MultiZone Security for Arm Cortex-M

The quick and safe way to add security and separation

MultiZone® Security is the quick and safe way to add security and separation to your Arm® Cortex®-M offering. MultiZone software can retrofit existing designs. If you don't have TrustZone®, or if you require finer granularity than one secure area, you can take advantage of high security separation without the need for hardware and software redesign – eliminating the complexity associated with managing a hybrid hardware/software security scheme.

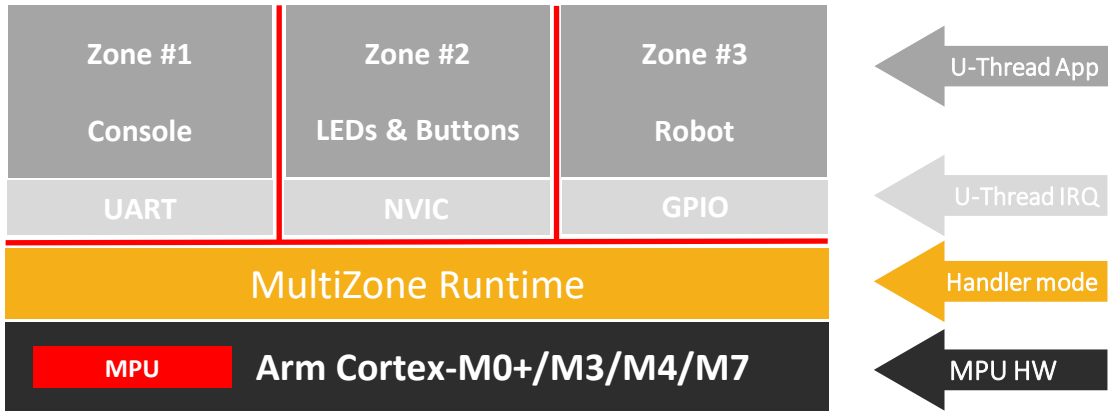
Securing microcontroller based applications has become increasingly difficult as complex new requirements are often met with the addition of readily available third party components that cannot always be trusted. Legacy designs lack the physical resources to provide the necessary separation, thus leading to larger attack surface and increased likelihood of vulnerability. To shield critical functionality from untrusted third-party components, MultiZone provides hardware-enforced, software-defined separation of multiple trusted execution environments. Unlike traditional solutions, MultiZone is self-contained, presents an extremely small attack surface, and it is policy driven, meaning that no coding is required – or in fact even allowed.

MultiZone enables the freedom to innovate with multi sourced components like open source software, third-party libraries, and legacy binaries that can be configured in minutes to achieve an unprecedented level of safety and security.

Technical Data

- Up to 8 separated Trusted Execution Environments (zones) – hardware-enforced, software-defined
- Up to 16 memory-mapped resources per zone – i.e. flash, ram, rom, i/o, uart, gpio, timers, etc.
- Preemptive scheduler for safety-critical applications: cooperative, round robin, configurable tick
- Secure inter-zone communications based on messages – no shared memory, no buffers, no stack, etc.
- Built-in trap & emulation for all privileged instructions – i.e. SVC, MRS, MSR, CPS, WFE, WFI
- Full support for secure user-mode interrupt handlers mapped to zones – up to 128 interrupt sources
- Full support for Wait For Interrupt and CPU suspend mode for low power applications
- Formally verifiable runtime ~2KB, 100% written in assembly, zero 3rd party dependencies
- C library wrapper for protected mode execution – optional for high speed / low-latency
- Hardware requirements: Arm Cortex-M0+/M3/M4/M7 processor w/ Memory Protection Unit
- System requirements: 4KB for program, 2KB for data - CPU overhead < 0.01%
- Development environment: any versions of Linux, Windows, Mac running Java 1.8 or greater
- GNU-based Open source SDK freely available at **https://github.com/hex-five/multizone-sdk-arm**

1 Break the firmware into separately linked binaries, one per zone



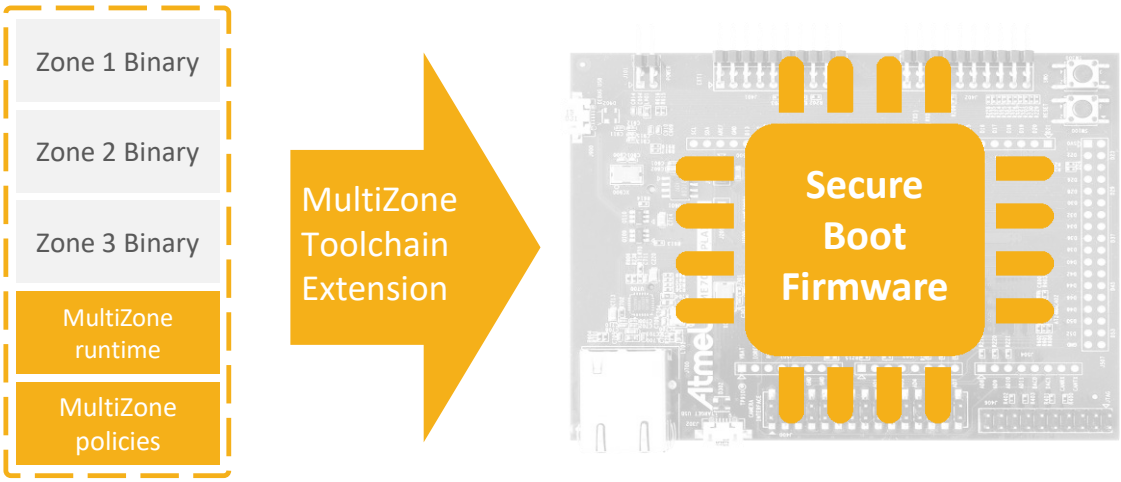
2 Define hardware separation policies mapping resources to zones

```
Zone 1   base = 0x00408000; size =   32K; rwx = rx # FLASH
         base = 0x2040F000; size =    4K; rwx = rw # RAM
         base = 0x40028000; size = 0x200; rwx = rw # UART
         base = 0x40088100; size =  0x20; rwx = rw # MATRIX
         base = 0x400E0600; size =  0x80; rwx = rw # PMC
         base = 0x400E0C00; size =  0x20; rwx = rw # EFC
         base = 0x400E0E00; size =  0x80; rwx = rw # PIOA (UART RX)
         base = 0x400E1000; size =  0x80; rwx = rw # PIOB (UART TX)

Zone 2   base = 0x00410000; size =   32K; rwx = rx # FLASH
         base = 0x20417000; size =    4K; rwx = rw # RAM
         base = 0x400E0E00; size = 0x100; rwx = rw # PIOA (BUTTON)
         base = 0x400E1200; size =  0x40; rwx = rw # PIOC (LED)

Zone 3   base = 0x00418000; size =   32K; rwx = rx # FLASH
         base = 0x2041F000; size =    4K; rwx = rw # RAM
         base = 0x400E0E00; size =  0x40; rwx = rw # PIOA (SPI_TDO)
         base = 0x400E1400; size =  0x80; rwx = rw # PIOD (SPI_TDI & SPI_TCK)
```

3 Run the toolchain extension to generate the secure boot firmware



0x5 HEX-Five Security