

# Introduction to the ARM<sup>®</sup> Cortex<sup>™</sup>-M Architecture

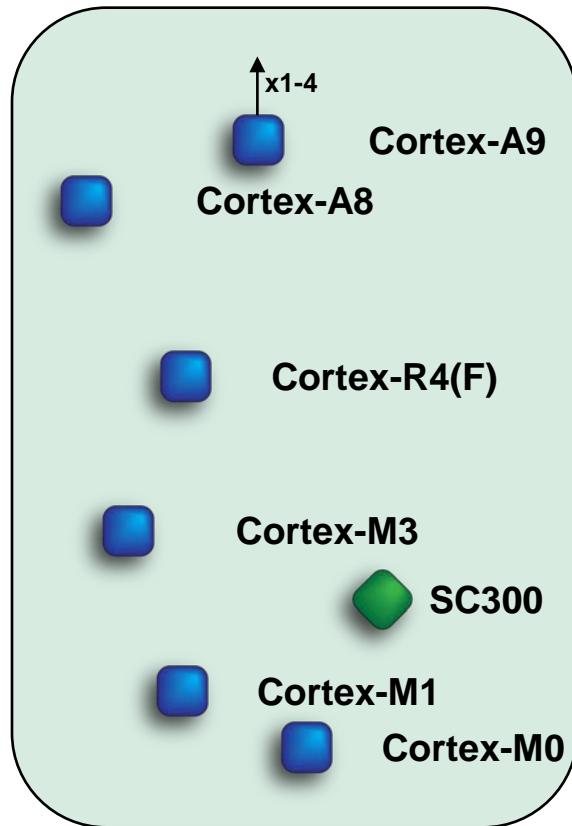
Delivering Cost-Sensitive High-Performance  
32-Bit Devices



# The ARM Cortex™ Family

# Cortex™

Intelligent Processors by ARM®



- **ARM Cortex A Series - Applications CPUs** focused on the execution of complex OS and user applications
- **ARM Cortex R Series - Deeply embedded processors** focused on **Real-time** environments
- **ARM Cortex M Series - Microcontroller cores** focused on very cost sensitive, deterministic, interrupt driven environments

---

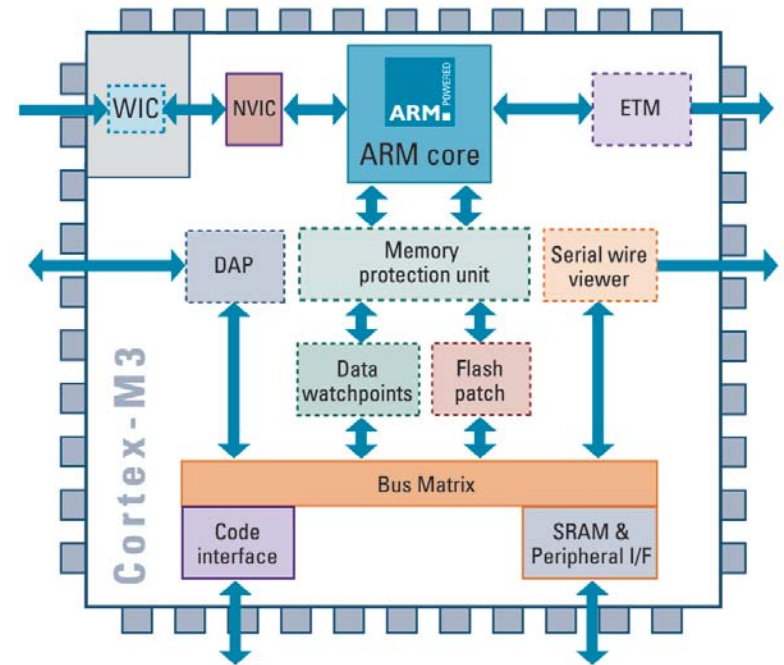
Introduction to the ARM® Cortex™-M Architecture

