provides it with an image file.

In **dual boot-mode** the Cortex-M4 starts the Cortex-A7 and automatically runs from an image, placed in the QSPI-flash memory.

In **low power boot-mode** the Cortex-M4 runs from an image, placed in the QSPI-flash memory. The Cortex-A7 can be booted from M4 on demand.

Attention

The default-configuration is single boot. For dual boot a fuse has to be burned, which cannot be undone!

The F&S-examples where all tested in single boot-mode. If you are interested in running them in dual boot-mode please contact F&S for further information.

4.2 Changes regarding official U-Boot

F&S provides you with a modified U-Boot which can make use of the Cortex-M4 via the *bootaux* command. Since our U-Boot is heavily modified compared to the official release from NXP, it's not advisable to use any other than the one provided by F&S.

F&S added some environment variables to simplify the auxiliary core handling:

Run

```
setenv bootauxfile <example_name>
```

to set the name of the example you want to load.

Run

```
run .auxcore_tftp
```

to load the examples with an Micro USB cable via tftp to the board. Make sure the `serverip` and `bootauxfile` variables are set correctly before running this command.

Run

```
run .auxcore_mmc
```

to load the examples from the mmc. Make sure the `bootauxfile` variable is set correctly before running this command. By default F&S provides the power_mode_switch example to be executed when Linux boots up. If you change this, make sure your example uses the srtm app or remove the rpmsg node from the device tree, or else Linux will freeze.

Run

```
run .auxcore_none
```

if you don’t want to use the Cortex-m4 at all. Make sure to remove the rpmsg node from the device tree, or Linux will freeze.

4.3 Using bootaux

Simple start

Using the auxiliary core can be achieved by using the following command line inside of the U-Boot environment:

```
tftp ${bootauxfile}; cp.b $loadaddr 0x1FFD0000 $filesize; bootaux
0x1FFD0000
```

This will load an image defined by *bootauxfile* via tftp to your board, move it to the TCM and start the auxiliary core.

5 Building the examples

To simplify the process of building, configuring the examples and cleaning up we provide you with a set of bash scripts located in the root directory of the FreeRTOS release.

5.1 Prepare.sh

This script will configure board relevant settings and create symlinks to the board specific header files. You can execute the script in your terminal by typing

```
./prepare.sh
```

and follow the instructions:

```
Choose one of the following boards for which you want to build the
examples:

efusa9x[1]    picocoma9x[2]   picocoremx6sx[3]     pico-
coremx7ulp[4]

Enter number in []-brackets for the corresponding board: 4

Do you want a Release or Debug build?

(r/d) [default: r]: r

All set up, starting cmake...
```

Most of the examples can be run from TCM or directly from the QSPI-flash. In future releases it will be possible to choose this in the prepare.sh script but for now only TCM is supported.

5.2 Make

The `prepare.sh` script will configure and invoke *cmake* to generate a Makefile. After this, you can run

```
make -jN
```

To build all examples located in `examples/fsimx7ulp` and install the binaries to `bin/$BOARD`.

`N` is the number of cores your CPU have.

If you want to build a specific example just type

```
make -jN example_name.elf && make install/fast
```

to build and install the binary of the chosen example. If you don't know the name Type

```
make help
```

for a list of possible examples for make.

By executing

```
make clean-all
```

you can clean up all build files and binaries. This will be necessary if you make changes to the `CmakeLists.txt` in the root directory of the FreeRTOS release.

5.3 Building the Images

In order to run the binaries on the Cortex-M4, they need to get converted into image-files with an Interrupt-Vector-Table (IVT) at the beginning, which will look like this

```
Combined Image Info:

-------------------------------------
   base_addr        = 0x1ffd0000
   ivt_offset       = 0x00001000
   hab_ivt.hdr      = 0x412000d1
   hab_ivt.entry    = 0x1ffd2311
   hab_ivt.self     = 0x1ffd1000
   hab_ivt.csf      = 0x00000000
   hab_ivt.boot_data = 0x1ffd1020
   hab_ivt.dcd      = 0x1ffd1040
   boot_data.start  = 0x1ffd0000
   boot_data.size   = 0x0000b7c0
   boot_data.plugin = 0x00000000

 Info for CSF file generation
-------------------------------------
   ivt_addr  = 0x1ffd1000, ivt_offset  = 0x00001000
   app_addr  = 0x1ffd2000, app_offset  = 0x00002000, app_size = 0x????????
-------------------------------------
```

NXP provides a small tool, located at `tools/imgutil`, to create these images.

If you are using the prepare.sh script, the created binaries will automatically get moved to the imgutil-directory, converted into images and moved back to the `bin/$boardname`-folder, but now with the ending img. If you still need the original binaries, you can find them at

```
build/$boardname/$examplename/${debug/release}/
```

If you want to convert a single binary without the prepare.sh-script, copy it to

```
tools/imgutil/evkmcimx7ulp
```

and run

```
/mkimg.sh [ram|flash] $binary-name
```

$binary-name without the ending .bin!

6 Adding custom boards

If you're using a custom board, you can choose 2 different ways how to add a custom board.

6.1 Prepare script

Duplicate the folder examples/fsimx7ulp e.g. examples/test123 and then tell the prepare.sh script about its existence and create some configuration files (or simply copy the existing ones).

To tell the script about it, change the following lines in the prepare.sh script:

```
declare -a SUPPORTED_BOARDS=("efusa9x" "picocoma9x" "picocoremx6sx" “pico-
coremx7ulp” "BOARDXYZ")
```

where *boardname* represent the name of your board and an entry to

```
declare -a SUPPORTED_SOCS=("fsimx6sx" "fsimx6sx" "fsimx6sx" "fsimx7ulp" “test123”)
```

so the number of your *boardname* matches the number of its specific SOC.

After that you have to setup the board specific files in the corresponding examples. The board specific files are:

- board.c
- board.h
- pin_mux.c
- pin_mux.h

These files are located in every example and have to setup in every example separate.

6.2 Default directory example/fsimx7ulp

It is also possible in the default fsimx7ulp directory which is located in examples/fsimx7ulp. There you have to setup the board specific files in the corresponding examples. The board specific files are:

- board.c
- board.h
- pin_mux.c
- pin_mux.h

These files are located in every example and have to setup in every example separate.

7 FreeRTOS examples

In this chapter we will provide you with necessary information on the demo and driver applications.

The “**Description**” will inform you about the demo's purpose.

In the “**Modifications made**” section you will find useful information if changes were made to certain files by F&S and the reason behind these changes.

“**Changes needed**” is the most important section. You will find the information necessary to successfully build and execute the examples here.

The last section, “**Execute binary**” will tell you the required steps to execute the image built.

7.1 General build and run information

Connect your board via Micro-USB to your PC and build up a RNDIS-connection.

Connect UART0 (Cortex M4) and UART4 (Cortex A7) with two serial cables (serial-to-modem) to your PC.

Open up two Terminals and connect the UARTs via the COM interface and the following settings:

Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none
Transmit delay: 0 msec/char 0 msec/line

Build the examples like described in **Building the examples** and copy them to you tftp-directory.

Change the boot mode of the Cortex-M4 to tftp by running

```
run .auxcore_tftp
```

in the U-Boot.

Now simply run the commands described in the **Execute binary** section of each example.

Attention

If you want to run multiple examples successively, make sure to hard-reset the Cortex-M4 after each example!

If you want to boot Linux after starting an example in U-Boot use

```
run .auxcore_none
```

before booting to prevent your program to be overwritten.

Attention!

If you want to boot Linux while running an example that does not use the srtm app, you have to remove all rpmsg nodes from the device tree or Linux will freeze during the boot process. Every example, which needs this change to work on Linux, is labelled with a box like this

Beware that this is only experimental!

If you want a stable version include the srtm app into your example.

7.2 demo_apps

Remark

The documentation is based on the MCUXpresso SDK_2.5.1_EVKMCIMX7ULP package from NXP.

Some of the software examples provided by NXP expect a certain module or sensor to be available on the board. Since F&S boards do NOT provide these, the associated examples weren't ported at all.

7.2.1 hello_world

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The Hello World demo application provides a sanity check for the new SDK build environments and board bring up. The Hello World demo prints the "Hello World" string to the terminal using the SDK UART drivers. The purpose of this demo is to show how to use the UART, and to provide a simple project for debugging and further development.

Modifications made

None.

Changes needed

None.

