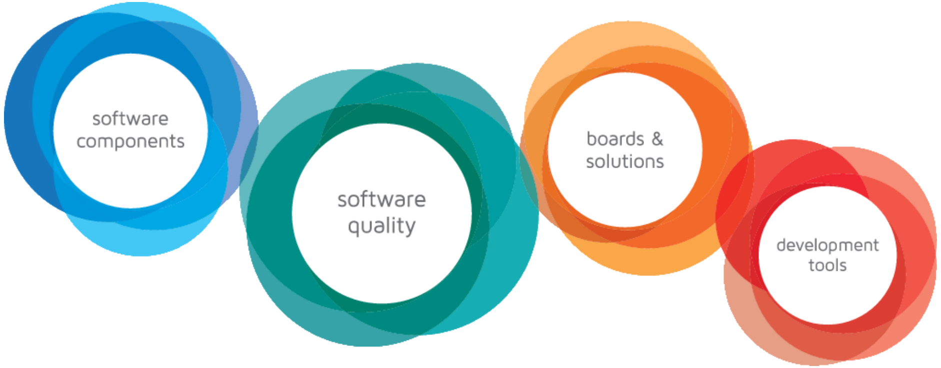


Software Design Challenges for heterogenic SOC's

René Janssen, Product manager Logic Technology

Agenda

1. Advantages of heterogenous devices

2. How to manage inter-processor communication

3. Example Hardware setup demo snapshots

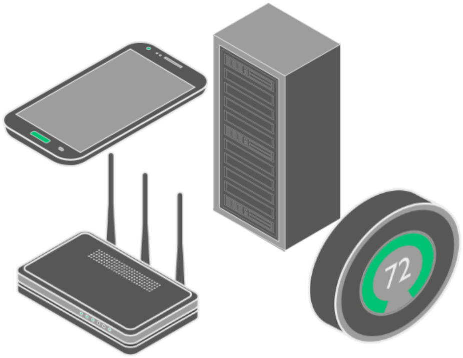
4. Summary

ARM architecture for diverse computing needs

Cortex-A

Highest performance

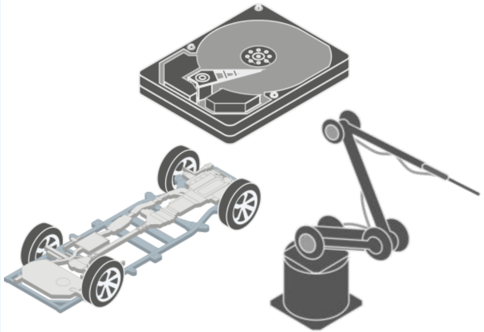
Optimized for
rich operating systems



Cortex-R

Fast response

Optimized for
high performance,
hard real-time applications

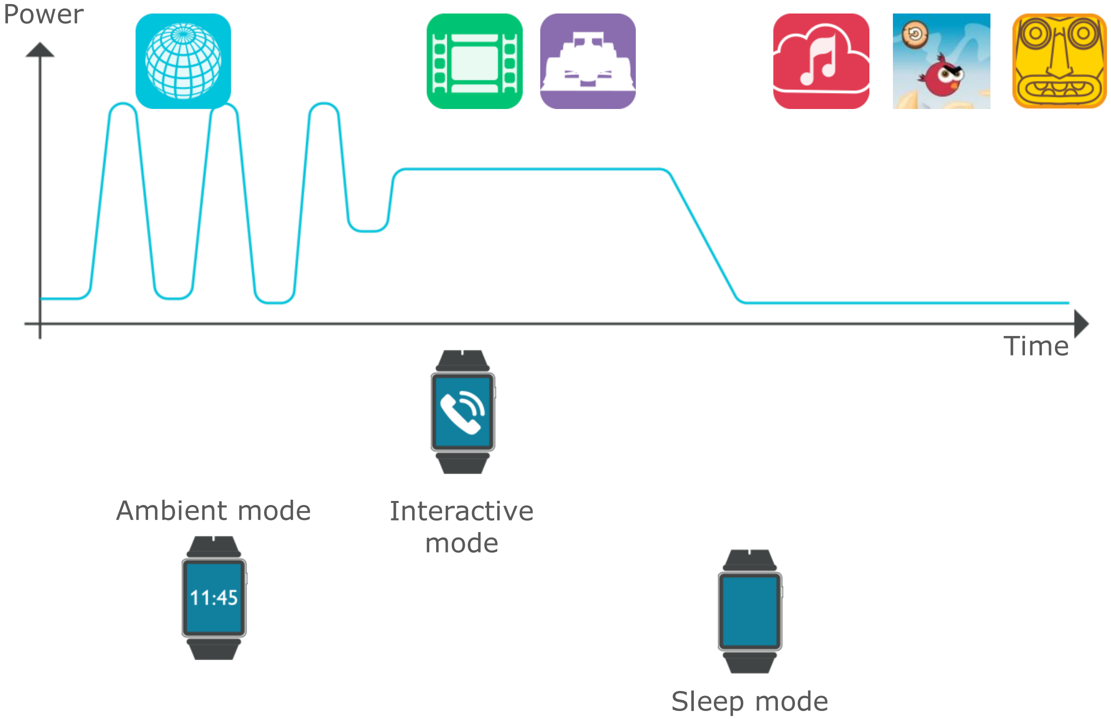


Cortex-M

Smallest/lowest power

Optimized for
discrete processing and
microcontrollers

Modern compute systems have diverse workloads

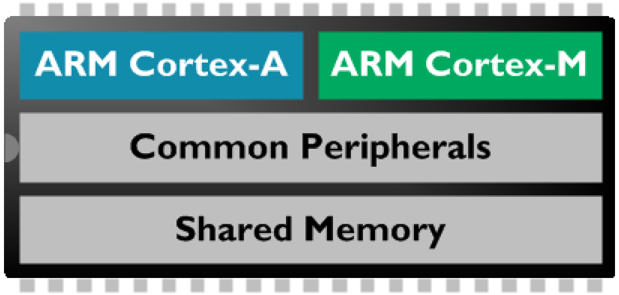


Why heterogeneous devices?