# CORTEX-M3

# Introduction to Cortex-M3 Processor

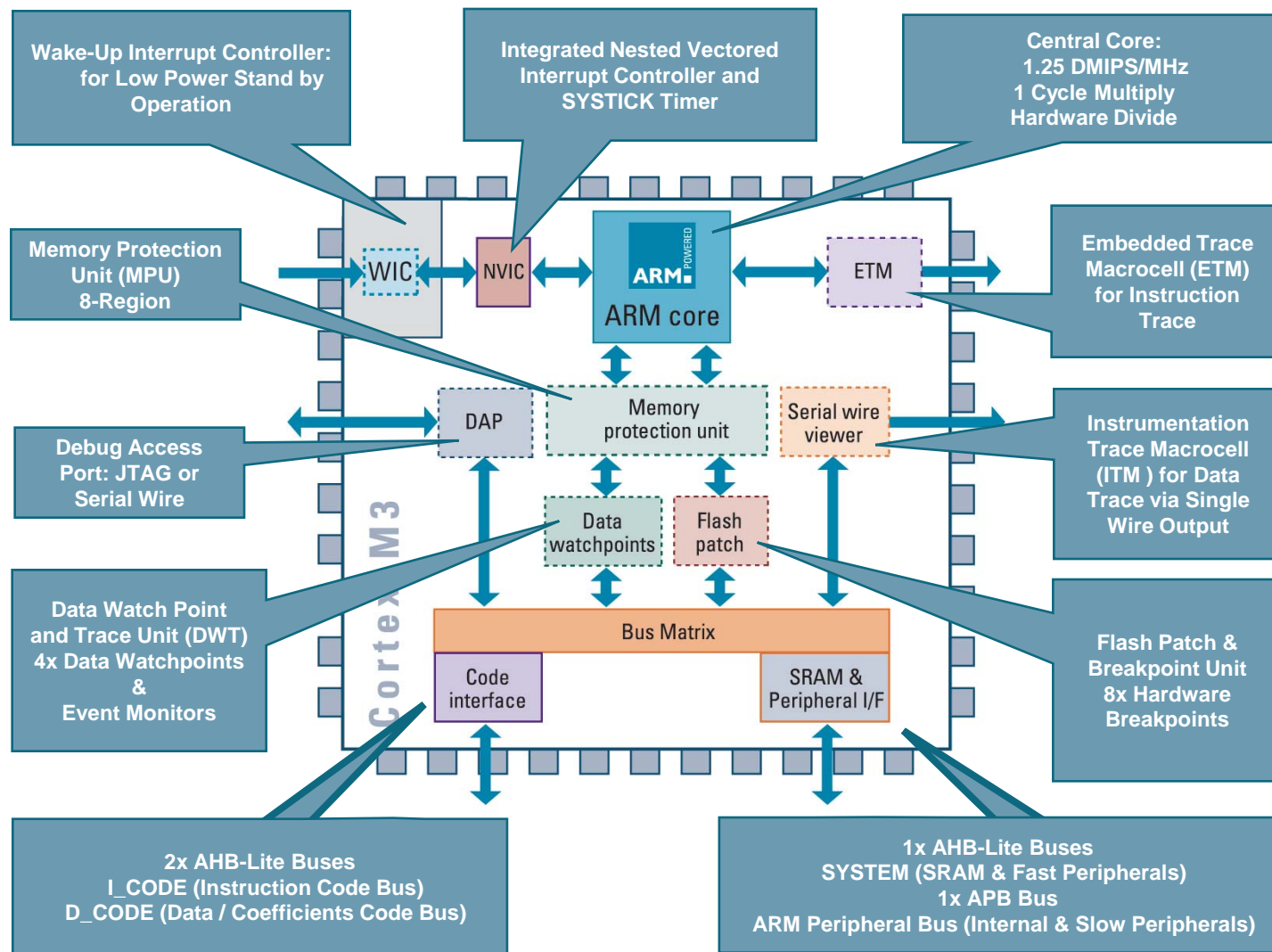
## ■ Cortex-M3 Architecture


- Harvard bus architecture
- 3-stage pipeline with branch speculation
- Integrated bus matrix
- Configurable nested vectored interrupt controller (NVIC)
- Advanced configurable debug and trace components
- Optional components for specific market requirements:
  - Wake-up Interrupt Controller (WIC)\*
  - Memory Protection Unit (MPU)
  - Embedded Trace Macrocell (ETM)
  - Fault Robust Interface\*



\* Cortex-M3 Release 2

# Cortex-M3 Processor Overview



 optional blocks, please consult your silicon manufacturers data sheet

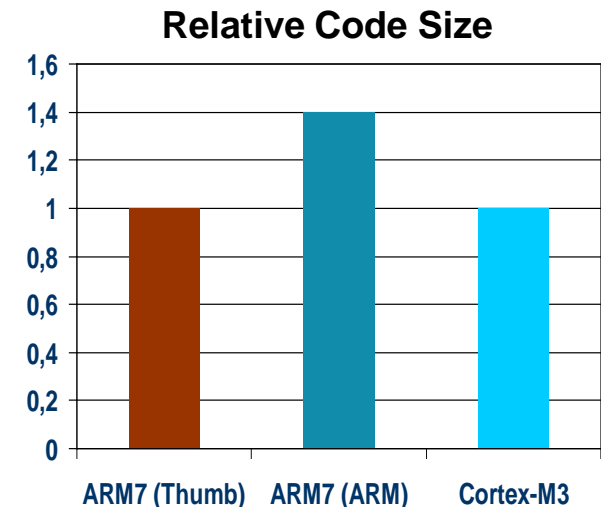
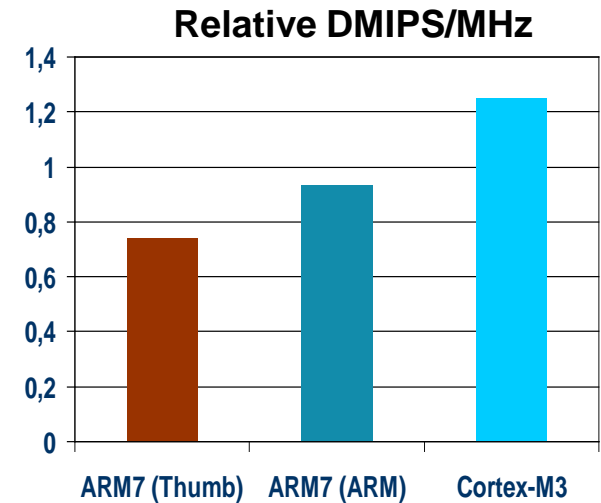
# ARM Cortex-M3 - Designed for Performance