Execute binary

Run

```
setenv bootauxfile "hello_world.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

The log below shows the output of the hello world demo in the terminal window:

```
hello world.
```

7.2.2 power_mode_switch

Description

The Power mode switch demo application demonstrates the use of power modes in the KSDK. The demo prints the power mode menu through the debug console, where the user can set the MCU to a specific power mode. The user can also set the wakeup source by following the debug console prompts. The purpose of this demo is to show how to switch between different power modes, and how to configure a wakeup source and wakeup the MCU from low power modes.

Tips:

This demo is to show how the various power mode can switch to each other. However, in actual low power use case, to save energy and reduce the consumption even more, many things can be done including:

- Disable the clock for unnecessary module during low power mode. That means, programmer can disable the clocks before entering the low power mode and re-enable them after exiting the low power mode when necessary.
- Disable the function for unnecessary part of a module when other part would keep working in low power mode. At the most time, more powerful function means more power consumption. For example, disable the digital function for the unnecessary pin mux, and so on.
- Set the proper pin state (direction and logic level) according to the actual application hardware. Otherwise, the pin current would be activated unexpectedly waste some energy.
- Other low power consideration based on the actual application hardware.
- Debug pins (e.g SWD_DIO) would consume additional power, had better to disable related pins or disconnect them.

Modifications made

Power_mode_switch.c, pin_mux.h, pin_mux.h, board.c, app_srtm.h:

Changed GPIO for the push-button to PTA31

Attention!

The Push-Button is not usable at the moment, we are already working on a solution for this Problem. Apart from this the example runs normally.

Changes needed

Remarkable configuration of the application (in power_mode_switch.c):

"SYSTICK_LLWU_WAKEUP":

The demo leverages LPTMR0 as systick timer, and supports FreeRTOS tickless idle. In tickless idle mode, LPTMR0 takes LPO 1kHz clock as clock source and will overflow in 65 seconds. If setting SYSTICK_LLWU_WAKEUP to "true", it means systick can wake up system in LLS/VLLS so that OS event like task delay or semaphore timeout may wake up SoC in addition to the wakeup source selected in application menu. Even no OS event occurs, the system will be woken up from LLS/VLLS every 65 seconds also to avoid LPTMR0 overflow which leads to systick loss. If setting SYSTICK_LLWU_WAKEUP to "false", then systick(LPTMR0) cannot wakeup SoC in LLS/VLLS.

"APP_ENABLE_GPIO_PAD_LOW_POWER":

This is an IO low power switch. If setting to "1", then the SoC IO leakage can be optimized with the limitation that only fixed voltage can be applied to the IO pads. Please read "GPIO pads operating range configuration" in Reference Manual SIM module carefully to avoid malfunction or even SoC pad damage.

If you want to use a push-button as wake up source connect

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
Push-button	J3_29	J8_21

and GND.

Function	ULPBB Rev 1.3
GND	J8_34

Execute binary

Run

```
setenv bootauxfile "power_mode_switch.img"; run .auxcore_tftp
run auxcore
```

to start the example.

If you want to test the Wake up operation, boot linux and then run

```
echo mem > /sys/power/state
```

at the A7-side to enter VLLS-mode.

Output

NOTE: Only input when the demo asks for input. Input entered at any other time might cause the debug console to overflow and receive the wrong input value.

```
Build Time: Feb 22 2018--15:36:23
    Core Clock: 115200000Hz
    Power mode: RUN

Select the desired operation

Press  A for enter: RUN      - Normal RUN mode
Press  B for enter: WAIT     - Wait mode
Press  C for enter: STOP     - Stop mode
Press  D for enter: VLPR     - Very Low Power Run mode
Press  E for enter: VLPW     - Very Low Power Wait mode
Press  F for enter: VLPS     - Very Low Power Stop mode
Press  G for enter: HSRUN    - High Speed RUN mode
Press  I for enter: VLLS     - Very Low Leakage Stop mode
Press  Q for query CA7 core power status.
Press  W for wake up CA7 core in VLLS/VLPS.
Press  T for reboot CA7 core.
Press  U for shutdown CA7 core.
Press  V for boot CA7 core.
Press  R for read PF1550 Register.
Press  S for set PF1550 Register.

Waiting for power mode select..
```

7.3 driver_examples

7.3.1 acmp

Interrupt

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The ACMP Interrupt project is a simple demonstration program that uses the SDK software. It compares the selected analog input with ACMP internal DAC output continuously. The purpose of this demo is to show how to use the ACMP driver in SDK software by interrupt way. The ACMP can be configured based on default configuration returned by the API ACMP_GetDefaultConfig(). The default configuration is: high speed is not enabled, invert output is not enabled, unfiltered output is not enabled, pin out is not enabled, offset level is level 0, hysteresis level is level 0.

This example project uses ACMP instance 1 to compare the voltage signal input from External Input with the voltage signal (half of VDDA) output by ACMP's internal DAC. The Terminal will print information corresponding to different comparison result.

Modifications made

None.

Changes needed

Connect

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
ACMP External Input6	J2_65	J8_10

to a stable external voltage generator to avoid floating voltage.

Function	ULPBB Rev 1.3
3v3	J8_2

Execute binary

Run

```
setenv bootauxfile "acmp_interrupt_print.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

```
The example compares analog input to the reference DAC output(CMP
positive port).
The terminal will print CMP's output value when press any key.
Please press any key to get CMP's output value.
The analog input is LOWER than DAC output
The analog input is HIGHER than DAC output
```

Polling

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The ACMP Polling project is a simple demonstration program that uses the SDK software. It compares the selected analog input with ACMP internal DAC output continuously. The purpose of this demo is to show how to use the ACMP driver in SDK software by polling way. The ACMP can be configured based on default configuration returned by the API ACMP_GetDefaultConfig(). The default configuration is: high speed is not enabled, invert output is not enabled, unfiltered output is not enabled, pin out is not enabled, offset level is level 0, hysteresis level is level 0.

This example project uses ACMP instance 1 to compare the voltage signal input from External Input with the voltage signal (half of VDDA) output by ACMP's internal DAC. The Terminal will print information corresponding to different comparison result.

Modifications made

None.

Changes needed

Connect

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
ACMP External Input6	J2_65	J8_10

to a stable external voltage generator to avoid floating voltage.

Function	ULPBB Rev 1.3
3v3	J8_2

Execute binary

Run

```
setenv bootauxfile "acmp_polling_print.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

```
The example compares analog input to the reference DAC output(CMP
positive port).
The terminal will print CMP's output value when press any key.
Please press any key to get CMP's output value.
The analog input is LOWER than DAC output
The analog input is HIGHER than DAC output
```

7.3.2 crc

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The CRC Example project is a demonstration program that uses the SDK software to generate checksums for an ASCII string. Several CRC protocols are implemented using the CRC driver API.

Modifications made

None.

Changes needed

None.

Execute binary

Run

```
setenv bootauxfile "crc.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the example runs successfully, the following message is displayed in the terminal:

```
CRC Peripheral Driver Example


Test string: 123456789
CRC-16 CCIT FALSE: 0x29b1
CRC-16 MAXIM: 0x44c2
CRC-16 KERMIT: 0x2189
CRC-32: 0xcbf43926
CRC-32 POSIX: 0x765e7680
```

7.3.3 dac12

basic

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The dac12_basic example shows how to use DAC12 module simply as the general DAC12 converter.

When the DAC12's fifo feature is not enabled, Any write to the DATA register will replace the data in the buffer and push data to analog conversion without trigger support. In this example, it gets the value from terminal, outputs the DAC12 output voltage through DAC12 output pin.

Modifications made

None.

Changes needed

Connect a Voltmeter to

PicoCore7ULP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
DAC0_OUT	J2_69	J7_21

and **GND** to measure the DAC-output.

Function	ULPBB Rev 1.3
GND	J8_34

Execute binary

Run

```
setenv bootauxfile "dac12_basic.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the demo runs successfully, following information can be seen on the terminal:

```
DAC basic Example.


