Inter-Processor Communication (IPC) for AM64x processors

Thank you for joining! We will begin the webinar soon.

Agenda

- IPC Overview
- Drivers: IPC Notify, IPC RPMsg, Linux RPMsg
- How to load remote cores from Linux
- How to build the IPC examples
- How to run the Linux <-> R5 IPC Demo

Useful links for each slide will be displayed at the bottom of the screen

What is IPC?

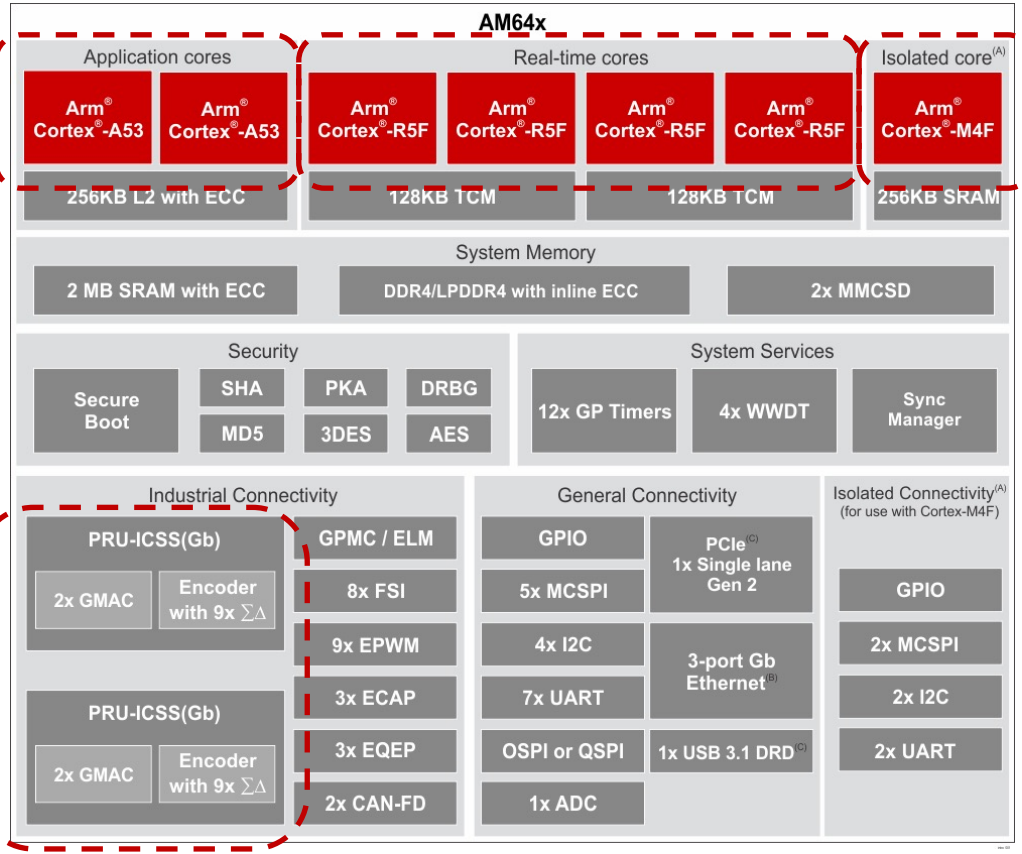
- IPC stands for Inter-Processor Communication
- The AM64x is a flexible device with many possible IPC implementations. In this webinar, we will discuss one specific IPC implementation that TI provides in our SDKs (IPC Notify, IPC RPMsg, Linux RPMsg)

What cores are involved in IPC?

R5F cores

A53 cores

M4F core



Each PRU_ICSSG instance has:
2 PRUs
2 RTUs
2 TX_PRUs

IPC on A53

- A53 Linux SDK 7.3 onwards: Linux RPMsg
- A53 FreeRTOS / NORTOS:
  - FreeRTOS / NORTOS on A53 (including IPC Notify and IPC RPMsg) is still under development
  - A53 FreeRTOS / NORTOS is a non-supported, experimental feature in MCU+ SDK 8.1

https://software-dl.ti.com/processor-sdk-linux/esd/docs/08_00_00_21/linux/Foundational_Components_IPC64x.html

https://software-dl.ti.com/mcu-plus-sdk/esd/AM64X/08_01_00_36/exports/docs/api_guide_am64x/RELEASE_NOTES_08_01_00_PAGE.html#EXPERIMENTAL_FEATURES