## ■ High Performance

- High efficiency processor core – 1.25 DMIPS/MHz
- Advanced instructions for data manipulation
  - Single Cycle Multiply
  - Hardware Division
  - Bit Field Manipulation
- Exceptional performance at low frequency

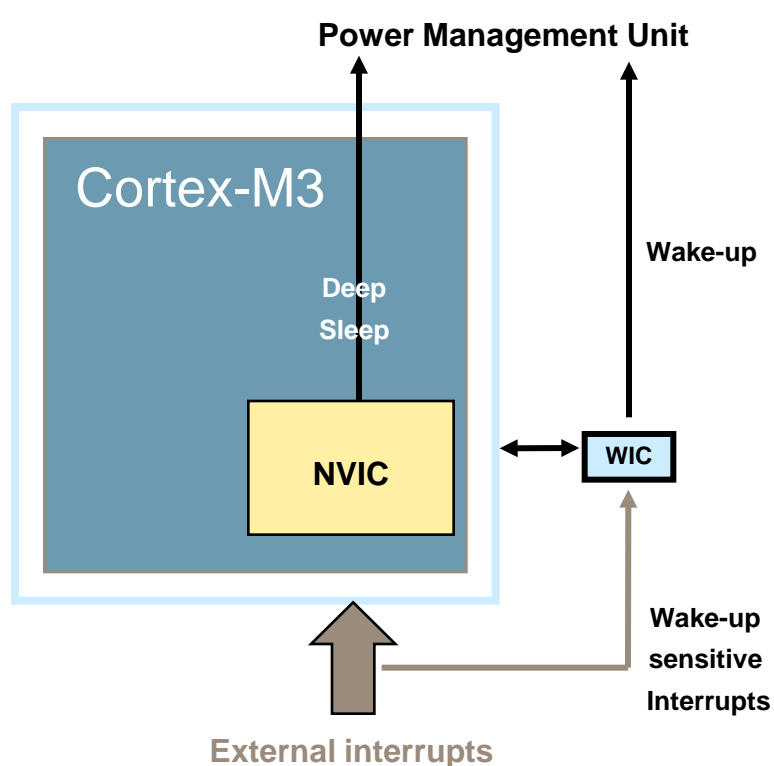
## ■ Excellent Code Density

- Thumb-2 Instruction Set Architecture (ISA)
- Optimized applications for performance and code size
  - 32-Bit code performance
  - 16-Bit code density
- No interworking required between code objects



# ARM Cortex-M3 - Designed for Low Power

- **Cortex-M3 features architected support for sleep states**
  - Enables ultra low-power standby operation
  - Critical for extended life battery based applications
  - Includes very low gate count Wake-Up Interrupt Controller (WIC)

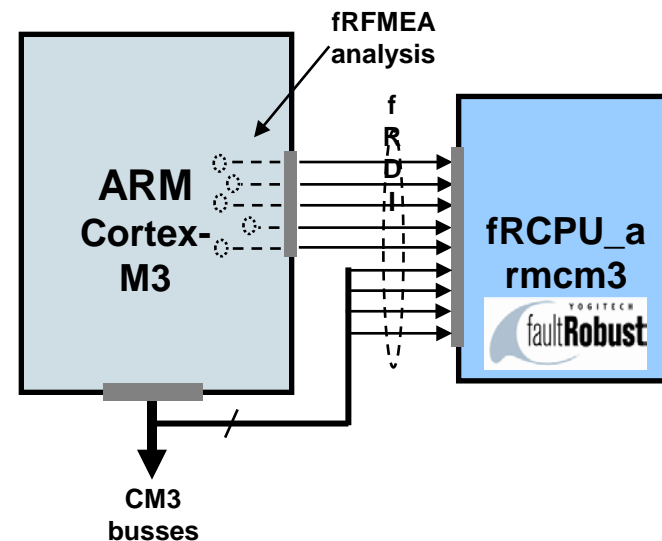
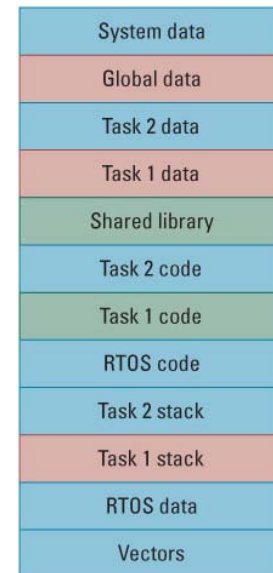