Combines best of both worlds: feature-rich Linux and MCU with deterministic RTOS

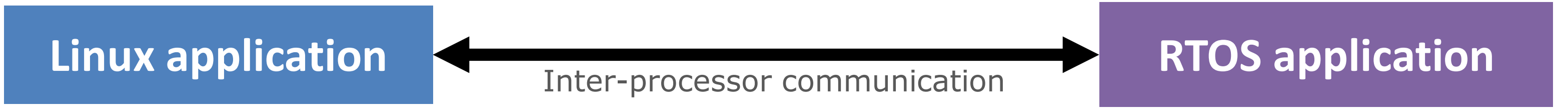
Cortex-A benefits

- Feature rich OS
- Complex applications
- Sophisticated HMI



Cortex-M benefits

- Low I/O latency
- Low power standby
- Fast system start-up time



Use cases of HMP systems in embedded

Industrial

Consumer

Medical

Cortex-A	Rich UI and OS, high performance		
Cortex-M	Real-time control and monitoring	Deterministic sensor control	Real-time monitoring

Overview of software challenges

Task partitioning

How to optimally partition tasks?

Data sharing

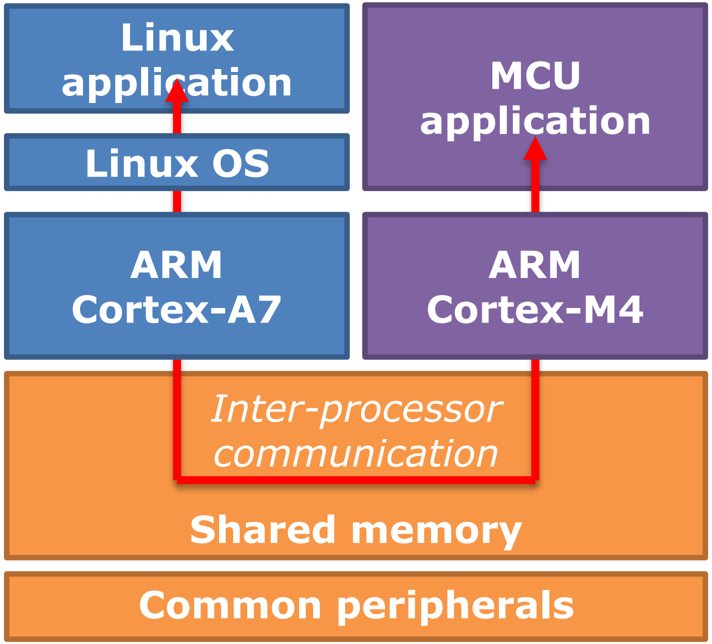
Is coherency necessary?

Developer productivity

Usability, portability, debugging

Complexity increases with heterogeneous devices

- Running multiple operating systems
- Debugging bare-metal code and Linux applications at the same time
- Controlling communication between Linux application and bare-metal application

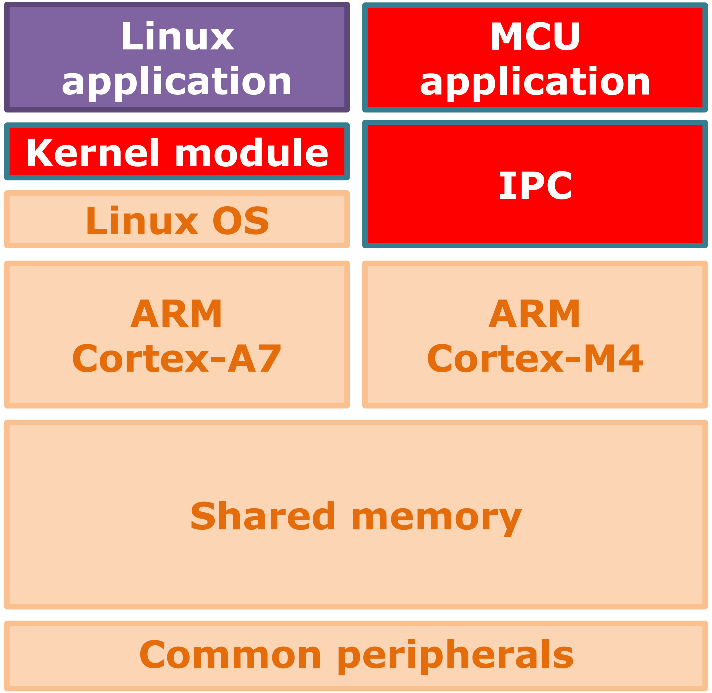


Software development challenges

There's always a Logic Solution!

Manually manage communication between Cortex-A and Cortex-M

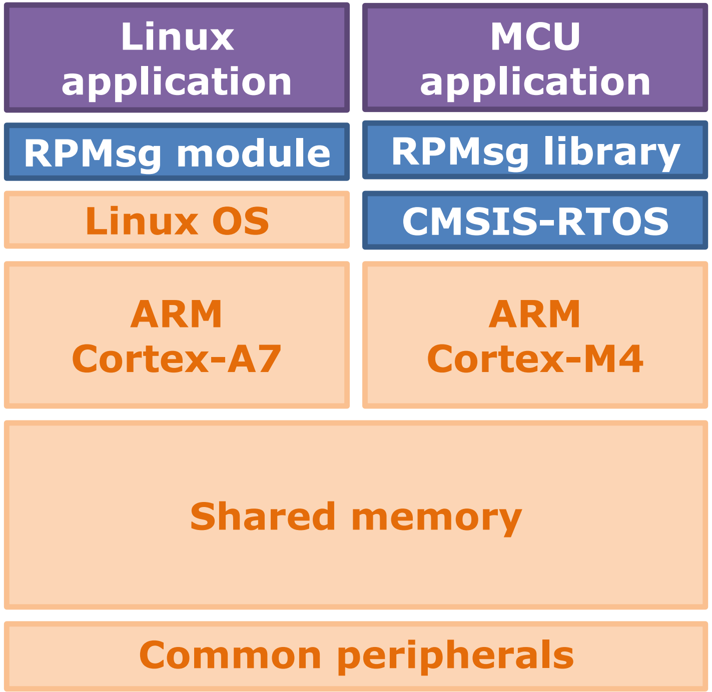
- Requires to write a Linux kernel module
  - » Memory segmentation
  - » Manage concurrency
  - » Kernel mode to User mode communication
- Bare-metal
  - » Manage concurrency
  - » Integration with RTOS
  - » Handle interrupts and memory management



