Hercules™ ARM® Cortex®-R4
System Architecture

Processor Overview
What is Hercules™?

TI’s 32-bit ARM® Cortex™-R4/R5 MCU family for Industrial, Automotive, and Transportation Safety

Broad Safety MCU Portfolio
- CPU performance from 80MHz to 330MHz
- Flash memory options ranging from 128KB to 4MB

High Reliability
- Proven hardware based on 20+ years of TI safety-critical system expertise
- Development flow refined for high quality & low DPPM

Hardware Safety Features
- Lockstep CPU Architecture
- CPU & RAM Built in Self Test
- RAM & Flash ECC
- Clock & Voltage Monitoring

Hercules™
- ARM Cortex-R4F and R5F CPUs
- Up to 330MHz
- IEC 61508 & ISO 26262 Safety
- Automotive AEC Q100 Qualification
- Development kits starting under $20

Scalable Memory & Peripherals
- 128KB to 4MB of Embedded Flash Memory
- 10-Bit & 12-Bit ADCs
- Ethernet, USB, FR, CAN, LIN
- SPI, I2C, UART, NHET

Reduce Systematic and Random Faults
- Development flow certified to IEC61508 & ISO26262
- Integrated diagnostics protect against random faults

Time To Market
- Hardware safety features reduce software development
- Tools, software and safety documentation
Hercules™ MCUs
Scalable platform for functional safety applications

External certification: ISO 26262, IEC 61508
Documentation: Safety Manual, FMEDA reports
Software: Drivers, libraries, RTOS, Autosar, tools, debug
Development Kits: LaunchPad, HDK, SafeTI CSP, SafeTI CQK
Cortex-R: Ideal for safety-critical applications

Safety features
- Supports Lockstep
- Memory Protection Unit (MPU)
- Error-Correcting Code (ECC)

Higher performance
- 8-stage processor pipeline
- Dual issue – two instructions can execute in parallel
- Load store unit reduces stalling
- Pre-fetch and Branch Prediction Units
- Cached*

Real-time / determinism
- Tightly Coupled Memory (TCM)
- Fast interrupt response
- Deterministic interrupt response
Cortex-R4 features

- ARMv7-R architecture, supports ARM and Thumb2 instruction sets
- 8-stage processor pipeline
- Pre-fetch and Branch Prediction Units
- Floating-Point Unit
- Fast interrupt response
- Tightly Coupled Memory (TCM) with ECC
- Memory Protection Unit (MPU)
- Performance Monitoring Unit (PMU)
Cortex-R4 Hercules Processor

Prefetch unit
- Fetches instructions from the TCMs, or external memory
- Predicts the outcome of branches in the instruction stream

Data Processing Unit (DPU)
- Decodes and executes instructions
- Interfaces with LSU to transfer data to or from the memory system
- Holds general-purpose registers, status registers and control registers (CP15, CP14, etc.)

Load/store unit
- Manages all load & store operations, interfacing with the DPU, TCMs, and memory

AXI master interface
- Provides a high-bandwidth interface to on-chip RAM, peripherals, and interfaces to external memory
- Consists of a single AXI port with a 64-bit read/write channel for instruction & data fetches
- Can run at the same frequency as the processor
Pipeline

The following stages make up the pipeline:

- Fetch stages (Fe1, Fe2)
- Pre-Decode and Decode stages (Pd, De)
- Issue stage (Iss)
- Execution stages (Ex1, Ex2, etc.)
Floating Point Unit (FPU)

- FPU is compliant to IEEE754
- 16 double-word (64 bits) registers
- 32 single-word (32 bits) registers
- Supports features:
  - Single-precision and double-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations
  - Conversions between fixed-point and floating-point data formats, etc
  - Comparisons
  - Underflow
  - Exceptions
Processor modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervisor (SVC)</td>
<td>Entered on reset and when a Software Interrupt instruction (SWI) is executed</td>
</tr>
<tr>
<td>Undef</td>
<td>Used to handle undefined instructions</td>
</tr>
<tr>
<td>Abort</td>
<td>Used to handle memory access violations</td>
</tr>
<tr>
<td>FIQ</td>
<td>Entered when a high priority (fast) interrupt is raised</td>
</tr>
<tr>
<td>IRQ</td>
<td>Entered when a low priority (normal) interrupt is raised</td>
</tr>
<tr>
<td>System</td>
<td>Privileged mode using the same registers as User mode</td>
</tr>
<tr>
<td>User</td>
<td>Mode under which most Applications / OS tasks run</td>
</tr>
</tbody>
</table>