- **Sleep**
  - CPU can be clock gated
  - NVIC remains sensitive to interrupts
- **Deep sleep**
  - WIC remains sensitive to selected interrupts
  - Cortex-M3 inc. NVIC can be put into state retention
- **WIC signals wake-up to PMU**
  - Cortex-M3 can be woken almost instantaneously
  - React to critical external events

# ARM Cortex-M3 - Designed for Robustness

## Cortex-M3 supports design of robust applications

- **Processor Modes**
  - Separation of main and interrupt code
- **Privilege Levels**
  - Separation of RTOS and user application
- **NMI**
  - Inform processor of critical events
- **SYSTICK**
  - Protected system timer for pre-empting RTOS
- **Fixed Memory Map**
- **MPU\***
  - Separation of user application tasks
- **Fault Robust Observation Interface\***
  - IEC61508 standard SIL3 certification

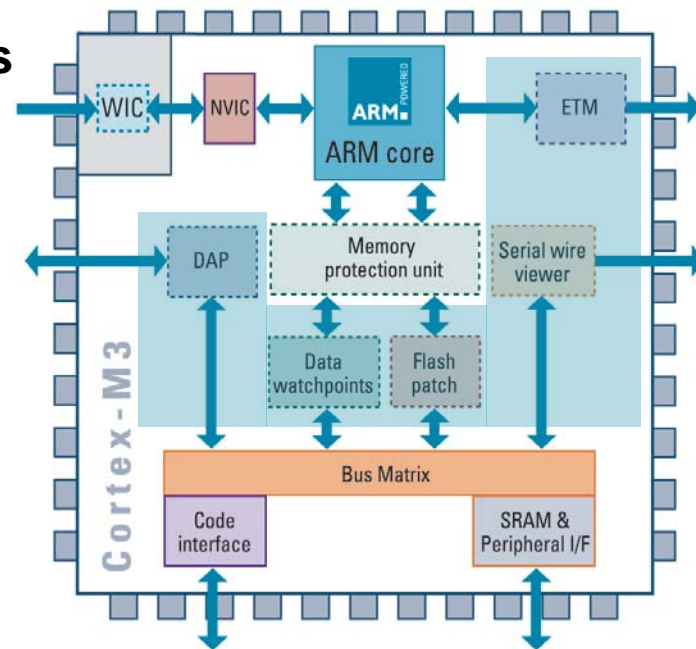


\*optional



# Coresight™ Debug & Trace

- **ARM CoreSight is a complete on-chip debug and real-time trace solution for the entire system-on-chip (SoC)**
- **Configurable to adapt for market requirements**
  - Debug components can be configured or even removed (check device manufacturer data sheet)
- **Enhanced debugging modes and features**
  - Up to 8 (6 Instructions + 2 Literal) HW Breakpoints
  - Debug Access Port (DAP) allows memory access while processor is running
  - Up to 4 Data Watch Points (DWT)
  - Event Counters (DWT)
- **Serial Wire Debug (SWD) Mode**
  - 2 wire interface
  - Offers extra functionality over JTAG using less I/O
- **Serial Wire Viewer (SWV)**
  - Real-time trace with no extra trace hardware, pins or silicon overhead
- **Embedded Trace Macrocell**
  - Instruction Trace using 4-bit port



---

Introduction to the ARM® Cortex™-M Architecture

# CORTEX-M1

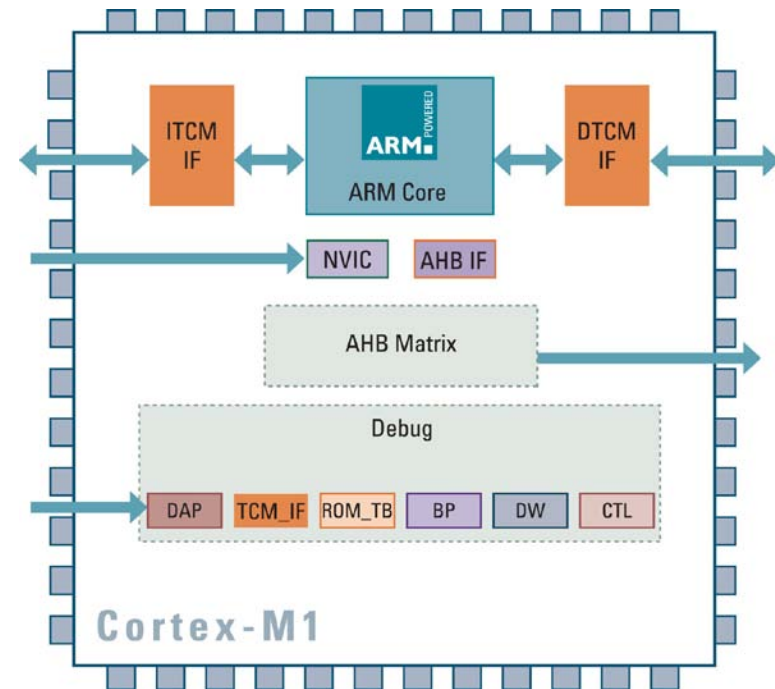
# ARM Cortex-M1

## ■ Soft processor for FPGA

- Upwards compatible with Cortex family on ASIC/ASSP/MCU
- 3 stage pipeline
- Delivers 0.8 DMIPS/MHz
- Capable of up to 200MHz