Standardization of software interfaces

CMSIS COMPLIANT
ARM® Cortex™ Microcontroller Software Interface Standard

- CMSIS adopting OpenAMP
  - CMSIS - Cortex Microcontroller Software Interface Standard
  - Open source on Github
- OS support for HMP systems
  - Remote Processor Messaging (RPMsg) for inter-processor communication
  - Management framework using remoteproc

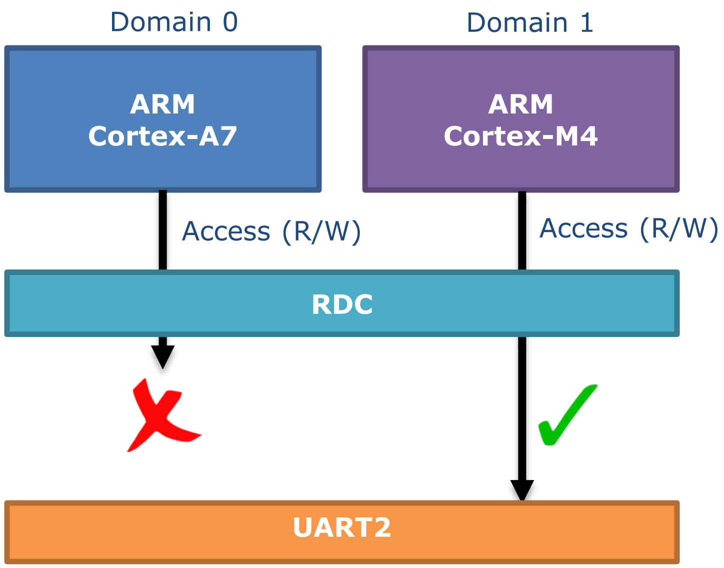


Resource Domain Controller (RDC)

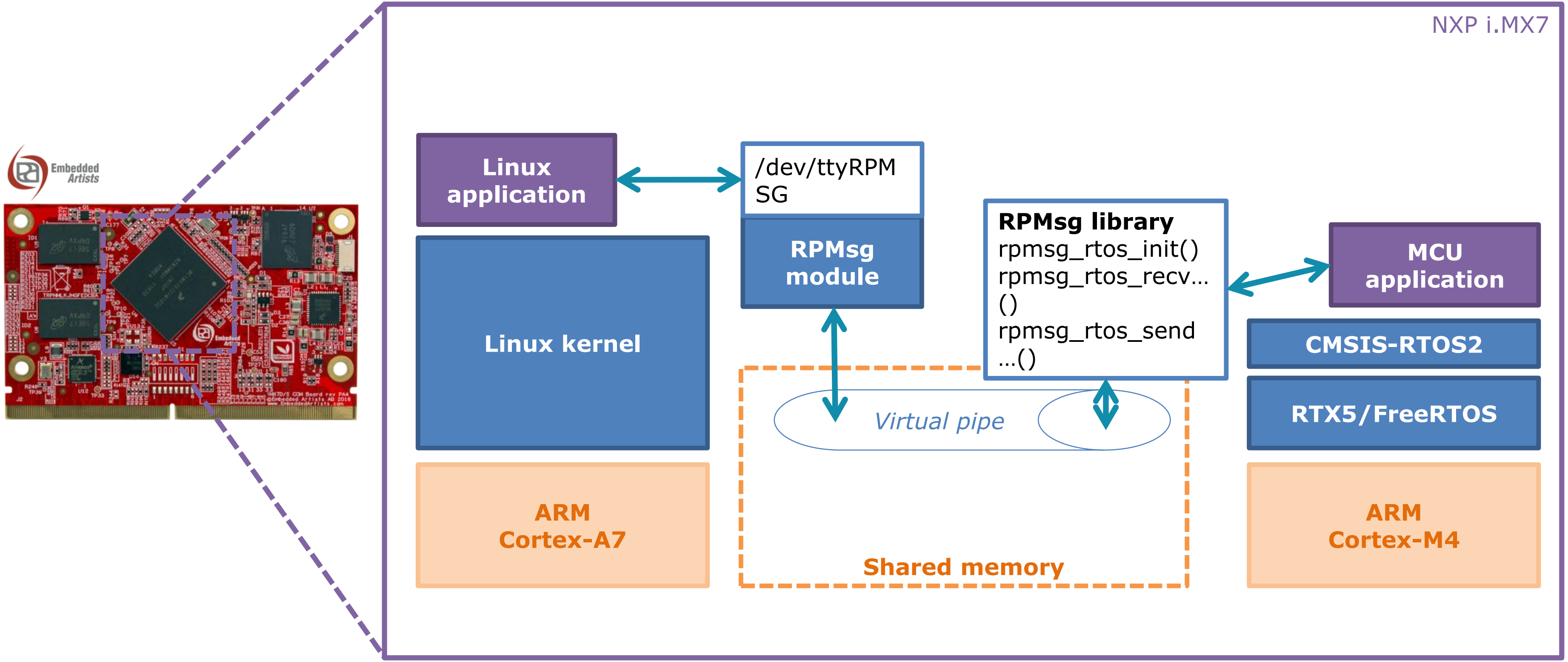
- Improve security by assigning peripherals to different domains
- Reduce risk of concurrent access to peripherals

