CEC Continuing Education Center

DesignNews

Introduction to Multicore RTOS-based Application Development

DAY 5 : Writing Multicore Microcontroller Applications

Sponsored by

Digi-Key ELECTRONICS

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.
- Submit questions for the lecturer using the Q&A widget. They will follow-up after the lecture portion concludes.

Course Sessions

- Multicore Application Architecture Design
- A Quick Review of RTOS Fundamentals
- Digging into the Dual-Core STM32H7 MCU's
- Toolchain Setup for Dual Core MCU's
- Writing Multicore Microcontroller Applications

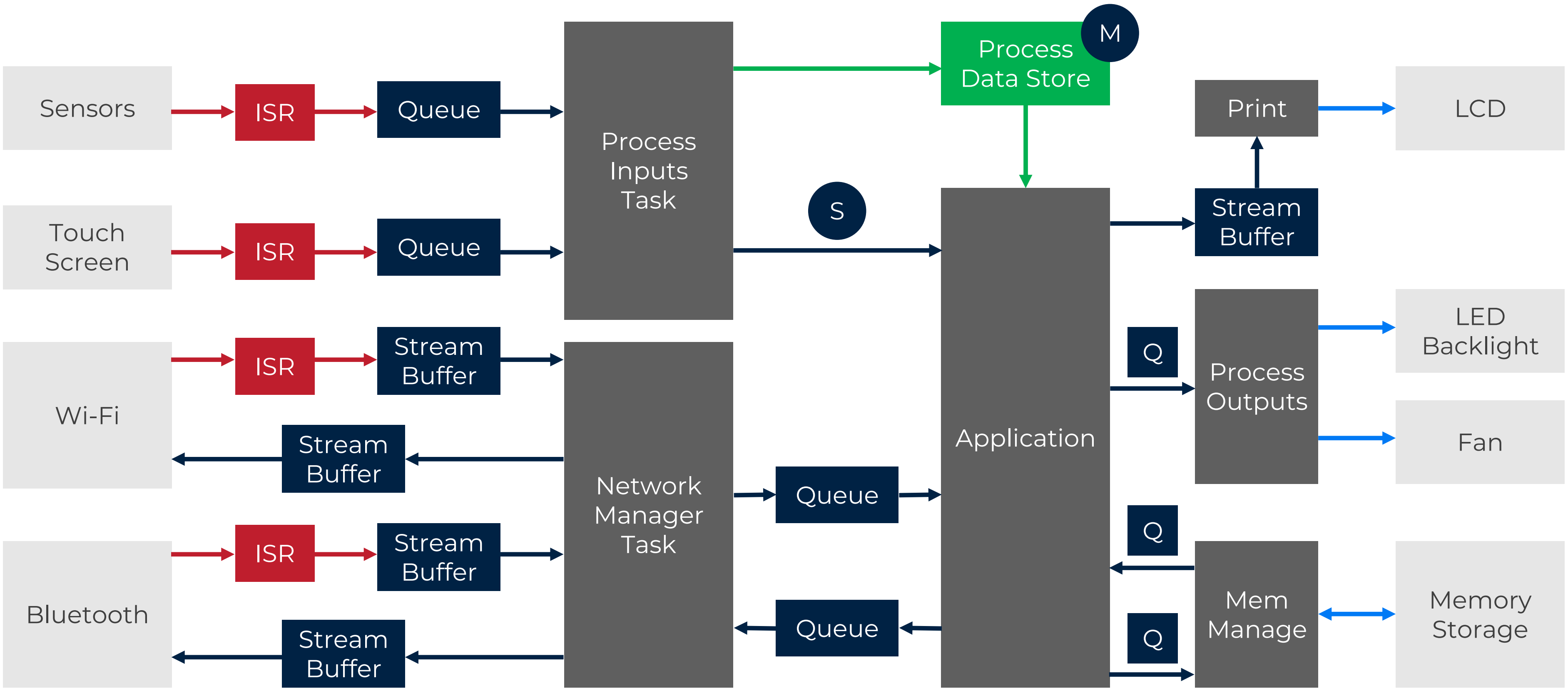
When is your next multicore application going to be started?