Please input a value (0 - 4095) to output with DAC:
Input value is 4095
DAC out: 4095
```

fifo_interrupt

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The dac12_fifo_interrupt example shows how to use DAC12 FIFO interrupt.

When the DAC12 FIFO watermark interrupt is enabled firstly, the application would enter the DAC12 ISR immediately, since remaining FIFO data is less than the watermark. Then the FIFO would be feed inside the ISR. Then the DAC12 interrupt could be restrained. Once the DAC12 FIFO is triggered in while loop, the data in FIFO is read out, then it becomes less than the watermark, so the FIFO would be feed again in DAC12 ISR.

With this example, user can define the DAC12 output array to generate the different wave output. Also the software trigger can be called in some timer ISR so that the DAC12 would output the analog signal in indicated period. Or even use the hardware trigger to release the CPU.

Modifications made

None.

Changes needed

Connect a Voltmeter to

PicoCore7ULP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
DAC0_OUT	J2_69	J7_21

and **GND** to measure the DAC-output.

Function	ULPBB Rev 1.3
GND	J8_34

Execute binary

Run

```
setenv bootauxfile "dac12_fifo_interrupt.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the demo runs successfully, following information can be seen on the terminal:

```
Press any key to trigger the DAC...
DAC next output: 0
DAC next output: 100
DAC next output: 200
DAC next output: 300
DAC next output: 400
DAC next output: 500
DAC next output: 600
DAC next output: 700
...
DAC next output: 3100
```

7.3.4 edma

memory_to_memory

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The EDMA memory to memory example is a simple demonstration program that uses the SDK software. It executes one shot transfer from source buffer to destination buffer using the SDK EDMA drivers. The purpose of this example is to show how to use the EDMA and to provide a simple example for debugging and further development.

Modifications made

None.

Changes needed

None.

Execute binary

Run

```
setenv bootauxfile "edma_memory_to_memory.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the example runs successfully, you can see the similar information from the terminal as below.

```
EDMA memory to memory transfer example begin.

Destination Buffer:
0       0       0       0

EDMA memory to memory transfer example finish.

Destination Buffer:
1       2       3       4
```

scatter_gather

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The EDMA scatter_gather example is a simple demonstration program that uses the SDK software. It executes one shot transfer from source buffer to destination buffer using the SDK EDMA drivers. The purpose of this example is to show how to use the EDMA and to provide a simple example for debugging and further development.

Modifications made

None.

Changes needed

None.

Execute binary

Run

```
setenv bootauxfile "edma_scatter_gather.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

If the example runs successfully, you can see similar information as in the terminal below.

```
EDMA scatter_gather transfer example begin.

Destination Buffer:
0        0        0        0        0        0        0        0

EDMA scatter_gather transfer example finish.

Destination Buffer:
1        2        3        4        5        6        7        8
```

7.3.5 ewm

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The EWM (External Watchdog Monitor**)** Example project is to demonstrate usage of the KSDK EWM driver. In the example, EWM counter is continuously refreshed until the button is pressed. Once the button is pressed, EWM counter will expire and interrupt will be generated. After the first pressing, another interrupt can be triggered by pressing button again.

Modifications made

pin_mux.h, pin_mux.h, board.c:

Changed GPIO for the push-button to PTA31

Changes needed

Connect a push-button to

PicoCoreMX7ULP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
Push-button	J3_29	J8_21

and 3.3V.

Function	ULPBB Rev 1.3
3v3	J8_2

Execute binary

Run

```
setenv bootauxfile "ewm.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the example runs successfully, the following message is displayed in the terminal:

```
EWM example
Press the button to expire EWM
EWM interrupt is occurred
Press the button to expire EWM again
```

7.3.6 flexio

i2c

interrupt_lpi2c_transfer

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The flexio_lpi2c_interrupt example shows how to use flexio i2c master driver in the interrupt way: In this example, a flexio simulated i2c master connect to a LPI2C slave.

Modifications made

None.

Changes needed

Please note this application can only run well with RAM link file!

If run it in QSPI flash in place, there's high latency when instruction fetch cache miss. The FlexIO I2C has critical timing requirement that I2C data must be read/write in time, otherwise the state machine works abnormally.

The example requires doing connection between FLEXIO pins and LPI2C pins.

Connect

PicoCoreMX7ULP FlexIO

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
I2C0_SCL	J2_63	J8_11
I2C0_SDA	J2_65	J8_10

to

PicoCoreMX7ULP LP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
I2C3_SCL	J2_49	J8_26
I2C3_SDA	J2_51	J8_24

Execute binary

Run

```
setenv bootauxfile "flexio_i2c_interrupt_lpi2c_transfer.img"; run
.auxcore_tftp
run auxcore
```

to start the example.

Output

When the example runs successfully, the following message is displayed in the terminal:

```
FlexIO I2C interrupt - LPI2C interrupt
Master will send data:
0x00  0x01  0x02  0x03  0x04  0x05  0x06  0x07
0x08  0x09  0x0A  0x0B  0x0C

Slave received data:
0x00  0x01  0x02  0x03  0x04  0x05  0x06  0x07
0x08  0x09  0x0A  0x0B  0x0C
```

Pwm

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

This demo describes how to use SDK drivers to implement the PWM feature by FLEXIO IP module.

It outputs the PWM signal with fixed frequency defined by "DEMO_FLEXIO_FREQUENCY" in source code and dynamic duty from 99 to 1 to one of the FLEXIO pin.

Modifications made

None.

Changes needed

Connect an oscilloscope to

PicoCoreMX7ULP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
PWM_OUT	J2_51	J8_24

Execute binary

Run

```
setenv bootauxfile "flexio_pwm.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the demo runs successfully, you can use oscilloscope probe to touch the PWM_OUT pin to see the waveform.

You may see the duty cycle of the pwm change gradually.

And these messages are displayed/shown on the terminal window:

```
FLEXIO_PWM demo start.
```

spi

edma_lpspi_transfer

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

master

Description

The flexio_spi_master_edma_lpspi_slave example shows how to use flexio spi master driver in edma way: In this example, a flexio simulated master connect to an lpspi slave.

Modifications made

pin_mux.c, flexio_spi_edma_lpspi_transfer_master.c:

Changed FLEXIO_SPI_SIN_PIN 4U to U15.

Changes needed

Please note this application can only run well with RAM link file!

If run it in QSPI flash, there's high latency when instruction cache miss. Although the LPSPI has 4 words FIFO it still cannot adapt to the cache miss latency in slave side. To run LPSPI slave in QSPI flash, either use DMA driver or do synchronization for data exchange.

Connect
PicoCoreMX7ULP_FlexIO

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
CLK	J2_65	J8_10
PCS	J3_31	J8_20
SOUT	J2_63	J8_11
SIN	J3_29	J8_21

To

PicoCoreMX7ULP_LP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
CLK	J3_35	J8_18
PCS	J3_33	J8_19
SOUT	J3_37	J8_7

SIN	J3_39	J8_8

Like this:

FLEXIO_SPI_master	**--**	**LPSPI_slave**
CLK	**--**	**CLK**
PCS	**--**	**PCS**
SOUT	**--**	**SIN**
SIN	**--**	**SOUT**

Execute binary

Run

```
setenv bootauxfile "flexio_spi_edma_lpspi_transfer_master.img";
run .auxcore_tftp

run auxcore
```

to start the example.

Output

When the example runs successfully, you can see the similar information from the terminal as below.

```
FLEXIO Master edma - LPSPI Slave interrupt example start.
This example use one flexio spi as master and one lpspi instance
as slave on one board.
Master uses edma and slave uses interrupt way.
Please make sure you make the correct line connection. Basically,
the connection is:
FLEXIO_SPI_master -- LPSPI_slave
       CLK        --    CLK
       PCS        --    PCS
       SOUT       --    SIN
       SIN        --    SOUT
This is LPSPI slave call back.
FLEXIO SPI master <-> LPSPI slave transfer all data matched!
End of example.
```

slave

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The flexio_spi_slave_edma_lpspi_master example shows how to use flexio spi slave driver in edma way: In this example, a flexio simulated slave connect to an lpspi master.

Modifications made

Please see **edma_lpspi_transfer master**

Changes needed

Please note this application can only run well with RAM link file!

If run it in QSPI flash, there's high latency when instruction cache miss. Although the LPSPI has 4 words FIFO it still cannot adapt to the cache miss latency in slave side. To run LPSPI slave in QSPI flash, either use DMA driver or do synchronization for data exchange.

Please see **edma_lpspi_transfer master**

Execute binary

Run

```
setenv bootauxfile "flexio_spi_edma_lpspi_transfer_slave.img"; run .auxcore_tftp

run auxcore
```

to start the example.