IPC on R5F

- R5F MCU+ SDK 7.3 onwards: IPC Notify, IPC RPMsg
  - Supported on both FreeRTOS and NORTOS

https://software-dl.ti.com/mcu-plus-sdk/esd/AM64X/08_01_00_36/exports/docs/api_guide_am64x/IPC_GUIDE.html

IPC on M4F

- M4F MCU+ SDK 8.1 onwards: IPC Notify, IPC RPMsg
  - Supported on both FreeRTOS and NORTOS

https://software-dl.ti.com/mcu-plus-sdk/esd/AM64X/08_01_00_36/exports/docs/api_guide_am64x/IPC_GUIDE.html

IPC on PRU_ICSSG

- A53 Linux SDK 8.1 onwards: PRU RPMsg
- R5F MCU+ SDK 7.3 onwards: PRUICSS driver (interrupts & memory writes)
- An example of IPC between two cores within PRU_ICSSG can be found in PRU Software Support Package (PSSP) v6.0.0 onwards (PRU_Direct_ConnectX)
- IPC with PRU_ICSSG cores is different than IPC with R5F/M4F cores. AM64x mailboxes do not have outputs to the PRU_ICSSG. So PRU IPC is based on interrupts instead of mailboxes

https://software-dl.ti.com/mcu-plus-sdk/esd/AM64X/08_01_00_36/exports/docs/api_guide_am64x/DRIVERS_PRUICSS_PAGE.html

https://git.ti.com/cgit/pru-software-support-package/pru-software-support-package/
Navigate to examples/am64x/PRU_Direct_ConnectX

IPC Notify

- Used on R5F & M4F
- Low latency IPC. Uses mailboxes
  - Think of a mailbox like an interrupt signal paired with a 32 bit register. For more information, reference the AM64x Technical Reference Manual, Interprocessor Communication > Mailbox
- FreeRTOS / NORTOS to FreeRTOS / NORTOS

https://software-dl.ti.com/mcu-plus-sdk/esd/AM64X/08_01_00_36/exports/docs/api_guide_am64x/DRIVERS_IPC_NOTIFY_PAGE.html

IPC RPMsg

- Used on R5F & M4F
- Uses IPC Notify for interrupts, uses shared memory (VRING) for message buffers
- FreeRTOS / NORTOS to FreeRTOS / NORTOS:
  - Message size & number of message buffers are configurable. Shared memory can be in DDR or internal memory
  - To pass large messages of data, consider using IPC Notify to pass an offset to a shared data location between cores (or IPC RPMsg to pass the full address)
- FreeRTOS / NORTOS to Linux:
  - Message size & number of message buffers are fixed. Shared memory is in DDR

https://software-dl.ti.com/mcu-plus-sdk/esd/AM64X/08_01_00_36/exports/docs/api_guide_am64x/DRIVERS_IPC_RPMESSAGE_PAGE.html

Linux RPMsg

- Used on Linux A53
- Designed for ease of use rather than optimizing latency or throughput
  - Shared memory in DDR rather than internal memory
  - Fixed RPMsg size
  - Several copies required to get data from Linux Userspace to FreeRTOS / NORTOS core
- Linux to FreeRTOS / NORTOS
- Different driver from PRU RPMsg

https://software-dl.ti.com/processor-sdk-linux/esd/docs/08_00_00_21/linux/Foundational_Components_IPC64x.html

Linux RPMsg from Userspace / kernel space

- The Linux RPMsg driver can enable RPMsg from Linux Userspace (e.g., Linux applications), or from Linux kernel space (e.g., Linux drivers)
- Example Linux userspace application is at https://git.ti.com/cgit/rpmsg/ti-rpmsg-char/tree/examples
- Example Linux driver is in the AM64x Linux Processor SDK under samples/rpmsg/rpmsg_client_sample.c

https://software-dl.ti.com/processor-sdk-linux/esd/docs/08_00_00_21/linux/Foundational_Components_IPC64x.html

How to load remote cores during Linux boot

- As long as the R5F cores are not disabled in the Linux device tree, then the Linux RemoteProc driver will load specific files from /lib/firmware into the R5F cores
- The easiest way to select which R5F firmware gets loaded is to update softlinks to point to the desired executable files