- Working on it right now
- Next 1 – 3 months
- Next 3 - 6 months
- Next 6 – 12 months
- Much later

RTOS Task Decomposition

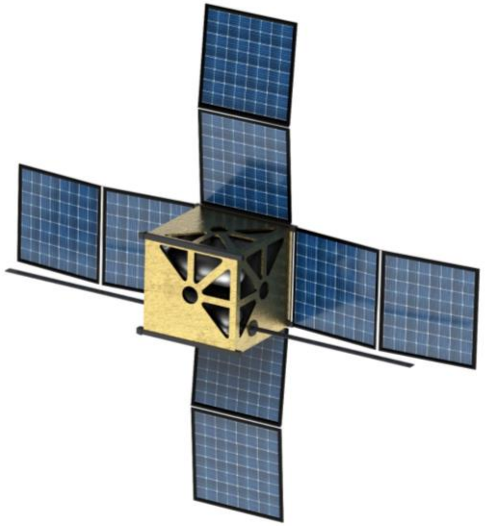
1. Identify the major components
2. Draw a high-level block diagram
3. Label the inputs
4. Label the outputs
5. Identify the first-tier tasks
6. Determine concurrency levels and dependencies
7. Identify second tier tasks (application only tasks)

RTOS Application Design – Single Core

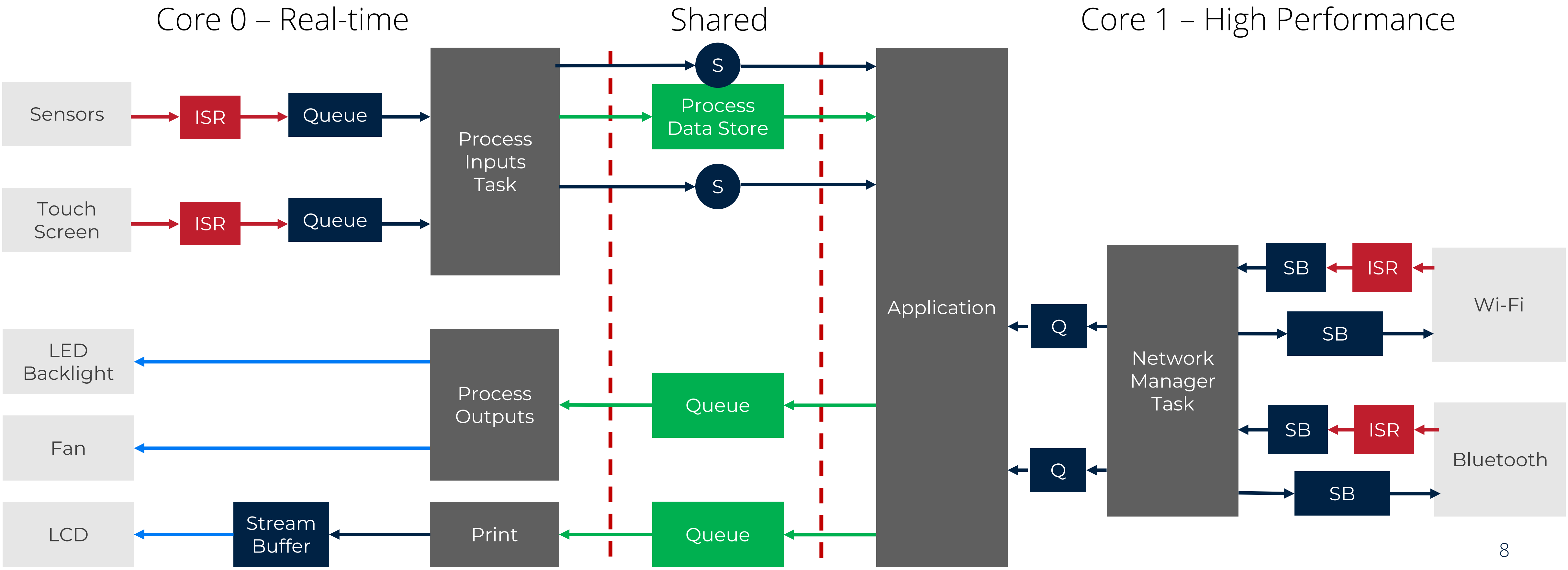


RTOS Application Design – Dual Core