Output

When the example runs successfully, you can see the similar information from the terminal as below.

```
LPSPI Master interrupt - FLEXIO SPI Slave edma example start.

This example use one lpspi instance as master and one flexio spi slave on one board.

Master uses interrupt and slave uses edma way.

Please make sure you make the correct line connection. Basically, the connection is:
```

```
LPSPI_master -- FLEXIO_SPI_slave
   CLK      --     CLK
   PCS      --     PCS
   SOUT     --     SIN
   SIN      --     SOUT
This is FLEXIO SPI slave call back.
LPSPI master <-> FLEXIO SPI slave transfer all data matched!
End of Example.
```

int_lpspi_transfer

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

master

Description

The flexio_spi_master_interrupt_lpspi_slave example shows how to use flexio spi master driver in interrupt way: In this example, a flexio simulated master connect to an lpspi slave.

Modifications made

Please see **edma_lpspi_transfer master**

Changes needed

Please note this application can only run well with RAM link file!

If run it in QSPI flash, there's high latency when instruction cache miss. Although the LPSPI has 4 words FIFO it still cannot adapt to the cache miss latency in slave side. To run LPSPI slave in QSPI flash, either use DMA driver or do synchronization for data exchange.

Please see **edma_lpspi_transfer master**

Execute binary

Run

```
setenv bootauxfile "flexio_spi_int_lpspi_transfer_master.img"; run
.auxcore_tftp

run auxcore
```

to start the example.

Output

When the example runs successfully, you can see the similar information from the terminal as below.

```
FLEXIO Master - LPSPI Slave interrupt example start.

This example use one flexio spi as master and one lpspi instance
as slave on one board.
```

```
Master and slave are both use interrupt way.

Please make sure you make the correct line connection. Basically,
the connection is:

FLEXIO_SPI_master -- LPSPI_slave

       CLK        --    CLK

       PCS        --    PCS

       SOUT       --    SIN

       SIN        --    SOUT

This is LPSPI slave call back.

FLEXIO SPI master <-> LPSPI slave transfer all data matched!
```

slave

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The flexio_spi_slave_interrupt_lpspi_master example shows how to use flexio spi slave driver in interrupt way: In this example, a flexio simulated slave connect to an lpspi master.

Modifications made

Please see **edma_lpspi_transfer master**

Changes needed

Please note this application can only run well with RAM link file!

If run it in QSPI flash, there's high latency when instruction cache miss. Although the LPSPI has 4 words FIFO it still cannot adapt to the cache miss latency in slave side. To run LPSPI slave in QSPI flash, either use DMA driver or do synchronization for data exchange.

Please see **edma_lpspi_transfer master**

Execute binary

Run

```
setenv bootauxfile "flexio_spi_int_lpspi_transfer_slave.img"; run
.auxcore_tftp

run auxcore
```

to start the example.

Output

When the example runs successfully, you can see the similar information from the terminal as below.

```
LPSPI Master interrupt - FLEXIO SPI Slave interrupt example start.

This example use one lpspi instance as master and one flexio spi
slave on one board.

Master and slave are both use interrupt way.
```

```
Please make sure you make the correct line connection. Basically,
the connection is:
LPSPI_master -- FLEXIO_SPI_slave
   CLK       --    CLK
   PCS       --    PCS
   SOUT      --    SIN
   SIN       --    SOUT
This is FLEXIO SPI slave call back.
LPSPI master <-> FLEXIO SPI slave transfer all data matched!
End of Example.FLEXIO SPI master <-> LPSPI slave transfer all data
matched!
```

uart

edma_transfer

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The flexio_uart_edma example shows how to use flexio uart driver in edma way: In this example, a flexio simulated uart connects to a PC through USB-Serial, the board will send back all characters that PC send to the board

Modifications made

None

Changes needed

Please note there's some limitation if running this application in QSPI flash in place.

If run it in QSPI flash, there's high latency when instruction cache miss. The FlexIO UART has no FIFO so it has critical timing requirement that UART data must be read in time, otherwise overflow may occur which causes data loss. So when running in QSPI flash, please don't input more than 8 characters each time.

Attention!

Depending on your USB2COM-Serial 3.3V TTL-cable, it may occur that after a power-reset the TX-connection holds the voltage-level and your board will not boot. To prevent this, just connect your USB2COM-Serial 3.3V TTL-cable only after the U-Boot has booted.

Connect a USB2COM-Serial 3.3V TTL cable to the following Pins:

PicoCoreMX7ULP_FlexIO

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
TX	J2_63	J8_11
RX	J2_65	J8_10

Function	ULPBB Rev 1.3
GND	J8_34

Execute binary

Run

```
setenv bootauxfile "flexio_uart_edma_transfer.img"; run
.auxcore_tftp

run auxcore
```

to start the example.

Output

When the demo runs successfully, the log would be seen on the UART Terminal port which connected to the USB2COM like:

```
Flexio uart edma example

Board receives 8 characters then sends them out

Now please input:
```

int_rb_transfer

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The flexio_uart_interrupt_ring_buffer example shows how to use flexio uart driver in interrupt way with RX ring buffer enabled: In this example, a flexio simulated uart connect to PC through USB-Serial, the board will send back all characters that PC send to the board.

Note: The example echo every 8 characters, so input 8 characters every time.

Modifications made

None

Changes needed

Attention!

Depending on your USB2COM-Serial 3.3V TTL-cable, it may occur that after a power-reset the TX-connection holds the voltage-level and your board will not boot. To prevent this, just connect your USB2COM-Serial 3.3V TTL-cable only after the U-Boot has booted.

Connect a USB2COM-Serial 3.3V TTL cable to the following Pins:

PicoCoreMX7ULP_FlexIO

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
TX	J2_63	J8_11
RX	J2_65	J8_10

Function	ULPBB Rev 1.3
GND	J8_34

Execute binary

Run

```
setenv bootauxfile "flexio_uart_int_rb_transfer.img"; run
.auxcore_tftp

run auxcore
```

to start the example.

Output
When the demo runs successfully, the log would be seen on the UART Terminal port which connected to the USB2COM like:

```
FLEXIO UART RX ring buffer example

Send back received data

Echo every 8 bytes:
```

interrupt_transfer

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The flexio_uart_interrupt example shows how to use flexio uart driver in interrupt way: In this example, a flexio simulated uart connect to PC through USB-Serial, the board will send back all characters that PC send to the board. Note: two queued transfer in this example, so please input even number characters.

Modifications made

None

Changes needed

Please note there's some limitation if running this application in QSPI flash in place.

If run it in QSPI flash, there's high latency when instruction cache miss. The FlexIO UART has no FIFO so it has critical timing requirement that UART data must be read in time, otherwise overflow may occur which causes data loss. So when running in QSPI flash, please don't input more than 8 characters each time.

Attention!

Depending on your USB2COM-Serial 3.3V TTL-cable, it may occur that after a power-reset the TX-connection holds the voltage-level and your board will not boot. To prevent this, just connect your USB2COM-Serial 3.3V TTL-cable only after the U-Boot has booted.

Connect a USB2COM-Serial 3.3V TTL cable to the following Pins:

PicoCoreMX7ULP_FlexIO

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
TX	J2_63	J8_11
RX	J2_65	J8_10

Function	ULPBB Rev 1.3
GND	J8_34

Execute binary

Run

```
setenv bootauxfile "flexio_uart_interrupt_transfer.img"; run
.auxcore_tftp

run auxcore
```

to start the example.

Output
When the demo runs successfully, the log would be seen on the UART Terminal port which connected to the USB2COM like:

```
Flexio uart interrupt example

Board receives 8 characters then sends them out

Now please input:
```

polling_transfer

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The flexio_uart_polling example shows how to use flexio uart driver in polling way: In this example, a flexio simulated uart connect to PC through USB-Serial, the board will send back all characters that PC send to the board.

Modifications made

None

Changes needed

Attention!

Depending on your USB2COM-Serial 3.3V TTL-cable, it may occur that after a power-reset the TX-connection holds the voltage-level and your board will not boot. To prevent this, just connect your USB2COM-Serial 3.3V TTL-cable only after the U-Boot has booted.

Connect a USB2COM-Serial 3.3V TTL cable to the following Pins:

PicoCoreMX7ULP_FlexIO

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
TX	J2_63	J8_11
RX	J2_65	J8_10

Function	ULPBB Rev 1.3
GND	J8_34

Execute binary

Run

```
setenv bootauxfile "flexio_uart_polling_transfer.img"; run
.auxcore_tftp

run auxcore
```

to start the example.