Core Name	RemoteProc Name	Description	Firmware File Name
R5F0-0	78000000.r5f	R5F cluster0-Core0	am64-main-r5f0_0-fw
R5F0-1	78200000.r5f	R5F cluster0-Core1	am64-main-r5f0_1-fw
R5F1-0	78400000.r5f	R5F cluster1-Core0	am64-main-r5f1_0-fw
R5F1-1	78600000.r5f	R5F cluster1-Core1	am64-main-r5f1_1-fw

https://software-dl.ti.com/processor-sdk-linux/esd/docs/08_00_00_21/linux/Foundational_Components_IPC64x.html

How to load remote cores during Linux runtime

- The RemoteProc driver does not support a graceful shutdown of remote cores through AM64x Linux SDK 8.1. That means R5F and M4F will be in an unknown state if the cores are initialized, and then stopped.
- For the moment, we recommend updating remote core firmware in /lib/firmware and rebooting the board in order to load a new binary

https://software-dl.ti.com/processor-sdk-linux/esd/docs/08_00_00_21/linux/Foundational_Components_IPC64x.html

How to build the MCU+ SDK IPC examples

- MCU+ SDK projects can be built with makefiles, or with CCS projects. As an example, let us build the IPC examples on a Linux machine using makefiles.
- Download, install, and setup the SDK and tools as per “Getting Started”
- Run make as per “Building System Examples with Makefiles”

```
$ make -s -C examples/drivers/ipc/ipc_rpmsg_echo_linux/am64x-evm/system_freertos all
```

https://software-dl.ti.com/mcu-plus-sdk/esd/AM64X/08_01_00_36/exports/docs/api_guide_am64x/SDK_DOWNLOAD_PAGE.html

https://software-dl.ti.com/mcu-plus-sdk/esd/AM64X/08_01_00_36/exports/docs/api_guide_am64x/GETTING_STARTED_BUILD.html

https://software-dl.ti.com/mcu-plus-sdk/esd/AM64X/08_01_00_36/exports/docs/api_guide_am64x/MAKEFILE_BUILD_PAGE.html#autotoc_md150

How to build the Linux ti-rpmsg-char example

- We will build the Linux Userspace example for Linux RPMsg by following the steps in the top-level README. Note that these steps were tested on Ubuntu 18.04. Later versions of Ubuntu may need different steps
- Download the git repo. Install GNU autoconf, GNU automake, GNU libtool, and v8 compiler as per the README
- Perform the Build Steps as per the README

https://git.ti.com/cgit/rpmsg/ti-rpmsg-char

https://git.ti.com/cgit/rpmsg/ti-rpmsg-char/tree/README

How to build the Linux rpmsg_client_sample driver

- The Linux rpmsg_client_sample driver is in the Linux SDK under samples/rpmsg/rpmsg_client_sample.c. That means we can build it by building the sample kernel code

  $ export PATH=<sdk path>/linux-devkit/sysroots/x86_64-arago-linux/usr/bin:$PATH

  $ make ARCH=arm64 CROSS_COMPILE=aarch64-none-linux-gnu- distclean

  $ make ARCH=arm64 CROSS_COMPILE=aarch64-none-linux-gnu- tisdk_am64xx-evm_defconfig

  $ make ARCH=arm64 CROSS_COMPILE=aarch64-none-linux-gnu- menuconfig

  – Verify Kernel hacking > Sample kernel code > Build rpmsg client sample is M

  – Make kernel & modules. Multithreading with –j is optional

  $ make ARCH=arm64 CROSS_COMPILE=aarch64-none-linux-gnu- -j8

https://software-dl.ti.com/processor-sdk-linux/esd/docs/08_00_00_21/linux/Foundational_Components_Kernel_Users_Guide.html

How to run the Linux <-> R5 IPC Demo

- Copy the R5F binaries from <MCU+_SDK>/examples/drivers/ipc/ipc_rpmsg_echo_linux/am64x-evm/r5fssX-X_freertos/ti-arm-clang/am64-main-r5fX_X-fw into Linux filesystem under /lib/firmware/
- Copy the Linux RPMsg Userspace application from <ti-rpmsg-char_repo>/examples/rpmsg_char_simple into the Linux filesystem
- Copy the Linux RPMsg kernel driver from <Linux_SDK>/board-support/linux-x.x.x/samples/rpmsg/rpmsg_client_sample.ko into the Linux filesystem
- Boot the board, update the R5F firmware symbolic links, reboot the board
- Run examples

https://software-dl.ti.com/processor-sdk-linux/esd/docs/08_00_00_21/linux/Foundational_Components_IPC64x.html