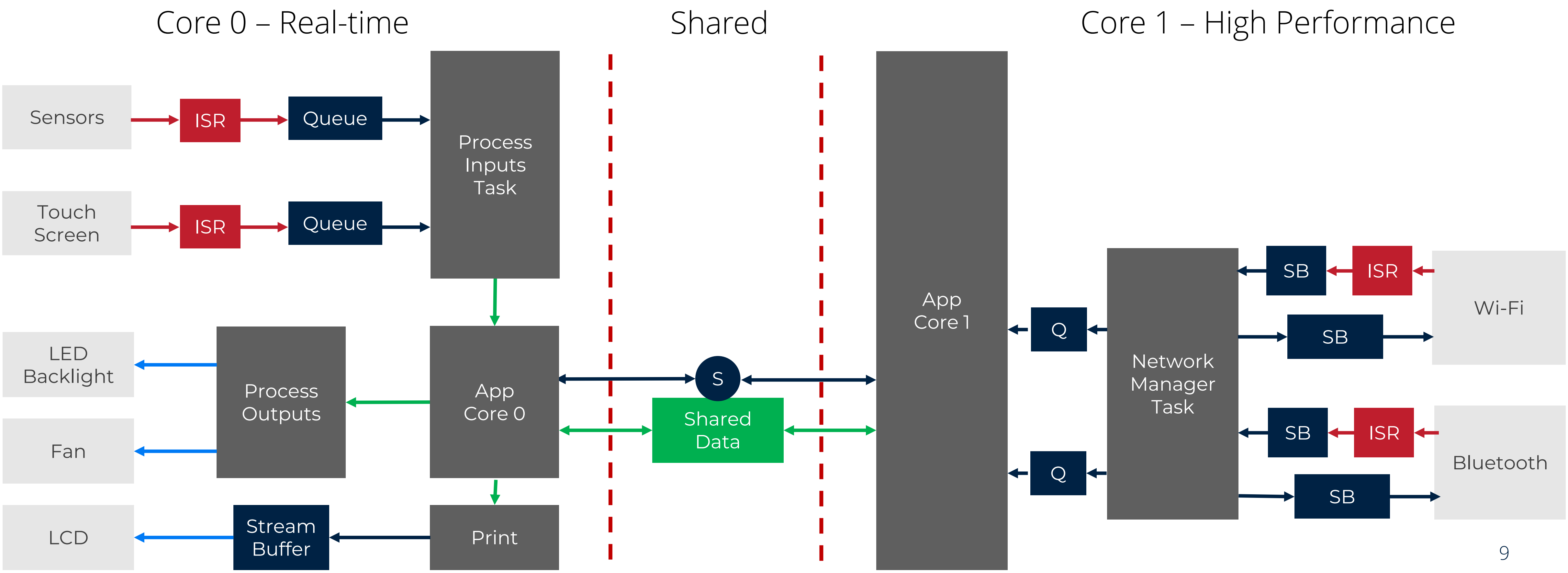
1. Partition the Application Domains
2. Decompose each execution domain
3. Identify domain concurrencies and shared resources
4. Synchronize cross domain tasks



RTOS Application Design – Dual Core Example #1



RTOS Application Design – Dual Core Example #1



Which of the two examples do you find to be the better solution?

- Example #1
- Example #2
- Neither

OpenAMP MW

OpenAMP is an **Open-source A**symmetric **M**ulti-**P**rocessing framework for developing applications on processors with multiple cores.