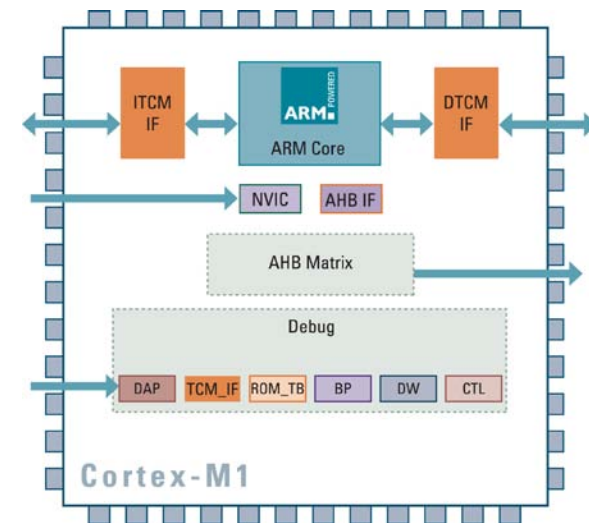
## ■ Designed for synthesis on multiple FPGA types, e.g:

- Actel ProASIC3, Actel Igloo and Actel Fusion
- Altera Cyclone-III, Altera Stratix-III
- Xilinx Spartan-3, Xilinx Virtex-5.



# ARM Cortex-M1 Processor Features

- **A 3-stage, 32-bit RISC processor**
  - Highly configurable to enable design trade-offs
  - Retains the same programmers model for software simplicity
- **Tightly Coupled Memories**
  - Internal FPGA block RAM used as single-cycle access memory
  - ITCM, DTCM configurable from 0k to 1024kBytes
- **Configurable debug**
  - JTAG or reduced pin-count SWD interface
  - Full – 2 watchpoints, 4 breakpoints
  - Small – 1 watchpoint, 2 breakpoints
  - None – removable for cost reduction and security



# ARM Cortex-M1 Processor Features (2)

## ■ Integrated Interrupt Controller

- Fast interrupt response
- Configurable 1, 8, 16, 32
- Software programmed priority levels (1-4)
- Non-Maskable Interrupt

## ■ Multiplier

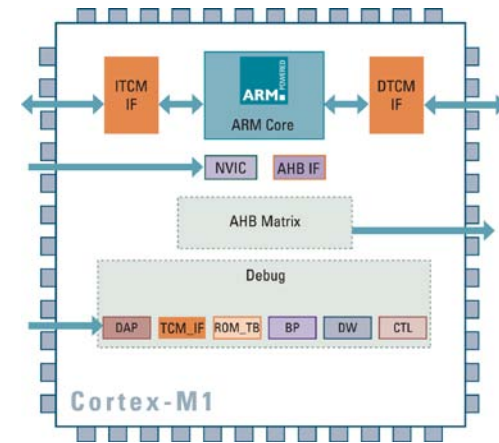
- Fast option uses FPGA DSP blocks
- Small option uses adder to save area, can use DSP blocks
- Program function is the same with either - no need for software modifications

## ■ AMBA AHB-lite 32-bit bus interface

- Connection to external memory and peripherals

## ■ Big or Little Endian

- Synthesis time configurable



# ARM Cortex-M1 Speed and Area

- **Results below are to give a guideline for MHz and area**
  - The nature of FPGA implementation means results may change per system
  - The results will also change as tools and FPGA evolve
- **These are speed targeted synthesis run results**
  - For smallest configuration (0k TCM, no debug)
  - Assuming fastest commercial speed grade

FPGA type	Example		Speed	Area (LUTS)
65nm	Altera Stratix III	Xilinx Virtex-5	200 MHz	1900
90nm	Altera Stratix II	Xilinx Virtex-4	150 MHz	2300
65nm	Altera Cyclone III		100 MHz	2900
90nm	Altera Cyclone II	Xilinx Spartan-3	80 MHz	2600
130nm	Actel ProASIC3	Actel Fusion	70 MHz	4300 tiles

# ARM Cortex-M1 Instruction Set

- Cortex-M1 implements an ISA based primarily on Thumb
  - The high density 16-bit ISA introduced in ARM7TDMI
- Cortex-M1 includes a few Thumb-2 system instructions
  - To allow operation in Thumb state only
  - Enables binary upwards compatible with Cortex-M3