- ARM has 7 basic operating modes.
- Modes other than user mode have privileged access rights.
- Usually the initial setup is done in SVC mode after reset, then switch to system or user mode afterwards.
- Privileged access rights are needed to access certain configuration registers in the processor and peripherals. For example, CP15, flash control registers.
- Cortex-R MCU has MPU, which can be used to set the memory access rights for certain regions.
## Processor registers

<table>
<thead>
<tr>
<th>User/System mode</th>
<th>FIQ mode</th>
<th>IRQ mode</th>
<th>Supervisor mode</th>
<th>Abort Exception</th>
<th>Undefined Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>R0</td>
<td>R0</td>
<td>R0</td>
<td>R0</td>
<td>R0</td>
</tr>
<tr>
<td>R1</td>
<td>R1</td>
<td>R1</td>
<td>R1</td>
<td>R1</td>
<td>R1</td>
</tr>
<tr>
<td>R2</td>
<td>R2</td>
<td>R2</td>
<td>R2</td>
<td>R2</td>
<td>R2</td>
</tr>
<tr>
<td>R3</td>
<td>R3</td>
<td>R3</td>
<td>R3</td>
<td>R3</td>
<td>R3</td>
</tr>
<tr>
<td>R4</td>
<td>R4</td>
<td>R4</td>
<td>R4</td>
<td>R4</td>
<td>R4</td>
</tr>
<tr>
<td>R5</td>
<td>R5</td>
<td>R5</td>
<td>R5</td>
<td>R5</td>
<td>R5</td>
</tr>
<tr>
<td>R6</td>
<td>R6</td>
<td>R6</td>
<td>R6</td>
<td>R6</td>
<td>R6</td>
</tr>
<tr>
<td>R7</td>
<td>R7</td>
<td>R7</td>
<td>R7</td>
<td>R7</td>
<td>R7</td>
</tr>
<tr>
<td>R8</td>
<td>R8 FIQ</td>
<td>R8 FIQ</td>
<td>R8 FIQ</td>
<td>R8 FIQ</td>
<td>R8 FIQ</td>
</tr>
<tr>
<td>R9</td>
<td>R9 FIQ</td>
<td>R9 FIQ</td>
<td>R9 FIQ</td>
<td>R9 FIQ</td>
<td>R9 FIQ</td>
</tr>
<tr>
<td>R10</td>
<td>R10 FIQ</td>
<td>R10 FIQ</td>
<td>R10 FIQ</td>
<td>R10 FIQ</td>
<td>R10 FIQ</td>
</tr>
<tr>
<td>R11</td>
<td>R11 FIQ</td>
<td>R11 FIQ</td>
<td>R11 FIQ</td>
<td>R11 FIQ</td>
<td>R11 FIQ</td>
</tr>
<tr>
<td>R12</td>
<td>R12 FIQ</td>
<td>R12 FIQ</td>
<td>R12 FIQ</td>
<td>R12 FIQ</td>
<td>R12 FIQ</td>
</tr>
<tr>
<td>R13 SP</td>
<td>R13 FIQ</td>
<td>R13 FIQ</td>
<td>R13 FIQ</td>
<td>R13 FIQ</td>
<td>R13 FIQ</td>
</tr>
<tr>
<td>R14 LR</td>
<td>R14 FIQ</td>
<td>R14 FIQ</td>
<td>R14 FIQ</td>
<td>R14 FIQ</td>
<td>R14 FIQ</td>
</tr>
<tr>
<td>R15 PC</td>
<td>R15 PC</td>
<td>R15 PC</td>
<td>R15 PC</td>
<td>R15 PC</td>
<td>R15 PC</td>
</tr>
</tbody>
</table>

- **Stack Pointer (SP)**
- **Link Register (LR)**
- **Program Counter (PC)**

**Current Program Status Register (CPSR)**

**Saved Program Status Register (SPSR)**

- CPSR
- SPSR FIQ
- SPSR IRQ
- SPSR SVC
- SPSR ABORT
- SPSR UNDEF
## Current Processor Status Register (CPSR)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>20</th>
<th>19</th>
<th>16</th>
<th>15</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
</table>