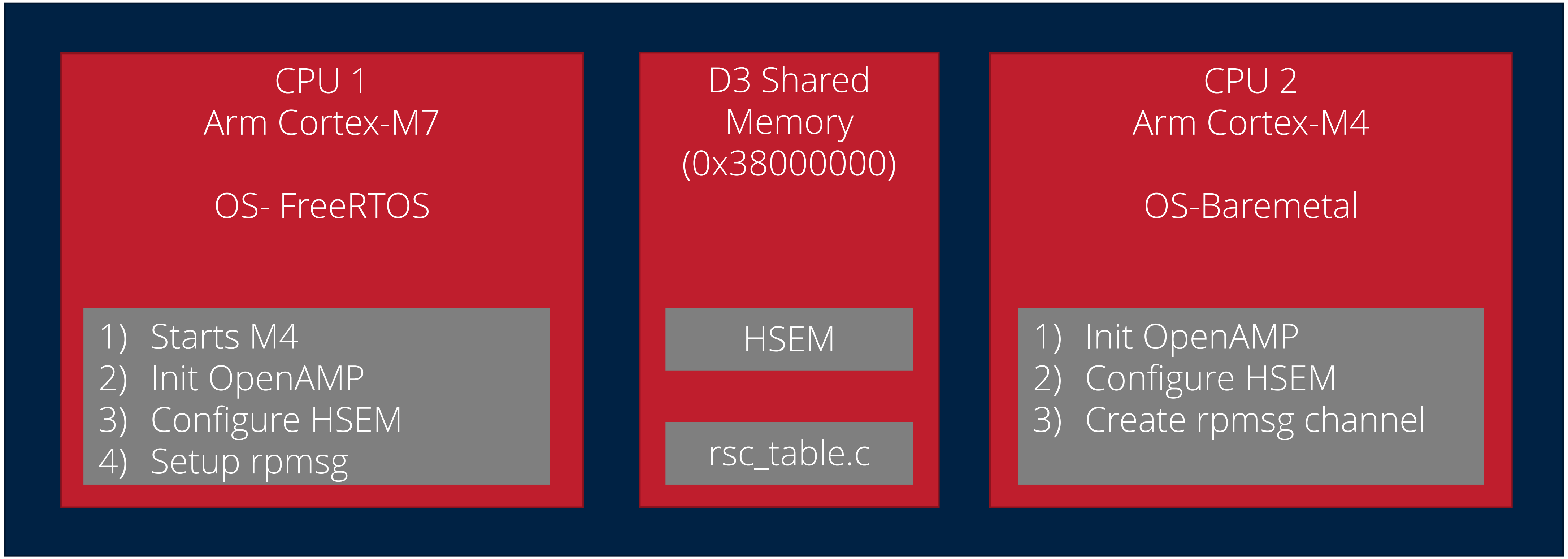
- Used where each process is under its own domain (no Linux or Windows)
- Based on libmetal which provides:
  - OS independent abstraction layer
  - A virtual device framework (Virtio)
  - A Virtio based messaging system (Rpmsg)
  - API’s for life cycle management (Remoteproc)

OpenAMP MW

Virtio – shared memory management framework that shares data through virtio rings, which are FIFO data queues. (Data buffers).

Rpmsg – virtio-based messaging bus that enables inter-processor communications. Can send and receive variable data length message data defined by the application. (Must create a communication channel which includes a source and destination address).

An Example OpenAMP Application – Ping Pong



CPU 1
Arm Cortex-M7

OS- FreeRTOS

1) Starts M4
2) Init OpenAMP
3) Configure HSEM
4) Setup rpmsg

D3 Shared
Memory
(0x38000000)