## Thumb

User code, compiler generated

ADC	ADD	ADR	AND	ASR	B
BIC	BL		BX	CMN	CMP
EOR	LDM	LDR	LDRB	LDRH	LDRSB
LDRSH	LSL	LSR	MOV	MUL	MVN
NEG	ORR	POP	PUSH	ROR	RSB
SBC	STM	STR	STRB	STRH	SUB
SVC	TST	BKPT	BLX	CPS	CPY
REV	REV16	REVSH	SXTB	SXTH	UXTB
UXTH					

## Thumb-2

OS & system

NOP	
SEV	WFE
WFI	YIELD
DMB	
DSB	
ISB	
MRS	
MSR	

## Cortex-M1 ISA

---

Introduction to the ARM® Cortex™-M Architecture

# CORTEX-M0



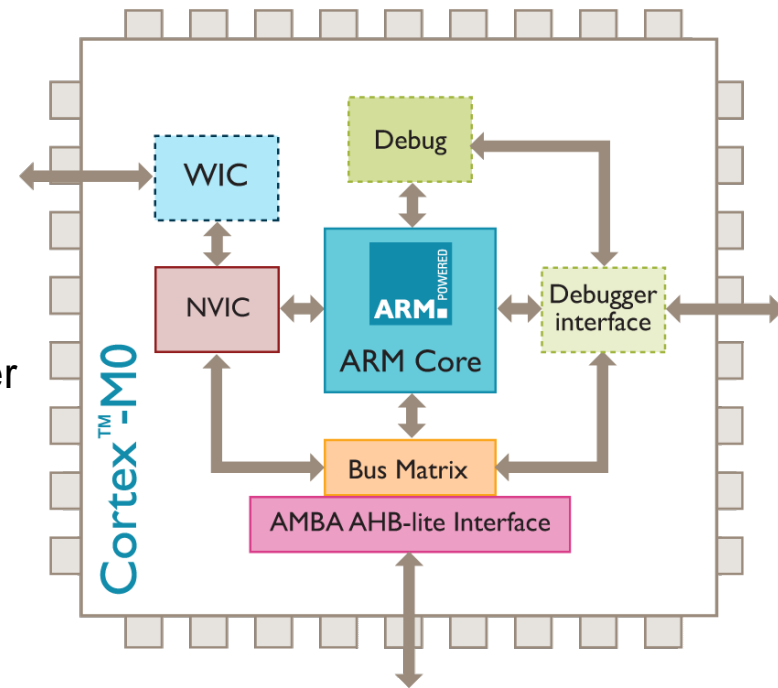
# ARM Cortex-M0 Processor

- **The smallest, lowest power ARM processor ever**

- A third of the area of ARM7TDMI-S
- 85  $\mu\text{W}/\text{MHz}$ , 12K gates \*

- Von-Neumann bus architecture
- 3-stage pipeline
- Delivers 0.9 DMIPS/MHz

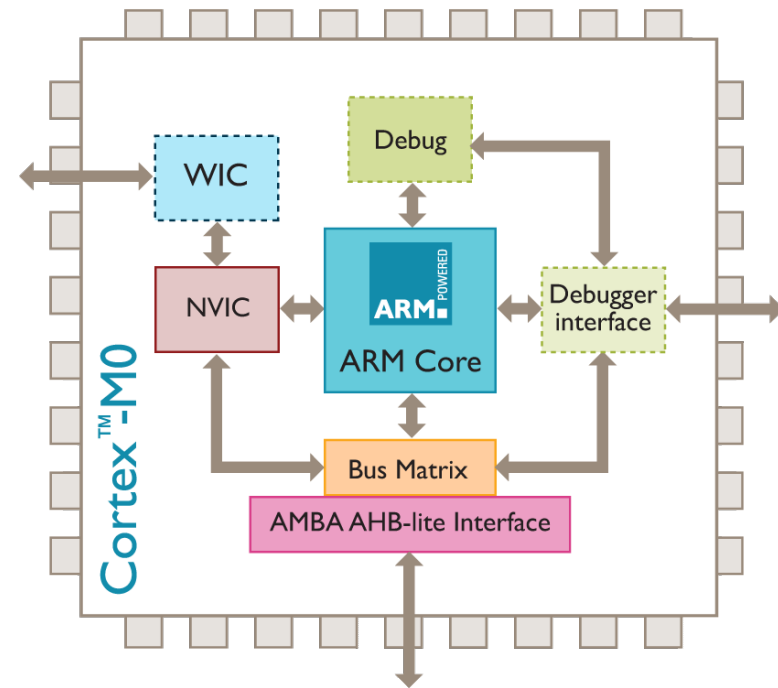
- Configurable nested vectored interrupt controller (NVIC)
- Optional Wake-up Interrupt Controller (WIC)
- Configurable Debug
- Binary and tools upwards compatible with ARM Cortex-M3 processor