- **Condition Code Flags**
  - N = ALU Negative result
  - Z = ALU Zero result
  - C = ALU Carry out
  - V = ALU arithmetic overflow
  - Q = ALU sticky overflow
- J = Java state bit (always reads 0)
- IT[7:0] = If-Then
- DNM (Do Not Modify)
- GE[3:0] = Greater Than or Equal To
- E = Endianism of Data

- **A** = Imprecise Abort Disable
- **Interrupt Disable bits**
  - I = 1, disables the IRQ
  - F = 1, disables the FIQ
- **State bit**
  - T = 0, 32-bit instruction set
  - T = 1, 16-bit instruction set
- **Mode (defines processor mode)**
  - M[4:0] = 10000  User mode
  - M[4:0] = 10001  FIQ mode
  - M[4:0] = 10010  IRQ mode
  - M[4:0] = 10011  Supervisor mode
  - M[4:0] = 10111  Abort mode
  - M[4:0] = 11011  Undefined mode
  - M[4:0] = 11111  System mode
Supported data types

• The processor supports the following data types:
  – Double Word (64 bit)
  – Word (32 bit)
  – Half Word (16 bit)
  – Byte (8 bit)

• Although the processor supports unaligned accesses, TI does not recommend using unaligned accesses for bus performance.
  – Above data types should be aligned at their respective size boundary.
  – Most unaligned accesses are converted into multiple aligned accesses.

• The TMS570 devices are storing their data in big endian format (BE32), and RM4/5x stores data in little endian format.
Exception handling and the vector table

<table>
<thead>
<tr>
<th>Vector Table Offset</th>
<th>Exception</th>
<th>Mode on Entry</th>
<th>Bits in CPSR</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Reset</td>
<td>SuperVisor</td>
<td>A: Set</td>
<td>F: Set</td>
<td>I: Set</td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>UNDEF</td>
<td>Undefined (Mode)</td>
<td>Unchanged</td>
<td>Unchanged</td>
<td>Unchanged</td>
<td>Set</td>
</tr>
<tr>
<td>0x08</td>
<td>SVC</td>
<td>SuperVisor</td>
<td>Unchanged</td>
<td>Unchanged</td>
<td>Unchanged</td>
<td>Set</td>
</tr>
<tr>
<td>0x0C</td>
<td>PABT</td>
<td>Abort</td>
<td>Set</td>
<td>Unchanged</td>
<td>Set</td>
<td>Set</td>
</tr>
<tr>
<td>0x10</td>
<td>DABT</td>
<td>Abort</td>
<td>Set</td>
<td>Unchanged</td>
<td>Set</td>
<td>Set</td>
</tr>
<tr>
<td>0x14</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>IRQ</td>
<td>IRQ</td>
<td>Set</td>
<td>Unchanged</td>
<td>Set</td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>FIQ</td>
<td>FIQ</td>
<td>Set</td>
<td>Set</td>
<td>Set</td>
<td></td>
</tr>
</tbody>
</table>

- Reset: Highest Priority
- DABT
- FIQ
- IRQ
- UNDEF
- PABT
- SVC: Lowest Priority

- The FIQ is implemented as a non-maskable interrupt in the Hercules MCU.
- nFIQ and nIRQ inputs are connected to VIM.
Exception handling

When an exception occurs, the CPU does the following:

- Copies CPSR into SPSR_<mode>
- Sets appropriate CPSR bits
  - Change to ARM state
  - Change to exception mode
  - When an IRQ interrupt is received, the CPU disables other IRQ interrupts
  - When an FIQ interrupt is received, the CPU disables both IRQ and FIQ interrupts
- Stores the return address in LR_<mode>
- Sets PC to vector address or ISR address
- Swaps banked registers

To return, the exception handler needs to:

- Restore CPSR from SPSR_<mode>
- Restore PC from LR_<mode>
Interrupt handling

The Hercules MCU supports three different modes to handle peripheral interrupts in hardware and software:

- **Index interrupts mode (legacy mode):** The interrupt dispatching has to be done completely in software (software dispatcher).

- **Register-vectored interrupt mode:** This mode allows the interrupt dispatching to be done in hardware, the software has only to load the interrupt vector of the ISR from the VIM module and branch to the vector.