HSEM

rsc_table.c

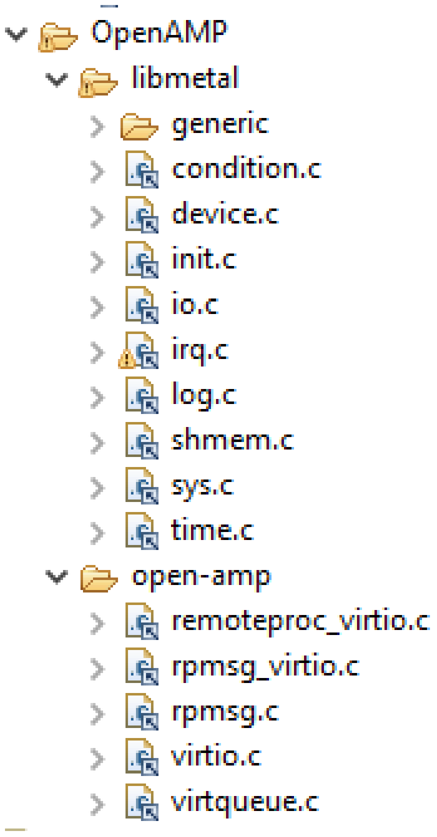
CPU 2
Arm Cortex-M4

OS-Baremetal

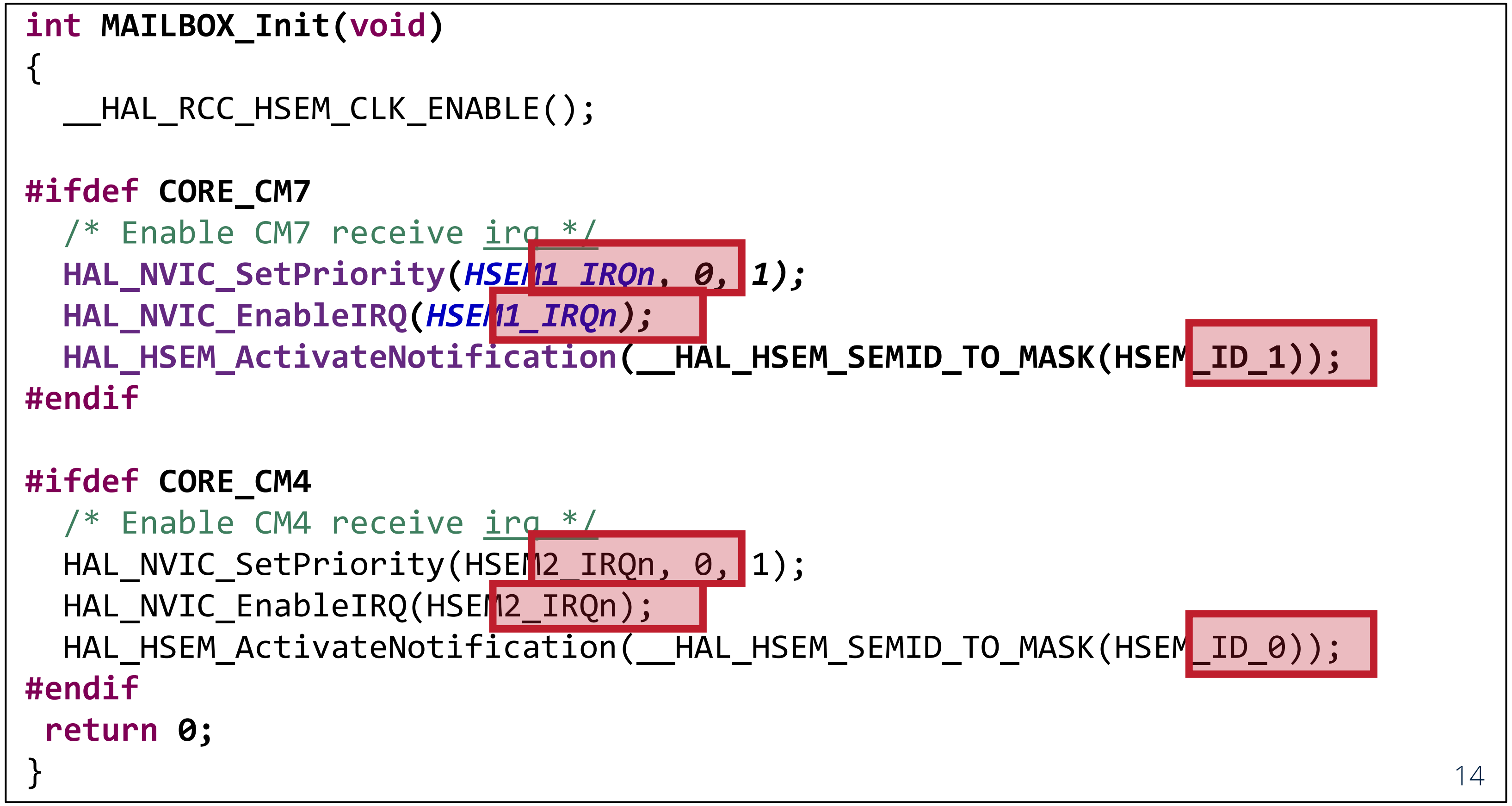
1) Init OpenAMP
2) Configure HSEM
3) Create rpmsg channel