Example code

```
#define BOARD_DEBUG_UART_RDC_PDAP rdcPdapUart2
#define M4_DOMAIN_ID 1
…
/* Set UART2 for M4 core domain access only */
RDC_SetPdapAccess(RDC,
                  BOARD_DEBUG_UART_RDC_PDAP,
                  3 << (M4_DOMAIN_ID * 2),
                  false, false);
```



Inter-processor communication



NXP i.MX7

Linux application

/dev/ttyRPMSG

RPMsg module

Linux kernel

ARM Cortex-A7

RPMsg library
rpmsg_rtos_init()
rpmsg_rtos_recv…()
rpmsg_rtos_send…()

Virtual pipe

Shared memory

MCU application

CMSIS-RTOS2

RTX5/FreeRTOS

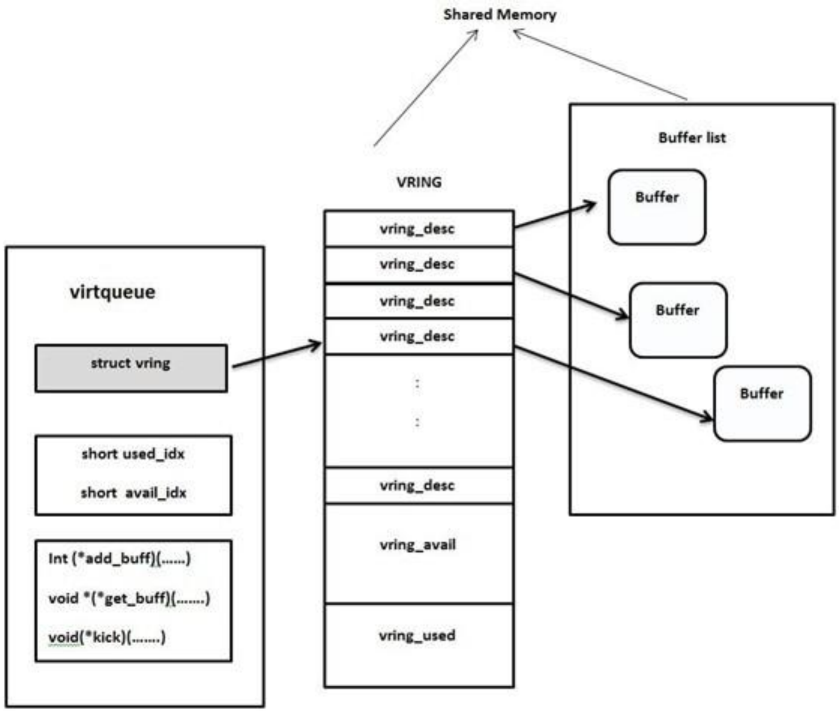
ARM Cortex-M4

Inter-processor communication

- Virtio provides **virtqueue** API that allows user drivers to transmit and receive data with the communicating counterpart using the **vring** structure.

- **Vring** is a buffer management component
  - Ring data structure to manage buffer descriptors located in shared memory.

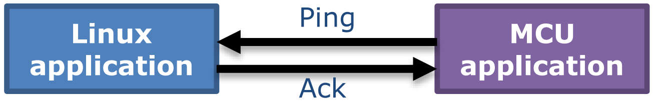
- Inter-Processor Interrupts is used for notifications



Ref: https://github.com/OpenAMP/open-amp/wiki/OpenAMP-RPMsg-Virtio-Implementation