\* Implemented on 180ULL with ARM Physical IP

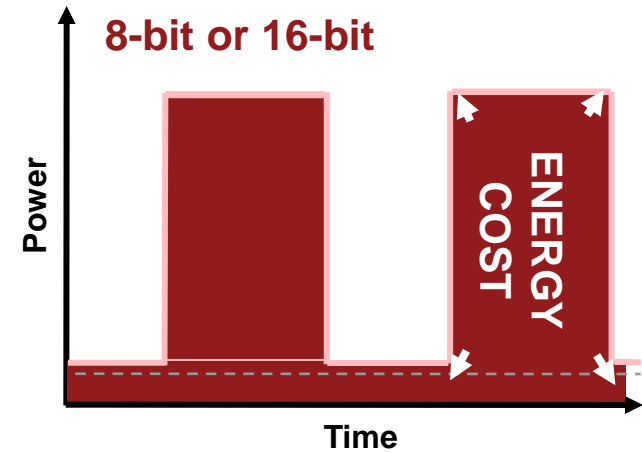
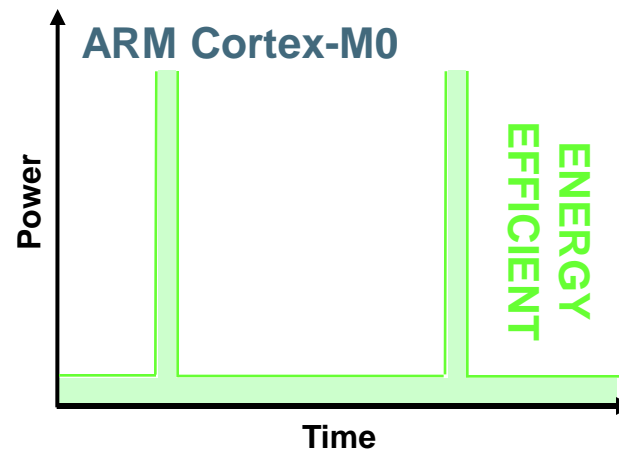
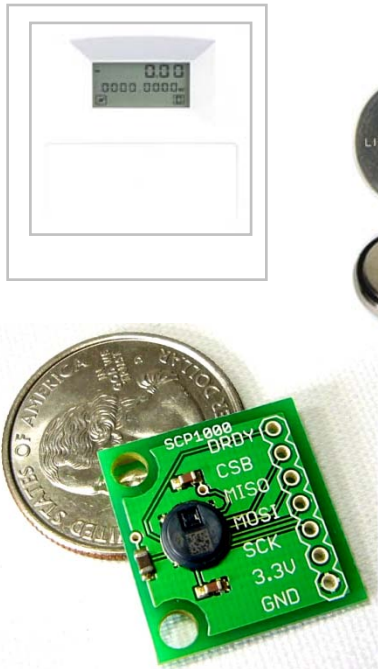
# ARM Cortex-M0 Processor Features

- **Cortex-M0 RTL is configurable**
  - Tune for your application
  - Check device manufacturer data sheet
- **Consistent programmer's model**
  - Software compatibility
  - All tools remain compatible
- **Integrated Interrupt Controller (NVIC)**
  - 1, 8, 16, 24 or 32 interrupts
- **Multiplier options**
  - Fast or small (1 or 32 cycle)
- **Optional OS extensions**
  - SYSTICK Timer
  - PendSV (Pending System Call)
- **Configurable debug**
  - 4 or 2 breakpoints, 2 or 1 watchpoints
  - JTAG or SWD interface