Message is sent -> CPU Rx and Increments -> Sends back -> Repeat

An Example OpenAMP Application – Ping Pong



```
v OpenAMP
  v libmetal
    > generic
    > condition.c
    > device.c
    > init.c
    > io.c
    > irq.c
    > log.c
    > shmem.c
    > sys.c
    > time.c
  v open-amp
    > remoteproc_virtio.c
    > rpmsg_virtio.c
    > rpmsg.c
    > virtio.c
    > virtqueue.c
```



```
int MAILBOX_Init(void)
{
  __HAL_RCC_HSEM_CLK_ENABLE();

#ifdef CORE_CM7
  /* Enable CM7 receive irq */
  HAL_NVIC_SetPriority(HSEM1_IRQn, 0, 1);
  HAL_NVIC_EnableIRQ(HSEM1_IRQn);
  HAL_HSEM_ActivateNotification(__HAL_HSEM_SEMID_TO_MASK(HSEM_ID_1));
#endif

#ifdef CORE_CM4
  /* Enable CM4 receive irq */
  HAL_NVIC_SetPriority(HSEM2_IRQn, 0, 1);
  HAL_NVIC_EnableIRQ(HSEM2_IRQn);
  HAL_HSEM_ActivateNotification(__HAL_HSEM_SEMID_TO_MASK(HSEM_ID_0));
#endif
 return 0;
}
```

An Example OpenAMP Application – Ping Pong

```
int MAILBOX_Poll(struct virtio_device *vdev)
{

  if (msg_received == RX_NEW_MSG)
  {
#ifdef CORE_CM7
    rproc_virtio_notified(vdev, VRING0_ID);
#endif
#ifdef CORE_CM4
    rproc_virtio_notified(vdev, VRING1_ID);
#endif
    msg_received = RX_NO_MSG;
    return 0;
  }

  return -EAGAIN;
}
```

```
int MAILBOX_Notify(void *priv, uint32_t id)
{
  (void)priv;
  (void)id;

#ifdef CORE_CM7
  HAL_HSEM_FastTake(HSEM_ID_0);
  HAL_HSEM_Release(HSEM_ID_0,0);
#endif
#ifdef CORE_CM4
  HAL_HSEM_FastTake(HSEM_ID_1);
  HAL_HSEM_Release(HSEM_ID_1,0);
#endif

  return 0;
}
```

An Example OpenAMP Application – Ping Pong

Application cm7

```
/* Initialize the mailbox use notify the other core on new message */
  MAILBOX_Init();

  /* Initialize the rpmsg endpoint to set default addresses to RPMSG_ADDR_ANY */
  rpmsg_init_ept(&rp_endpoint, RPMSG_CHAN_NAME, RPMSG_ADDR_ANY, RPMSG_ADDR_ANY,
                 NULL, NULL);
  /* Initialize OpenAmp and libmetal libraries */
  if (MX_OPENAMP_Init(RPMSG_MASTER, new_service_cb)!= HAL_OK)
    Error_Handler();

 /*Take HSEM */
  HAL_HSEM_FastTake(HSEM_ID_0);
  /*Release HSEM in order to notify the CPU2(CM4)*/
  HAL_HSEM_Release(HSEM_ID_0,0);

  OPENAMP_Wait_EndPointready(&rp_endpoint);
```

An Example OpenAMP Application – Ping Pong

Application cm7 - continued

```
/* Send the massage to the remote CPU */
 status = OPENAMP_send(&rp_endpoint, &message, sizeof(message));

while (message < 100)
{
  /* Receive the massage from the remote CPU */
   message = receive_message();
   message++;

  /* Send the massage to the remote CPU */
   status = OPENAMP_send(&rp_endpoint, &message, sizeof(message));
   osDelay(1);

   if (status < 0)
   {
     Error_Handler();
   }
}
```

Which configuration do you think you prefer for Dual Core applications?

- Baremetal <-> Baremetal
- Baremetal <-> RTOS
- RTOS <-> Baremetal
- RTOS <-> RTOS

Thank you for attending

Please consider the resources below:

- www.beningo.com
  - Blog, White Papers, Courses
  - Embedded Bytes Newsletter
    - http://bit.ly/1BAHYXm

From www.beningo.com under

- Blog > CEC – Introduction to Multicore RTOS-based Application Development

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

Digi-Key ELECTRONICS

informa markets