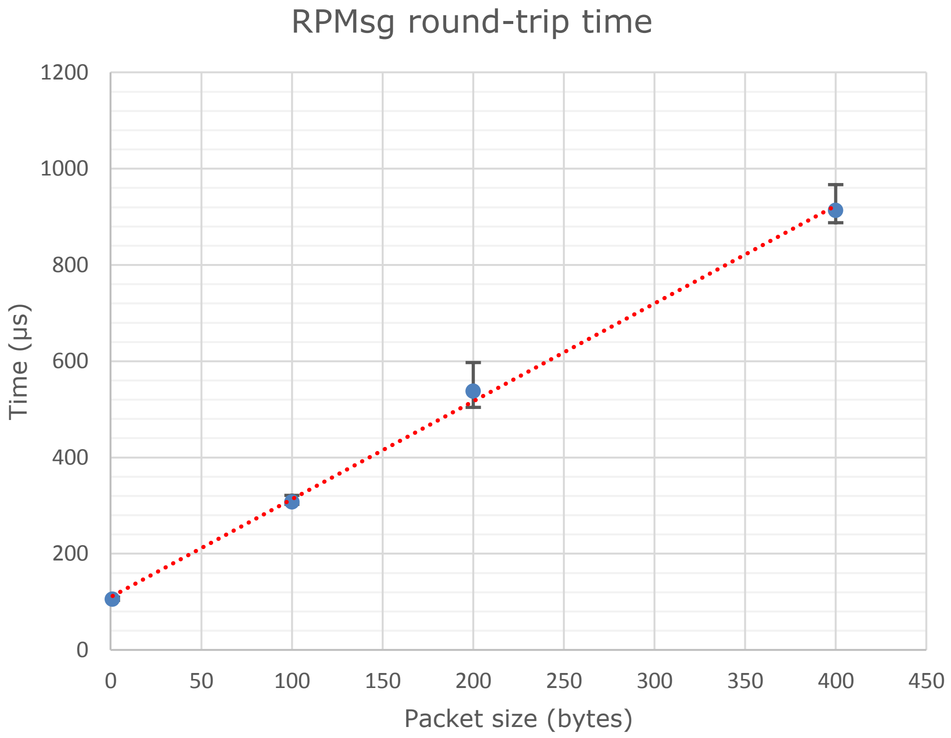
Performance

Measurement of round-trip time



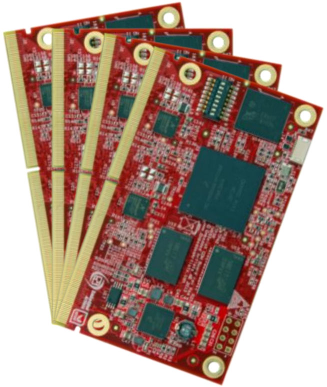
50 µs
Setup time

~1Mb/s
Bandwidth



Typical Hardware setup

Embedded Artists in a Nutshell



- NXP, ARM based Computer-on-Module solutions.
- A family of pin-compatible boards.
- Industrial focus, with 10+ years longevity.
- A Proven Partner to NXP.
- High quality technical support directly from our engineers.

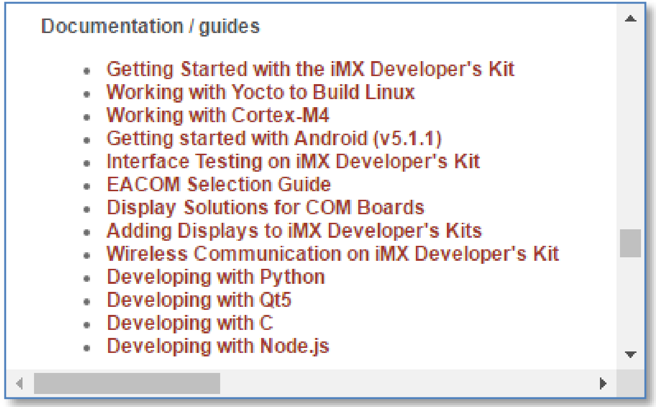
- The Art of Embedded Systems Development – made EASY™

Proven Partner NXP

iMX Developer's Kits

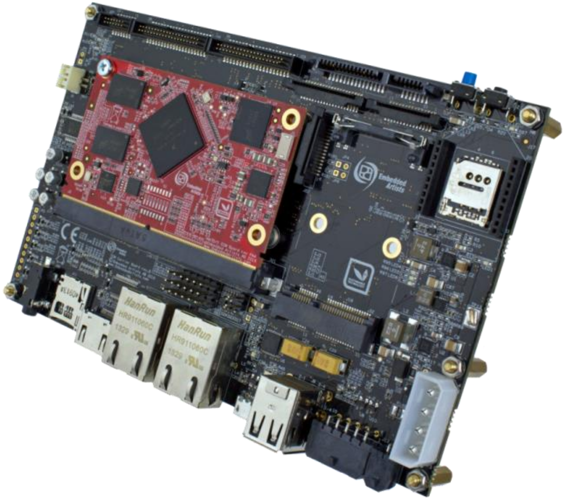
- Flexible hardware platforms..
- Documentation and guides…
- High quality support…

…to get up-and-running quickly!
…for Prototypes and Proof-of-concepts!



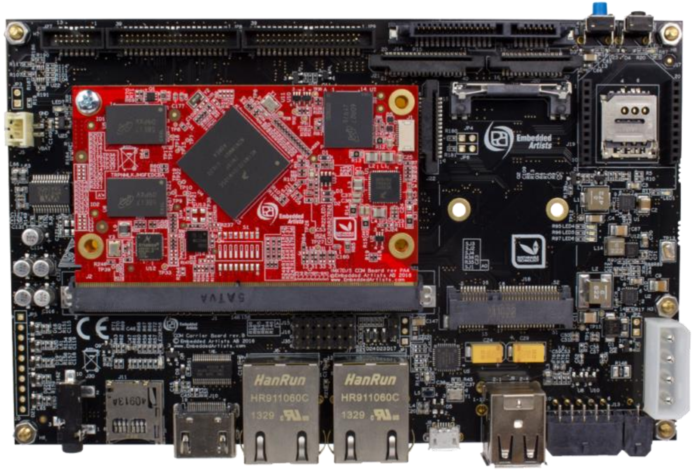
Documentation / guides

- Getting Started with the iMX Developer's Kit
- Working with Yocto to Build Linux
- Working with Cortex-M4
- Getting started with Android (v5.1.1)
- Interface Testing on iMX Developer's Kit
- EACOM Selection Guide
- Display Solutions for COM Boards
- Adding Displays to iMX Developer's Kits
- Wireless Communication on iMX Developer's Kit
- Developing with Python
- Developing with Qt5
- Developing with C
- Developing with Node.js



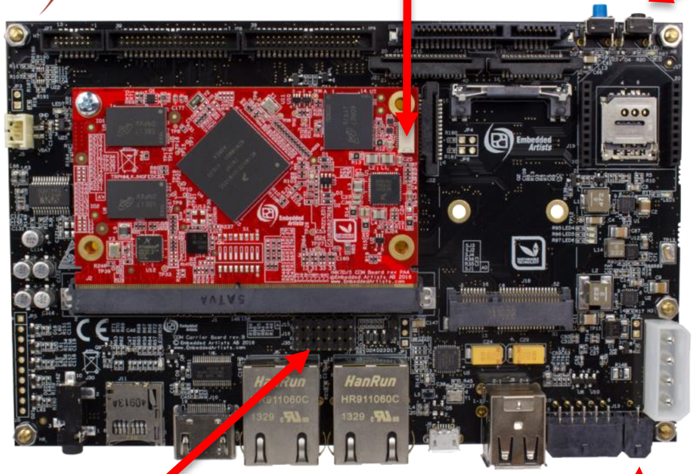
Development board

- iMX7 Dual Developer's Kit
  - 1 GHz dual-core Cortex-A7 (and 200 MHz Cortex-M4)
  - 1 GByte DDR3L RAM
  - 4 Gbyte eMMC Flash
  - One Gigabit Ethernet interface
- To get started with an iMX Developer's Kit
  - http://www.embeddedartists.com/com_getting_started