Output

When the demo runs successfully, the log would be seen on the UART Terminal port which connected to the USB2COM like:

```
Flexio uart polling example

Board will send back received characters
```

7.3.7 gpio

input_interrupt

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The GPIO Example project is a demonstration program that uses the KSDK software to manipulate the general-purpose outputs. The example is supported by the set, clear, and toggle write-only registers for each port output data register. The example uses the software button to control/toggle the LED.

Modifications made

pin_mux.h, pin_mux.h, board.c:

Changed GPIO for the push-button to PTA31

Changes needed

Connect a push-button to

PicoCoreMX7ULP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
Push-button	J3_29	J8_21
Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
Led	J2_48	J8_23

connect the button to 3.3V and the Led to GND.

Function	ULPBB Rev 1.3
3v3	J8_2

Function	ULPBB Rev 1.3
GND	J8_34

Execute binary

Run

```
setenv bootauxfile "gpio_input_interrupt.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the example runs successfully, the following message is displayed in the terminal:

```
GPIO Driver example
 Press the push-button to turn on/off a LED
 the push-button is pressed
 the push-button is pressed
 the push-button is pressed
```

led_output

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The GPIO Example project is a demonstration program that uses the KSDK software to manipulate the general-purpose outputs. The example is supported by the set, clear, and toggle write-only registers for each port output data register. The example takes turns to shine the LED.

Modifications made

None

Changes needed

Connect a Led to

PicoCoreMX7ULP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
Led	J2_48	J8_23

and GND.

Function	ULPBB Rev 1.3
GND	J8_34

Execute binary
Run

```
setenv bootauxfile "gpio_led_output.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output
When the example runs successfully, the following message is displayed in the terminal:

```
GPIO Driver example
The LED is blinking.
```

7.3.8 lpadc

single_interrupt

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The lpdc_single_interrupt example shows how to use interrupt with LPADC driver. In this example, the user should indicate a channel to provide a voltage signal (can be controlled by the user) as the LPADC's sample input. When running the project, typing any key into debug console would trigger the conversion. ADC watermark interrupt would be asserted once the number of datawords stored in the ADC Result FIFO is greater than watermark value. In ADC ISR, the watermark flag would be cleared by reading the conversion result value. Also, result information would be printed when the execution return to the main function.

Modifications made

None

Changes needed

Connect a voltage signal (max 1v8)

Function	ULPBB Rev 1.3
1v8	J8_1

to

PicoCoreMX7ULP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
ADC1_CH6A	J2_42	J8_27

and 3.3V.

Function	ULPBB Rev 1.3
3v3	J8_2

Execute binary

Run

```
setenv bootauxfile "lpadc_single_interrupt.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the example runs successfully, the following message is displayed in the terminal:

```
LPADC Interrupt Example
Please press any key to get user channel's ADC value.
ADC value: 2714
```

single_polling

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The lpadc_single_polling example shows the simplest way to use LPADC driver. In this example, the user should indicate a channel to provide a voltage signal (can be controlled by user) as the LPADC's sample input. When running the project, typing any key into debug console would trigger the conversion. The execution would check the FIFO valid flag in loop until the flag is asserted, which means the conversion is completed. Then read the conversion result value and print it to debug console.

Note, the default setting of initialization for the ADC converter is just an available configuration. User can change the configuration structure's setting in application to fit the special requirement.

Modifications made

None

Changes needed

Connect a voltage signal (max 1v8)

Function	ULPBB Rev 1.3
1v8	J8_1

to

PicoCoreMX7ULP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
ADC1_CH6A	J2_42	J8_27

Function	ULPBB Rev 1.3
3v3	J8_2

Execute binary
Run

```
setenv bootauxfile "lpadc_single_polling.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output
When the example runs successfully, the following message is displayed in the terminal:

```
LPADC Polling Example
Please press any key to get user channel's ADC value.
ADC value: 2714
```

7.3.9 lpit

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The LPIT project is a simple example of the SDK LPIT driver. It sets up the LPIT hardware block to trigger a periodic interrupt after every 1 second. When the LPIT interrupt is triggered a message is printed on the serial terminal. To toggle a LED on the board the function must be implemented.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "lpit.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the example runs successfully, the following message is displayed in the terminal:

```
Starting channel No.0 ...
 Channel No.0 interrupt is occurred !
 Channel No.0 interrupt is occurred !
 ……
```

7.3.10 lptmr

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The LPTMR (low power timer) project is a simple demonstration program of the SDK LPTMR driver. It sets up the LPTMR hardware block to trigger a periodic interrupt after every 1 second. When the LPTMR interrupt is triggered a message a printed on the UART terminal. To toggle a LED on the board, the function must be implemented.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "lptmr.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the example runs successfully, the following message is displayed in the terminal:

```
Low Power Timer Example
LPTMR interrupt No.1
LPTMR interrupt No.2
LPTMR interrupt No.3
......
```

7.3.11 ltc

aes

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

This project is a demonstration program that uses the KSDK software for encryption/decryption sample data using AES-CBC, AES-CCM, and XCBC-MAC algorithm.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "ltc_aes.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the example runs successfully, the following message is displayed in the terminal:

```
Testing input string:
          Once upon a midnight dreary…,

--------------------------- AES-CBC method ------------------------
AES CBC Encryption of 320 bytes.
AES CBC encryption finished.

AES CBC Decryption of 320 bytes.
AES CBC decryption finished.
Decrypted string :
          Once upon a midnight dreary…,
-------------------------- AES-CCM method ------------------------
AES CCM Encryption of 320 bytes.
    using iv length  : 12 bytes
    using aad length : 20 bytes
    using key length : 16 bytes
    using tag length : 8 bytes
AES CCM encryption finished.

AES CCM decryption of 320 bytes.
AES CCM decryption finished.
Decrypted string:
          Once upon a midnight dreary…

----------------------- AES-XCBC-MAC -----------------------------
AES XCBC-MAC Computing hash of 320 bytes
Computed hash:
9e c 5a 5a 10 bb 96 67 9f 98 3 29 94 f8 6f 9f
.........THE END OF THE LTC (AES) DRIVER EXAMPLE ........
```

aes_edma

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

This project is a demonstration program that uses the KSDK software for encryption/decryption sample data using EDMA for AES-CBC algorithm.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "ltc_aes_edma.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the example runs successfully, the following message is displayed in the terminal:

```
................ LTC (AES EDMA) DRIVER EXAMPLE ................


Testing input string:
          Once upon a midnight dreary,…


------------------------- AES-CBC method -------------------------
-------------
AES CBC Encryption of 320 bytes.
AES CBC encryption finished.


AES CBC Decryption of 320 bytes.
AES CBC decryption finished.
```

```
Decrypted string :
          Once upon a midnight dreary,…
....... THE END OF THE LTC (AES EDMA) DRIVER EXAMPLE .......
```

7.3.12 pf1550

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The pf1550 driver example demonstrates the usage of pf1550 SDK component driver. The example shows the usage of PF1550 API to:

1. Set regulator output voltages;

2. Query regulator output voltages;

3. Dump PF1550 register content;

4. Charge a Li-on Battery Cell. (Option only usable if a battery cell is connected to the board)

To use this example, the user needs to pay attention to the output voltage while updating the regulator output, and make sure the output voltage is set so it can’t cause hardware damage; you also need to pay attention to the charging voltage setting when charging the Li-on battery cell and make sure that the battery is not over-charged.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "pf1550.img"; run .auxcore_tftp

run auxcore
```

to start the example.

Output

The log below shows the output of the pf1550 example in the terminal window:

```
-------------- PF1550 on board PMIC driver example --------------


Please select the PMIC example you want to run:
[1]. Setting Regulator Output Voltage
[2]. Dumping Regulator Output Voltage
[3]. Dumping Selected Register Content
[4]. Charge Li-on Battery


User can press the number key to try the corresponding function of
the demo.