- **Hardware-vectored interrupt mode (IRQ only):** This mode has the advantage that the vector of the ISR has not been loaded by software. Instead, the vector is directly supplied to the MCU core via the VIC port, and saves some CPU cycles for lower interrupt latency compared to the second mode.
### Index interrupts mode

1. Events occur within peripherals.
2. Peripherals make FIQ/IRQ requests to the VIM.
3. VIM prioritizes the requests and provides the highest ISR to CPU.
4. CPU fetch from 0x18/0x1C.
5. Branch to ISR dispatcher.
7. Decide which ISR to execute.
8. Branch to ISR.

[Diagram showing the flow of events from peripheral interrupt to processor execution.]

---

**Peripheral Interrupt**

**VIM:** Prioritizing and signaling IRQ/FIQ to ARM Cortex-R4

**ARM Cortex-R4:** Executing IRQ/FIQ dispatcher, getting interrupt routine vector from table in local memory

**ARM Cortex-R4:** Executing actual interrupt routine
Register-vectored interrupts

1. Events occur within peripherals
2. Peripherals make FIQ/IRQ requests to the VIM
3. VIM prioritizes the requests & provides the addr of the highest ISR to CPU
4. CPU fetches from 0x18/0x1C
5. Branch to ISR (`LDR PC, [PC, #-0x1B0]`), (IRQVECREG/FIQVECREG.)
6. Branch to ISR

[Diagram showing the process of vectored interrupts]
1. Events occur within peripherals
2. Peripherals make FIQ/IRQ requests to the VIM
3. VIM prioritizes the requests
4. VIM provides address of highest pending request directly to the processors VIC port.
5. CPU branches directly to ISR.
Abort: Prefetch and Data

Prefetch Abort (PABT)
• CPU tries to execute an instruction from a protected or faulty memory location, such as:
  – The memory location is not implemented in the system.
  – The memory region is protected by the MPU.
  – An error is detected in the data by the ECC checking logic.
• All prefetch aborts are precise.

Data Abort (DABT)
• The CPU takes the data abort if data is read from or written to a protected or faulty memory location. This could be because of the following conditions:
  – The memory location is not implemented.
  – The memory location is read- or write-only in privileged mode (when processor is in User mode).
  – The memory location is read- or write-protected by the MPU.
  – If an error is detected in the data by the ECC checking logic.
• Data aborts can be precise or imprecise.
Abort type: Precise (synchronous), Imprecise (asynchronous)

**Precise or Synchronous Aborts**

- The abort is taken at the instruction that caused the exception.
- The abort handler could use the SPSR_abt and R14_abt (LR_abt) registers to determine the instruction that generated the abort and the CPU state when the abort occurred.
- Prefetch abort is always a precise abort.

**Imprecise or Asynchronous Aborts**

- If the exception is taken on an instruction later than the instruction that caused the exception.
- It is not possible to determine the exact instruction that caused the abort.
- This could be the case on writes to normal-type memory, where the write is stored in a buffer until the memory system is ready to perform it. In such cases, the exception will be generated and the abort will be taken after the appropriate store instruction was executed by the processor.
How to determine the cause of an abort

• The Cortex-R4/5 processor has a system control coprocessor implemented: The CP15. The CP15 offers the possibility to readout additional information about an abort.

• Four registers in the CP15 hold information about the cause of an abort:
  – Data Fault Status Register
  – Auxiliary Fault Status Registers
  – Data Fault Address Register
  – Instruction Fault Address Register
For more information

- **SafeTI Web Page**: [www.ti.com/safeti](http://www.ti.com/safeti)
- **Hercules Web Page**: [www.ti.com/hercules](http://www.ti.com/hercules)
  - Data sheets
  - Application notes
  - Software & tools downloads and updates
  - Order evaluation and development kits

- **Hercules Safety Microcontrollers Training Series**
  [training.ti.com/hercules](http://training.ti.com/hercules)
  - Cortex-R Processor Architecture
  - Peripherals
  - Software
  - Functional Safety

- **For questions about this training, refer to the Engineer-2-Engineer Support Forum** [www.ti.com/hercules-support](http://www.ti.com/hercules-support)
  - News and announcements
  - Ask technical questions
  - Search for technical content