Development environment

Embedded Artists IMX7D Developer Kit + Arm DSTREAM-ST



JTAG-10

Reset button

UARTs
(via FTDI cables)

Ethernet

Power

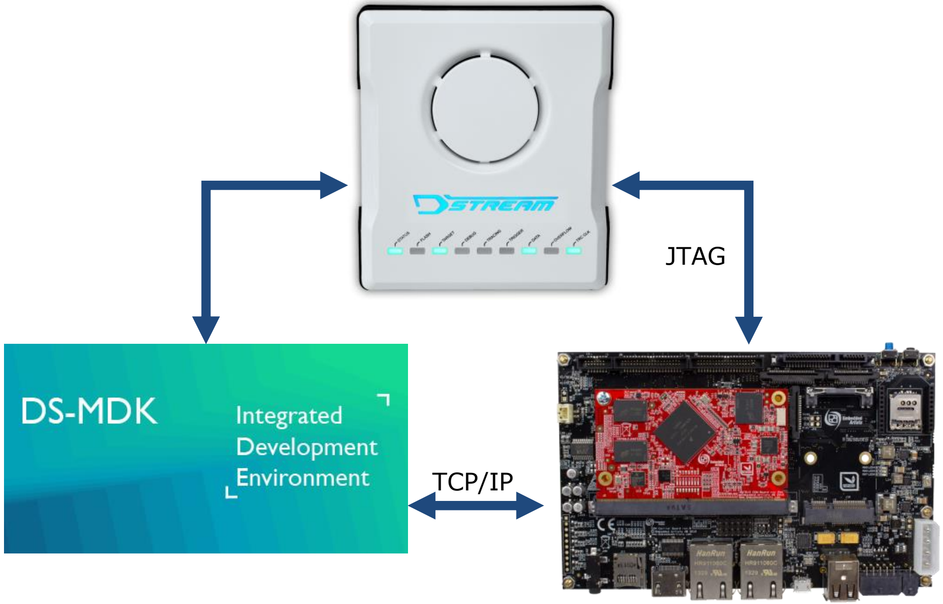
Power

USB or
Ethernet

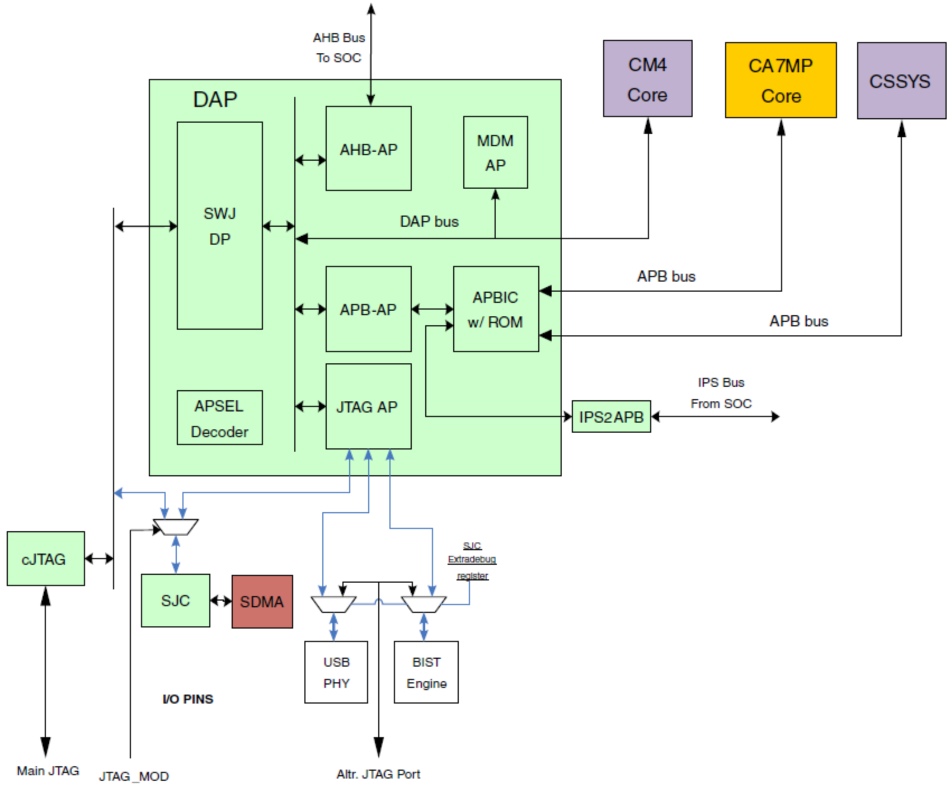
Multiple simultaneous debug connections

Complete visibility to all software applications in the heterogeneous system

- The Cortex-M application via DSTREAM and analyse with RTOS awareness and peripheral views.
- The Cortex-A Linux kernel and Linux kernel modules via DSTREAM and list kernel threads and processes.
- The Cortex-A Linux application via gdbserver on the running Linux system using Remote System Explorer.