Note
In order to protect the i.MX 7ULP from over-voltage, it is recom-
mended to setting regulator's output voltage with following re-
straint:
Buck Switch1's range: 800mV ~ 1100mV;
Buck Switch2's range: Fixed to 1200mV;
Buck Switch3's range: Fixed to 1800mV;
LDO1's range        : 3000mV to 3300mV;
LDO2's range        : Fixed to 3300mV;
LDO3's range        : Fixed to 1800mV.
The Setting Regulator Output Voltage function of this example is
used to demonstrate the usage of SDK PF1550 bare-bone driver, user
need to adjust the regulator output according to specific board
design.
```

7.3.13 sai

edma_transfer

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The sai_edma_transfer example shows how to use sai driver with EDMA: In this example, one sai instance playbacks the audio data stored in flash/SRAM using EDMA channel.

Modifications made

F&S uses a different codec.

Changes needed

Connect headphones to the audio jack.

Execute binary

Run

```
setenv bootauxfile "sai_edma_transfer.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the demo runs successfully, you can hear the tone and the log would be seen on the terminal like:

```
 ~~~~~~~~~~~~~~~~~~~
 SAI example started!
 default

 SAI EDMA example finished!
 ~~~~~~~~~~~~~~~~~~~
```

Interrupt

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The sai_interrupt example shows how to use sai functional API to implement interrupt playback: In this example, one sai instance playbacks the audio data stored in flash/SRAM using interrupt.

Modifications made

F&S uses a different codec.

Changes needed

Connect headphones to the audio jack.

Execute binary

Run

```
setenv bootauxfile "sai_interrupt.img"; run .auxcore_tftp

run auxcore
```

to start the example.

Output

When the demo runs successfully, you can hear the tone and the log would be seen on the terminal like:

```
~~~~~~~~~~~~~~~~~~~

 SAI functional interrupt example started!

 default


 SAI functional interrupt example finished!

 ~~~~~~~~~~~~~~~~~~~~
```

Interrupt_transfer

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The sai_interrupt_transfer example shows how to use sai driver with interrupt: In this example, one sai instance playbacks the audio data stored in flash/SRAM using interrupt.

Modifications made

F&S uses a different codec.

Changes needed

Connect headphones to the audio jack.

Execute binary

Run

```
setenv bootauxfile "sai_interrupt_transfer.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

When the demo runs successfully, you can hear the tone and the log would be seen on the terminal like:

```
~~~~~~~~~~~~~~~~~~~
 SAI example started!
 default


 SAI example finished!
 ~~~~~~~~~~~~~~~~~~~
```

7.3.15 sema42

U-Boot

Attention!

This example is solely intended to be used in the u-boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The sema42 U-Boot example shows how to use SEMA42 driver to lock and unlock a sema gate. This example should work together with U-Boot. This example runs on Cortex-M core, the U-Boot runs on the Cortex-A core.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "sema42_uboot.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

The log below in the Cortex-M terminal window shows the commands to use in the U-Boot:

```
***************************************************************
* Please make sure the uboot is started now.                  *
* Use the following commands in uboot for SEMA42 gate access *
* - md.b 0x4101b003 1 : Get SEMA42 gate status.               *
*   - 0 - Unlocked;                                           *
*   - 2 - Locked by Cortex-A;                                 *
*   - 3 - Locked by Cortex-M;                                 *
* - mw.b 0x4101b003 2 1 : Lock the SEMA42 gate.               *
* - mw.b 0x4101b003 0 1 : Unlock the SEMA42 gate.             *
***************************************************************
```

```
Press any key to start the example...


SEMA42 example started!


Now the SEMA42 gate is unlocked, checking status in uboot returns
0. Press any key to lock the SEMA42 gate...


Now the SEMA42 gate is locked, checking status in uboot returns 3.
Lock or unlock the SEMA42 gate in uboot, the status does not
change.


Press any key to unlock the SEMA42 gate...


Now the SEMA42 gate is unlocked, checking status in uboot returns
0.


Lock the SEMA42 gate in uboot, after locked, then press any key...


Cortex-A has locked the SEMA42 gate in uboot, Cortex-M could not
lock.


Press any key to reset the SEMA42 gate...


Now the SEMA42 gate is unlocked, checking status in uboot returns
0.


Press any key to finish the example...


SEMA42 uboot example succeeded.
```

7.3.16 snvs

snvs_hp_rtc

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The SNVS HP RTC project is a simple demonstration program of the SDK SNVS HP driver. The test will set up RTC date and time to a predefined value and starts the counter. RTC then triggers an alarm after a user specified time period. The alarm will be set with reference to this predefined date and time.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "snvs_hp_rtc.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

```
SNVS HP example:
Set up time to wake up an alarm.
Current datetime: 2014-12-25 19:00:00
Please input the number of second to wait for alarm and press en-
ter
The second must be positive value
16
Alarm will occur at: 2014-12-25 19:00:16


 Alarm occurs !!!! Current datetime: 2014-12-25 19:00:16
```

```
Please input the number of second to wait for alarm and press en-
ter
The second must be positive value
```

snvs_lp_rtc

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The SNVS LP SRTC project is a simple demonstration program of the SDK SNVS LP driver. The test will set up secure RTC (SRTC) date and time to a predefined value and starts the counter, then the SRTC counter value is synchronized with non-secure RTC (RTC). RTC then triggers an alarm after a user specified time period. The alarm will be set with reference to this predefined date and time.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "snvs_lp_srtc.img"; run .auxcore_tftp

run auxcore
```

to start the example.

Output

```
SNVS LP SRTC example:

RTC date and time has been synchronized with SRTC

Set up time to wake up an alarm.

Current datetime: 2014-12-25 19:00:00

Please input the number of second to wait for alarm and press en-
ter

The second must be positive value

16

Alarm will occur at: 2014-12-25 19:00:16


 Alarm occurs !!!! Current datetime: 2014-12-25 19:00:16

Please input the number of second to wait for alarm and press en-
ter
```

```
The second must be positive value
```

7.3.17 trgmux

lptmr_trigger_lpit

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The lptmr_trigger_lpit project is a simple example of the SDK LPIT,LPTMR and TRGMUX driver which shows how to use the LPTMR and TRGMUX to generate a LPIT trigger. It sets up the LPTMR hardware block to generate the trigger every 0.5 second, and the LPIT counter will load on the first trigger rising edge and then decrement down to zero on each trigger rising edge. A message would be printed on the serial terminal.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "trgmux_lptmr_trigger_lpit.img"; run
.auxcore_tftp

run auxcore
```

to start the example.

Output

```
Example Starts!

LPIT interrupt No.1

LPIT interrupt No.2

LPIT interrupt No.3

LPIT interrupt No.4

LPIT interrupt No.5

LPIT interrupt No.6

LPIT interrupt No.7

LPIT interrupt No.8

......
```

7.3.18 trng

random

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The True Random Number Generator (TRNG) is a hardware accelerator module that generates a 512-bit entropy as needed by an entropy consuming module or by other post processing functions. The TRNG example project is a demonstration program that uses the KSDK software to generate random numbers and prints them to the terminal.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "trng_random.img"; run .auxcore_tftp
run auxcore
```

to start the example.

Output

```
TRNG Peripheral Driver Example
Generate 10 random numbers:
Random[0] = 0xE4C973F5
Random[1] = 0x25BEBC2B
Random[2] = 0x1A889794
Random[3] = 0xF723958
Random[4] = 0xD9818CFE
Random[5] = 0x409950E7
Random[6] = 0xA9315CA1
Random[7] = 0x5060CAC3
......
```

7.3.19 tstmr

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The tstmr example shows the usage of TSTMR driver in application. The TSTMR module is a free running incrementing counter that starts running after system reset de-assertion and can be read at any time by the software for determining the software ticks. The TSTMR runs off the 1 MHz clock and resets on every system reset. In this example, it would output a time stamp information when the application is ready. And then, delay for 1 second with TSTMR_DelayUs() function. Before and after the delay, it would output the two time stamps information again.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "tstmr.img"; run .auxcore_tftp

run auxcore
```

to start the example.

Output

When the demo runs successfully, will get the similar messages on the terminal.

```
 Timestamp1 = 01c98d6


 Test the delay function, delay for 1 second


 Start time = 01cb123


 End time = 02bfbd3
```

7.3.20 xrdc

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The xrdc example shows how to control the memory and peripheral access policy using XRDC. In this example, one peripheral and a memory region are set inaccessible, then the hard fault occurs.

Modifications made

None

Changes needed

Please note the XRDC can't access the MRGD register unless the related memory region clock has been enabled!

Execute binary

Run

```
setenv bootauxfile "xrdc.img"; run .auxcore_tftp
run auxcore
```

Output

```
XRDC example start
Set the peripheral not accessible
Violent access at address: 0x41025000
The peripheral is accessible now
Set the memory not accessible
Violent access at address: 0x2F020000
The memory is accessible now
XRDC example Success
```

7.4 mmcau_examples

Memory-Mapped Cryptographic Acceleration Unit (MMCAU).

7.4.1 mmcau_api

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

This project is a demonstration program that uses the KSDK software for encryption/decryption sample data using AES-CBC, DES3-CBC and Hash algorithms MD5, SHA1 and SHA256.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "mmcau.img"; run .auxcore_tftp
run auxcore
```

Output

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
............................. MMCAU DRIVER EXAMPLE
.............................


