Porting a bare metal OS to the ARMv8-R Architecture

Cortex-R52 Processor

Outline

- Introduction
- Porting of a bare metal OS to the Cortex-R52
  - Interrupt and Exception Model
  - Memory Model and Memory Protection Model
  - Stack Model
- Conclusions

Introduction

- ARM is developing a new processor architecture that is designed to support features which will help in the task of providing solutions to the automotive market demands

- This presentation focuses on porting of a bare metal OS to the Cortex-R52

Relevant aspects for porting a bare metal OS to the Coretx-R52

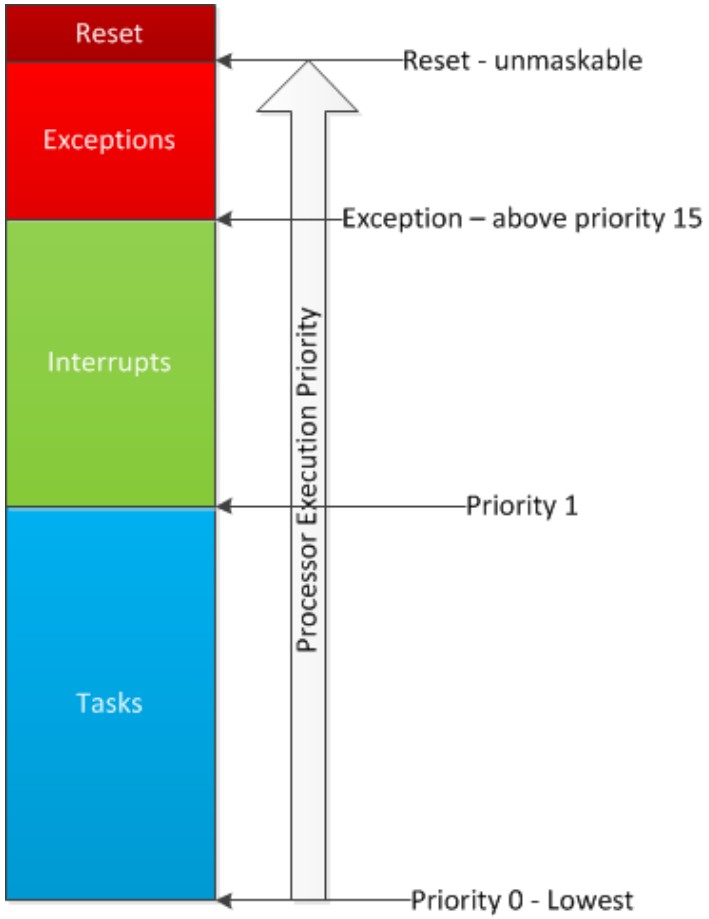
- Porting of a bare metal OS to the Cortex-R52 Architecture

- This task can be broken into the following
  - Interrupt and Exception Model
  - Memory Model and Memory Protection Model
  - Stack Model

Interrupt and Exception Model

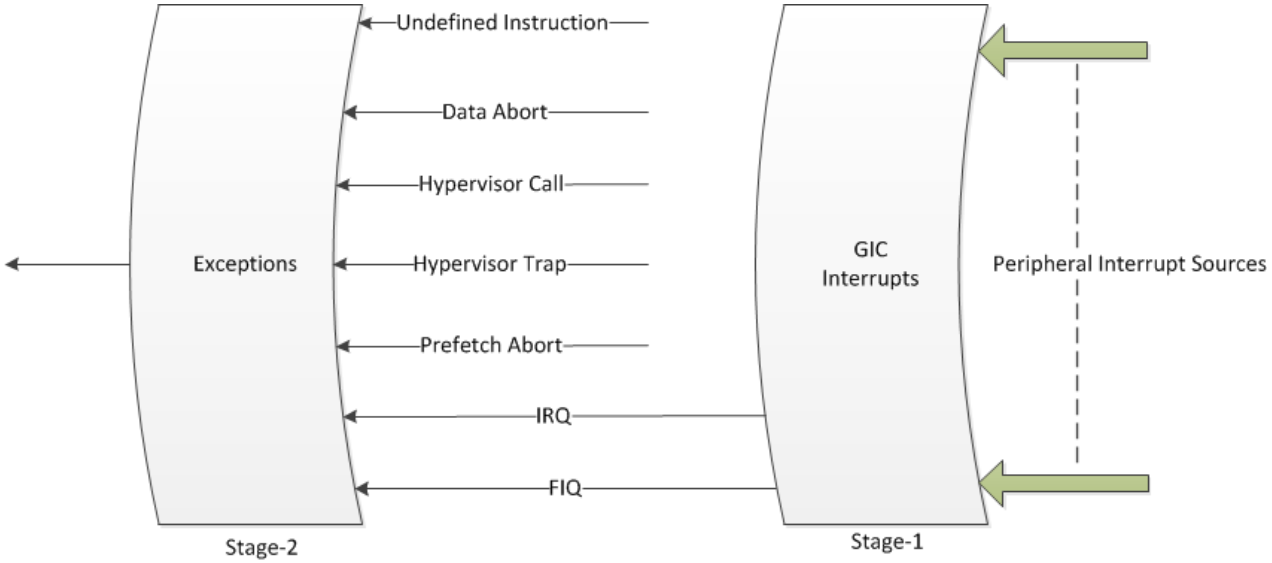
- The OS must provide an abstracted priority space in which interrupts and tasks can execute
- Typically, tasks execute at CPU priority 0 and interrupts above that (but below the reset level)

- In the Cortex-R52 interrupts have lower priority than exceptions
- Also the interrupt grouping is supported



Interrupt and Exception Model

- The Cortex-R52 provides a two level approach
  - Exception are managed at Exception Level 1 or Exception Level 2
  - Interrupts are just like another exception type
  - Interrupt arbitration is managed by the Generic Interrupt Controller (GIC)



Memory Model and Memory Protection Model

- The OS must support the hardware memory model
- This includes the memory protection model

- In order to support AUTOSAR partitions, the OS must provide support for memory protection

- The Cortex-R52 provides a two stage Memory Protection Unit
- Stage-2 not relevant for this use case (typically used for hypervisors)

- Stage-1 MPU can be used to configure memory regions, including:
  - ROM and RAM access
  - I/O access

Stack Model

- The OS must provide support for single and multiple user stacks
- The Cortex-R52 provides banked copies of the stack pointer
  - The hardware automatically switches to a different stack exceptions and interrupts occur
- A separate safe memory region should be configured for the abort stack
  - Used when a stack fault occurs in exception processing

Conclusions

- The Cortex-R52 provides a full comprehensive set of features to support the automotive industry RTOS requirements

- In order to minimise porting costs, an embedded OS should use Exception Level 0 and Exception Level 1 as much as possible

- The Cortex-R52's key features
  - Memory Protection Unit
  - Three level execution privilege scheme
    - User level (i.e. prevented to access to MPU configuration)
    - Guest OS level
    - Hypervisor level
  - Interrupt controller
    - Grouping capability is available (i.e. grouping by type or by priority)

Thank you