Coresight debug interface



I.mx7 Debug interface

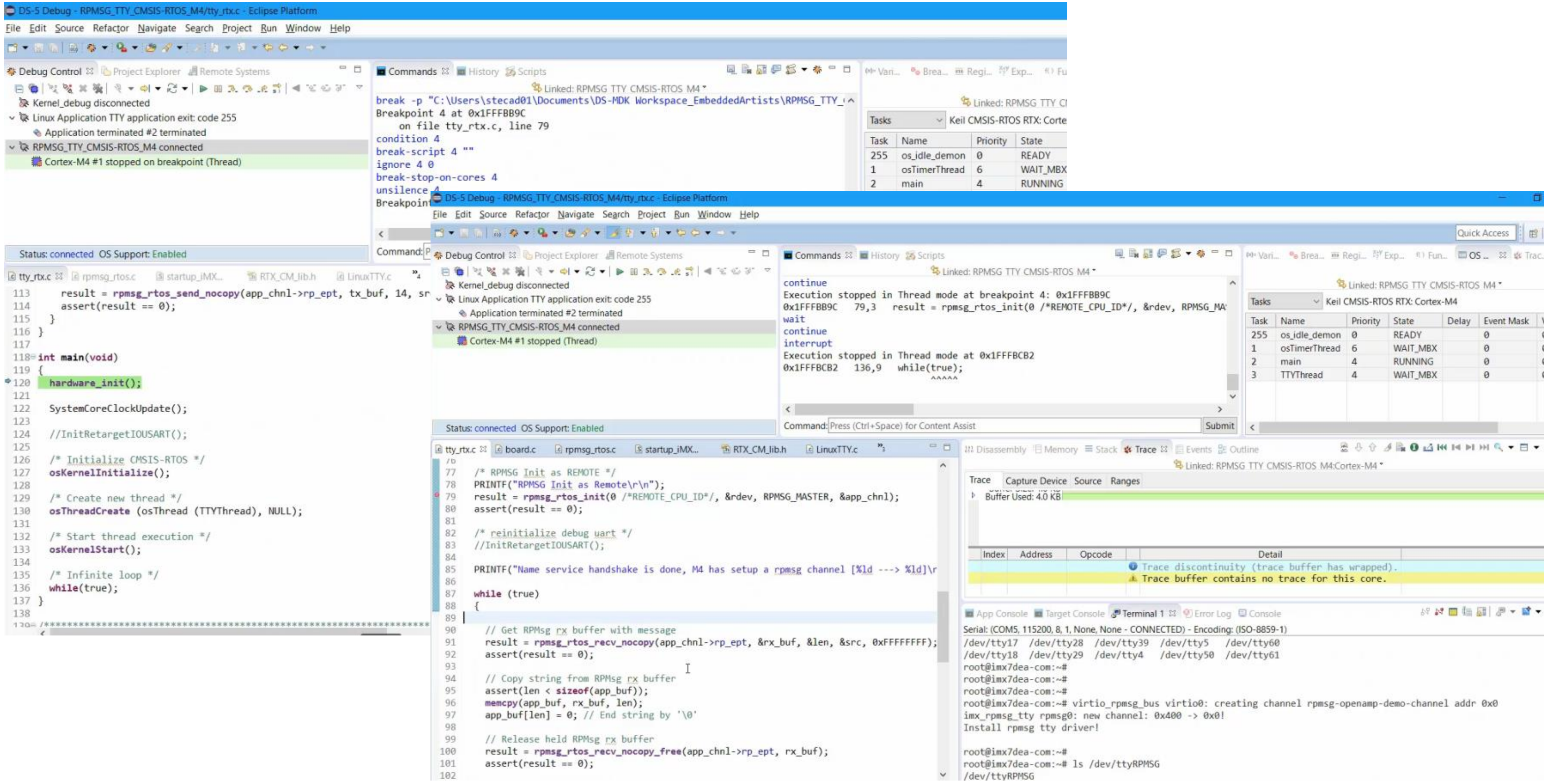
The i.MX7D debug is based on ARM's CoreSight "HUGO" platform, with support for Cortex-A7 and Cortex-M4 core from ARM.

- Support 5-pins (JTAG), 2-pins (cJTAG, ARM SWD) interface
- Support both non-intrusive and halt-mode trace / debug options
- MDM-AP registers for debugger to control mutli-core halt / resume cores
- Trace Memory Controller (TMC) is used to enable capturing trace
  - 4 KB in SOC trace block
  - ETR (4 G memory range, 64-bit wide at 266 MHz) is used to allow routing tracedata to system memory
- Support ARM real time trace interface: TPIU (16-bit x 133 MHz)
- Support cross trigger between CA7 and CM4
- Four JTAG security levels, via SJC security functions together with e-Fuse(challenge response, field return, intrusive detection

Source : i.MX 7Dual Applications Processor Reference Manual, Rev. 0.1, 08/2016

Demo

Demo



Kernel_debug disconnected
Linux Application TTY application exit: code 255
Application terminated #2 terminated
RPMSG_TTY_CMSIS-RTOS_M4 connected
Cortex-M4 #1 stopped on breakpoint (Thread)

Status: connected OS Support: Enabled

Linked: RPMSG TTY CMSIS-RTOS M4

```
break -p "C:\Users\stecad01\Documents\DS-MDK Workspace_EmbeddedArtists\RPMSG_TTY_
Breakpoint 4 at 0x1FFFBB9C
    on file tty_rtx.c, line 79
condition 4
break-script 4 ""
ignore 4 0
break-stop-on-cores 4
unsilence 4
Breakpoint
```

Tasks — Keil CMSIS-RTOS RTX: Corte

Task	Name	Priority	State
255	os_idle_demon	0	READY
1	osTimerThread	6	WAIT_MBX
2	main	4	RUNNING

tty_rtx.c — rpmsg_rtos.c — startup_iMX... — RTX_CM_lib.h — LinuxTTY.c

```c
    result = rpmsg_rtos_send_nocopy(app_chnl->rp_ept, tx_buf, 14, sr
    assert(result == 0);
  }
}

int main(void)
{
  hardware_init();

  SystemCoreClockUpdate();

  //InitRetargetIOUSART();

  /* Initialize CMSIS-RTOS */
  osKernelInitialize();

  /* Create new thread */
  osThreadCreate (osThread (TTYThread), NULL);

  /* Start thread execution */
  osKernelStart();

  /* Infinite loop */
  while(true);
}

/*****************************************************************
```