Testing input string:
          Once upon a midnight dreary,. . .
----------------------------------- AES-CBC method ---------------
-----------------------
AES CBC Encryption of 320 bytes.
AES CBC encryption finished. Speed 1.554970 MB/s.
```

```
AES CBC Decryption of 320 bytes.
AES CBC decryption finished. Speed 1.613277 MB/s.
Decrypted string :
          Once upon a midnight dreary, . . .
----------------------------------- DES3-CBC method --------------
------------------------
DES3 CBC Encryption of 320 bytes.
DES3 CBC encryption finished. Speed 0.738141 MB/s.

DES3 CBC decryption of 320 bytes.
DES3 CBC decryption finished. Speed 0.746226 MB/s.
Decrypted string :
          Once upon a midnight dreary, . . .
--------------------------------------- HASH ---------------------
---------------------
Computing hash of 64 bytes.
Input string:
          The quick brown fox jumps over the lazy dog

Computed SHA1 at speed 2.992902 MB/s:
2fd4e1c67a2d28fced849ee1bb76e7391b93eb12

Computed SHA256 at speed 1.943140 MB/s:
d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592

Computed MD5 at speed 5.926416 MB/s:
9e107d9d372bb6826bd81d3542a419d6

.............. THE END OF THE MMCAU DRIVER EXAMPLE
................................
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

7.5 mulitcore_examples

7.5.1 erpc_matrix_multiply

Description

The Multicore eRPC Matrix Multiply RTOS project is a simple demonstration program that uses the MCUXpresso SDK software and the Multicore SDK to show how to implement the Remote Procedure Call between cores of the multicore system. The primary core (eRPC client) releases the secondary core (eRPC server) from the reset and then the erpcMatrixMultiply() eRPC call is issued to let the secondary core to perform the multiplication of two randomly generated matrices. The original matrices and the result matrix is printed out to the serial console by the primary core. RPMsg-Lite erpc transport layer is used in this example application.

Shared memory usage:

This multicore example uses the shared memory for data exchange. The shared memory region is defined and the size can be adjustable in the linker file. The shared memory region start address and the size have to be defined in linker file for each core equally. The shared memory start address is then exported from the linker to the application.

eRPC documentation
eRPC specific files are stored in: middleware\multicore_<version>\erpc
eRPC documentation is stored in: middleware\multicore_<version>\erpc\doc
eRPC is open-source project stored on github: https://github.com/EmbeddedRPC/erpc
eRPC documentation can be also found in: http://embeddedrpc.github.io

Modifications made

None

Changes needed

1. Make sure you activated the python_erpc package in your buildroot config.
2. Copy python folder containing the Linux part of the eRPC multicore demo from the `examples/fsimx7ulp/mulitocre_examples/erpc_matrix_mulitply_`rtos folder to your root file system

Execute binary

1 .Run

```
setenv bootauxfile "erpc_matrix_multiply_rpmsg_rtos_imxcm4.img";
run .auxcore_tftp

boot
```

2. After login, make sure imx_rpmsg_tty kernel module is inserted (lsmod) or insert it

```
modprobe imx_rpmsg_tty
```

3. In Linux console, browse to the python directory containing `example.py`

4. Figure out the rpmsg tty device, i.e. `/dev/ttyRPMSG101` and replace "`/dev/ttyRPMSG`" in example.py with your device i.e.:
transport = erpc.transport.SerialTransport('/dev/ttyRPMSG101', 115200)
5. Run

```
python example.py /dev/ttyRPMSG101
```

6. Press any key to generate a random 5x5 matrix and send a multiplication request to Cortex-M4.

Output
The log below shows the output of the eRPC Matrix Multiply RTOS demo in the Cortex-A terminal window after run example.py:

```
Selected ttyRPMSG Transport
Matrix #1
  21   33   37   37    9
  23   45   43    0   32
  38   44    8   15   36
  18   18   38   44   16
  22   23    0   38    7

Matrix #2
  11   23   27   45   11
   7   19   23   24    6
  32   26   49   43   16
  22   48   36   34   41
  27   20   32   31   11

eRPC request is sent to the server

Result matrix
2703 4028 4759 4865 2637
2808 3142 4787 4956 1563
2284 3358 4122 4736 1821
2940 4176 4858 4868 2894
1428 2907 2715 3051 2015
```

7.5.2 rpmsg_lite_pingpong_rtos

Description

The Multicore RPMsg-Lite pingpong RTOS project is a simple demonstration program that uses the MCUXpresso SDK software and the RPMsg-Lite library and shows how to implement the inter-core communication between cores of the multicore system. The primary core releases the secondary core from the reset and then the inter-core communication is established. Once the RPMsg is initialized and endpoints are created the message exchange starts, incrementing a virtual counter that is part of the message payload. The message pingpong finishes when the counter reaches the value of 100. Then the RPMsg-Lite is deinitialized and the procedure of the data exchange is repeated again.

Shared memory usage

This multicore example uses the shared memory for data exchange. The shared memory region is defined and the size can be adjustable in the linker file. The shared memory region start address and the size have to be defined in linker file for each core equally. The shared memory start address is then exported from the linker to the application.

Modifications made

None

Changes needed

None

Execute binary

First run

```
setenv bootauxfile "rpmsg_lite_pingpong_rtos_imxcm4.img"; run
.auxcore_tftp

boot
```

then wait for Linux OS to finish booting. Log in, and type

```
modprobe imx_rpmsg_pingpong
```

to load the pingpong Linux side module.

Output

```
After the Linux RPMsg pingpong module was installed, the ARM Cor-
tex-M4 terminal displays the following information:

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Sending pong...

Waiting for ping...

Sending pong...
```

```
Waiting for ping...
Sending pong...
......
Waiting for ping...
Sending pong...
Ping pong done, deinitializing...
Looping forever...
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
The Cortex-A terminal displays the following information:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
get 1 (src: 0x1e)
get 3 (src: 0x1e)
......
get 99 (src: 0x1e)
get 101 (src: 0x1e)
```

7.5.3 rpmsg_lite_str_echo_rtos

Description

The Multicore RPMsg-Lite string echo project is a simple demonstration program that uses the MCUXpresso SDK software and the RPMsg-Lite library and shows how to implement the inter-core communication between cores of the multicore system.

It works with Linux RPMsg master peer to transfer string content back and forth. The name service handshake is performed first to create the communication channels. Next, Linux OS waits for user input to the RPMsg virtual tty. Anything which is received is sent to M4. M4 displays what is received, and echoes back the same message as an acknowledgement. The tty reader on the Linux side can get the message, and start another transaction. The demo demonstrates RPMsg's ability to send arbitrary content back and forth. Note: The maximum message length supported by RPMsg is now 496 bytes. String longer than 496 will be divided by virtual tty into several messages.

Shared memory usage

This multicore example uses the shared memory for data exchange. The shared memory region is defined and the size can be adjustable in the linker file. The shared memory region start address and the size have to be defined in linker file for each core equally. The shared memory start address is then exported from the linker to the application.

Modifications made

None

Changes needed

None

Execute binary

First run

```
setenv bootauxfile "rpmsg_lite_str_echo_rtos_imxcm4.img"; run
.auxcore_tftp

boot
```

then wait for Linux OS to finish booting. Log in, and type

```
modprobe imx_rpmsg_tty
```

to load the pingpong Linux side module.

Run

```
echo test > /dev/ttyRPMSG<num>
```

<num> here is the allocated ttyRPMsg channel number. Please find out the number in the file system by "ls" command.

Output

```
RPMSG String Echo FreeRTOS RTOS API Demo...
Nameservice sent, ready for incoming messages...
```

After the Linux RPMsg tty module was installed, the ARM Cortex-M4 terminal displays the following information:

```
Get Messgae From Master Side : "hello world!" [len : 12]
```

After the user write into the ttyRPMSG –device the Cortex-M4 terminal displays the following information:

```
Get Message From Master Side : "test" [len : 4]
Get New Line From Master Side
```

rtos_examples

7.5.4 freertos_event

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

This example shows how a task waits for an event (defined set of bits in event group). This event can be set by any other process or interrupt in the system. The example application creates three tasks. Two write tasks (write_task_1 and write_task_2) continuously setting event bit 0 and bit 1. A Read_task is waiting for any event bit and printing actual state on console. Event bits are automatically cleared after read task is entered.