DS-5 Debug - RPMSG_TTY_CMSIS-RTOS_M4/tty_rtx.c - Eclipse Platform

Kernel_debug disconnected
Linux Application TTY application exit: code 255
Application terminated #2 terminated
RPMSG_TTY_CMSIS-RTOS_M4 connected
Cortex-M4 #1 stopped (Thread)

Status: connected OS Support: Enabled

Linked: RPMSG TTY CMSIS-RTOS M4

```
continue
Execution stopped in Thread mode at breakpoint 4: 0x1FFFBB9C
0x1FFFBB9C   79,3   result = rpmsg_rtos_init(0 /*REMOTE_CPU_ID*/, &rdev, RPMSG_MA
wait
continue
interrupt
Execution stopped in Thread mode at 0x1FFFBCB2
0x1FFFBCB2   136,9   while(true);
                           ^^^^^
```

Command: Press (Ctrl+Space) for Content Assist

Linked: RPMSG TTY CMSIS-RTOS M4

Tasks — Keil CMSIS-RTOS RTX: Cortex-M4

Task	Name	Priority	State	Delay	Event Mask
255	os_idle_demon	0	READY		0
1	osTimerThread	6	WAIT_MBX		0
2	main	4	RUNNING		0
3	TTYThread	4	WAIT_MBX		0

tty_rtx.c — board.c — rpmsg_rtos.c — startup_iMX... — RTX_CM_lib.h — LinuxTTY.c

```c
/* RPMSG Init as REMOTE */
PRINTF("RPMSG Init as Remote\r\n");
result = rpmsg_rtos_init(0 /*REMOTE_CPU_ID*/, &rdev, RPMSG_MASTER, &app_chnl);
assert(result == 0);

/* reinitialize debug uart */
//InitRetargetIOUSART();

PRINTF("Name service handshake is done, M4 has setup a rpmsg channel [%ld ---> %ld]\r

while (true)
{

  // Get RPMsg rx buffer with message
  result = rpmsg_rtos_recv_nocopy(app_chnl->rp_ept, &rx_buf, &len, &src, 0xFFFFFFFF);
  assert(result == 0);

  // Copy string from RPMsg rx buffer
  assert(len < sizeof(app_buf));
  memcpy(app_buf, rx_buf, len);
  app_buf[len] = 0; // End string by '\0'

  // Release held RPMsg rx buffer
  result = rpmsg_rtos_recv_nocopy_free(app_chnl->rp_ept, rx_buf);
  assert(result == 0);
```

Linked: RPMSG TTY CMSIS-RTOS M4:Cortex-M4

Trace — Capture Device — Source — Ranges

Buffer Used: 4.0 KB

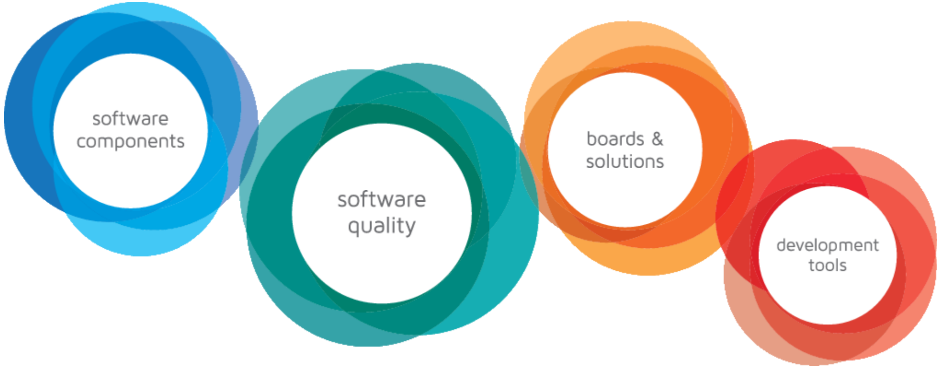
Index	Address	Opcode	Detail
			Trace discontinuity (trace buffer has wrapped).
			Trace buffer contains no trace for this core.

Serial: (COM5, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)

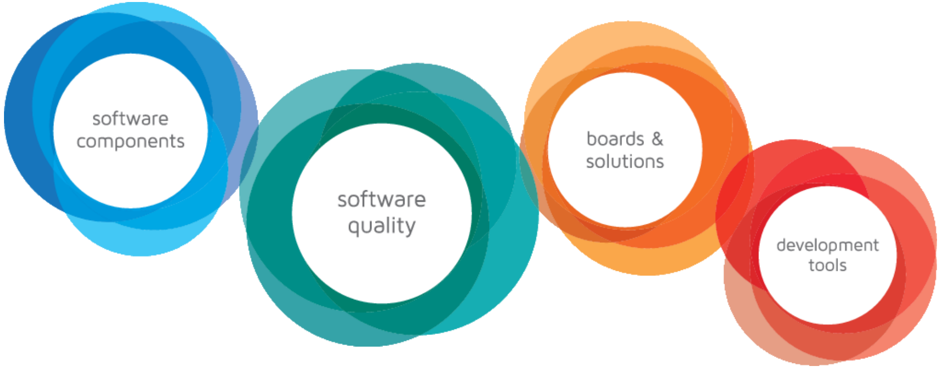
```
/dev/tty17  /dev/tty28  /dev/tty39  /dev/tty5   /dev/tty60
/dev/tty18  /dev/tty29  /dev/tty4   /dev/tty50  /dev/tty61
root@imx7dea-com:~#
root@imx7dea-com:~#
root@imx7dea-com:~#
root@imx7dea-com:~# virtio_rpmsg_bus virtio0: creating channel rpmsg-openamp-demo-channel addr 0x0
imx_rpmsg_tty rpmsg0: new channel: 0x400 -> 0x0!
Install rpmsg tty driver!

root@imx7dea-com:~#
root@imx7dea-com:~# ls /dev/ttyRPMSG
/dev/ttyRPMSG
```

Q & A



Do you have any questions?



Thank you for your attention

See you at the Logic Technology stand