Three possible states can occur:

Both bits are set.

Bit B0 is set.

Bit B1 is set.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "freertos_event.img"; run .auxcore_tftp
run auxcore
```

Output

```
Bit B1 is set.
Bit B0 is set.
Bit B1 is set.
Bit B0 is set.
Bit B1 is set.
. . .
```

7.5.5 freertos_generic

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

This example is based on code FreeRTOS documentation from

http://www.freertos.org/Hardware-independent-RTOS-example.html.

It shows combination of several tasks with queue, software timer, tick hook and semaphore. The example application creates three tasks. The prvQueueSendTask is periodically sending data to xQueue queue. The prvQueueReceiveTask is waiting for incoming message and counting the number of received messages. Task prvEventSemaphoreTask is waiting for xEventSemaphore semaphore given from vApplicationTickHook. The tick hook gives a semaphore every 500ms. The other hook types used for RTOS and resource statistics are also demonstrated in this example:

* vApplicationIdleHook

* vApplicationStackOverflowHook

* vApplicationMallocFailedHook

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "freertos_generic.img"; run .auxcore_tftp
run auxcore
```

Output

```
Event task is running.
Receive message counter: 1.
Receive message counter: 2.
Receive message counter: 3.
Receive message counter: 4.
Receive message counter: 5.
Receive message counter: 6.
Receive message counter: 7.
. . .
```

7.5.6 freertos_hello

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The Hello World project is a simple demonstration program that uses the SDK UART driver in combination with FreeRTOS. The purpose of this demo is to show how to use the debug console and to provide a simple project for debugging and further development. The example application creates one task called hello_task. This task prints "Hello world." Message via debug console utility and suspend itself.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "freertos_hello.img"; run .auxcore_tftp

run auxcore
```

Output

```
Hello world.
```

7.5.7 freertos_lpi2c

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The LPI2C Example project is a demonstration program that uses the KSDK software to manipulate the Low Power Inter-Integrated Circuit. The example uses two instances of LPI2C, one in configured as master and the other one as slave. The LPI2C master sends data to LPI2C slave. The slave will check the data it receives and shows the log.

Modifications made

None

Changes needed

Connect

PicoCoreMX7ULP FlexIO

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
I2C0_SCL	J2_63	J8_11
I2C0_SDA	J2_65	J8_10

To

PicoCoreMX7ULP LP

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
I2C3_SCL	J2_49	J8_26
I2C3_SDA	J2_51	J8_24

Execute binary

Run

```
setenv bootauxfile "freertos_lpi2c.img"; run .auxcore_tftp
run auxcore
```

Output

```
LPI2C example -- MasterInterrupt_SlaveInterrupt.
Master will send data :
0x 0  0x 1  0x 2  0x 3  0x 4  0x 5  0x 6  0x 7
0x 8  0x 9  0x a  0x b  0x c  0x d  0x e  0x f
0x10  0x11  0x12  0x13  0x14  0x15  0x16  0x17
0x18  0x19  0x1a  0x1b  0x1c  0x1d  0x1e  0x1f

I2C master transfer completed successfully.
I2C slave transfer completed successfully.
 Transfer successfully!
Slave received data :
0x 0  0x 1  0x 2  0x 3  0x 4  0x 5  0x 6  0x 7
0x 8  0x 9  0x a  0x b  0x c  0x d  0x e  0x f
0x10  0x11  0x12  0x13  0x14  0x15  0x16  0x17
0x18  0x19  0x1a  0x1b  0x1c  0x1d  0x1e  0x1f.
```

7.5.8 freertos_lpuart

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

The LPUART example for FreeRTOS demonstrates the possibility to use the LPUART driver in the RTOS. The example uses single instance of LPUART IP and writes string into, then reads back chars. After every 4 Bits (characters) received, the resulting string is printed on the cortex-M terminal.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "freertos_lpuart.img"; run .auxcore_tftp
run auxcore
```

Output

```
FreeRTOS LPUART driver example!
xxxx
```

7.5.9 freertos_mutex

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

This example shows how a mutex manages access to a common resource (terminal output). The example application creates two identical instances of write_task. Each task will lock the mutex before printing and unlock it after printing to ensure that the outputs from tasks are not mixed together. The test_task accept output message during creation as function parameter. Output message have two parts. If xMutex is unlocked, the write_task_1 acquires xMutex and prints the first part of message. Then rescheduling is performed. In this moment the scheduler checks if some other task could run, but second task write_task+_2 is blocked because xMutex is already locked by first write task. The first write_task_1 continues from the last point by printing of the second message part. Finally the xMutex is unlocked and the second instance of write_task_2 is executed.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "freertos_mutex.img"; run .auxcore_tftp
run auxcore
```

Output

```
"ABCD | EFGH"
"1234 | 5678"
"ABCD | EFGH"
"1234 | 5678"
```

7.5.10 freertos_queue

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

This example introduces a simple logging mechanism based on message passing. It is divid-ed in two parts: The first part is a logger. It contains three tasks:

log_add().....Adds new message into the log. Calls xQueueSend function to pass a new message into the message queue.

log_init()....Initializes logger (create logging task and message queue log_queue).

log_task()....Task responsible for printing of log output.

The second part is an application of this simple logging mechanism. Each of two tasks (write_task_1 and write_task_2) print 5 messages into the log.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "freertos_queue.img"; run .auxcore_tftp
run auxcore
```

Output

```
Log 0: Task1 Message 0
Log 1: Task2 Message 0
Log 2: Task1 Message 1
Log 3: Task2 Message 1
. . .
Log9:  Task2 Message 4
```

7.5.11 freertos_sem

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

This example shows how semaphores works. Two different tasks are synchronized in bilateral rendezvous model. The example uses four tasks. One producer_task and three consumer_tasks. The producer_task starts by creating two semaphores (xSemaphore_producer and xSemaphore_consumer). These semaphores control the access to a virtual item. The synchronization is based on bilateral rendezvous pattern. Both of consumer and producer must be prepared to enable transaction.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "freertos_sem.img"; run .auxcore_tftp
run auxcore
```

Output

```
Producer_task created.
Consumer_task 0 created.
Consumer_task 1 created.
Consumer_task 2 created.
Consumer number: 0
Consumer 0 accepted item.
Consumer number: 1
Consumer number: 2
Producer released item.
Consumer 0 accepted item.
Producer released item.
```

```
Consumer 1 accepted item.
Producer released item.
Consumer 2 accepted item.
. . .
```

7.5.12 freertos_swtimer

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

This example shows usage of software timer and its callback. The example application creates one software timer SwTimer. The timer's callback SwTimerCallback is periodically executed and the text "Tick." is printed to the terminal.

Modifications made

None

Changes needed

None

Execute binary

Run

```
setenv bootauxfile "freertos_swtimer.img"; run .auxcore_tftp
run auxcore
```

Output

```
Tick.
Tick.
Tick.
. . .
```

7.5.13 freertos_tickless

Attention!

This example is solely intended to be used in the U-Boot on the A7, if Linux is booting it will freeze. If you still consider using it for Linux you will have to do some modifications on your own risk. The changes needed are described on page 27.

Description

This example shows how the CPU enters sleep mode and will be waking up by an expired time delay that uses the GPT module or by external interrupt caused by a user defined button.

Modifications made

pin_mux.h, pin_mux.h, board.c:

Changed GPIO for the push-button to PTA31

Changes needed

Connect a push-button to

PicoCoreMX7ULP Execute binary

Function	PCOREMX7ULP Rev 1.2	ULPBB Rev 1.3
Push-button	J3_29	J8_21

and 3.3V.

Function	ULPBB Rev 1.3
3v3	J8_2

Run

```
setenv bootauxfile "freertos_tickless.img"; run .auxcore_tftp
run auxcore
```

Output

```
Press any key to start the example
Tickless Demo example
Press or turn on the push-button to wake up the CPU
```

```
Tick count :
0
5000
CPU woken up by external interrupt
10000
CPU woken up by external interrupt
15000
. . .
```

8 Appendix

List of Figures

List of Tables

Third Party Agreement from Real Time Engineers Ltd.

Important Notice