# Energy Efficiency

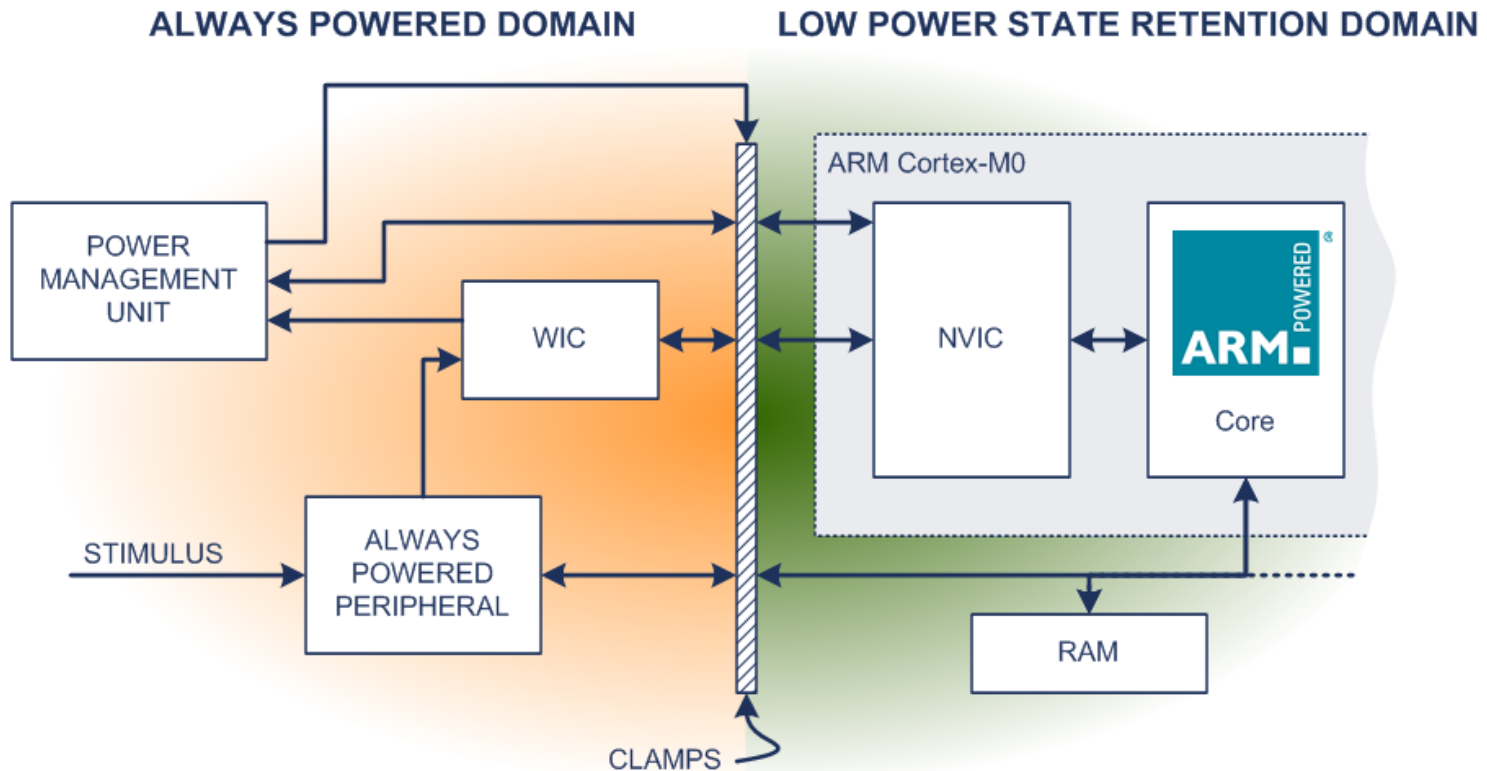
- **Cortex-M0 designed for excellent power efficiency**
  - Significantly less activity required to match 8/16-bit device performance
  - Fast interrupt response minimizes time in active state
- **Architected for ultra low power deep sleep**
  - Excellent static power results using ARM Physical IP Metro 180ULL libraries and PMK
  - Easy integration to power management unit via Wake-up Interrupt Controller



Lower energy for an identical task

# Architected Sleep States

- Cortex-M0 processor supports ultra low-power standby implementation
- Critical for extended life battery-based applications
- Includes very low gate count Wake-Up Interrupt Controller (WIC)



# ARM Cortex-M0 Instruction Set

- Cortex-M0 implements an ISA based primarily on Thumb
  - The high density 16-bit ISA introduced in ARM7TDMI
- Cortex-M0 includes a few Thumb-2 system instructions
  - To allow operation in Thumb state only
  - Enables binary upwards compatible with Cortex-M3

## Thumb

User code, compiler generated

ADC	ADD	ADR	AND	ASR	B
BIC	BL		BX	CMN	CMP
EOR	LDM	LDR	LDRB	LDRH	LDRSB
LDRSH	LSL	LSR	MOV	MUL	MVN
NEG	ORR	POP	PUSH	ROR	RSB
SBC	STM	STR	STRB	STRH	SUB
SVC	TST	BKPT	BLX	CPS	CPY
REV	REV16	REVSH	SXTB	SXTH	UXTB
UXTH					

## Thumb-2

OS & system

NOP	
SEV	WFE
WFI	YIELD
DMB	
DSB	
ISB	
MRS	
MSR	

Cortex-M0 ISA

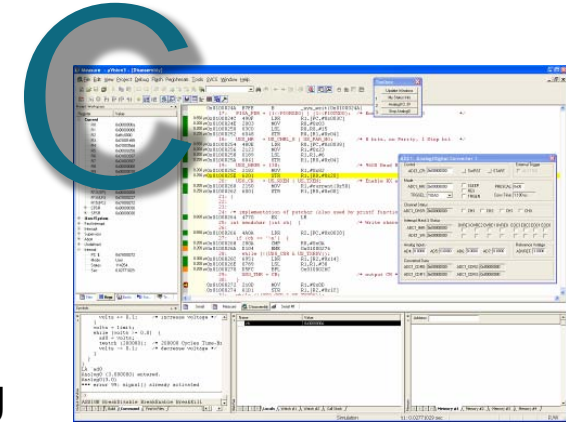
---

Introduction to the ARM® Cortex™-M Architecture

# CORTEX-M - ARCHITECTURE

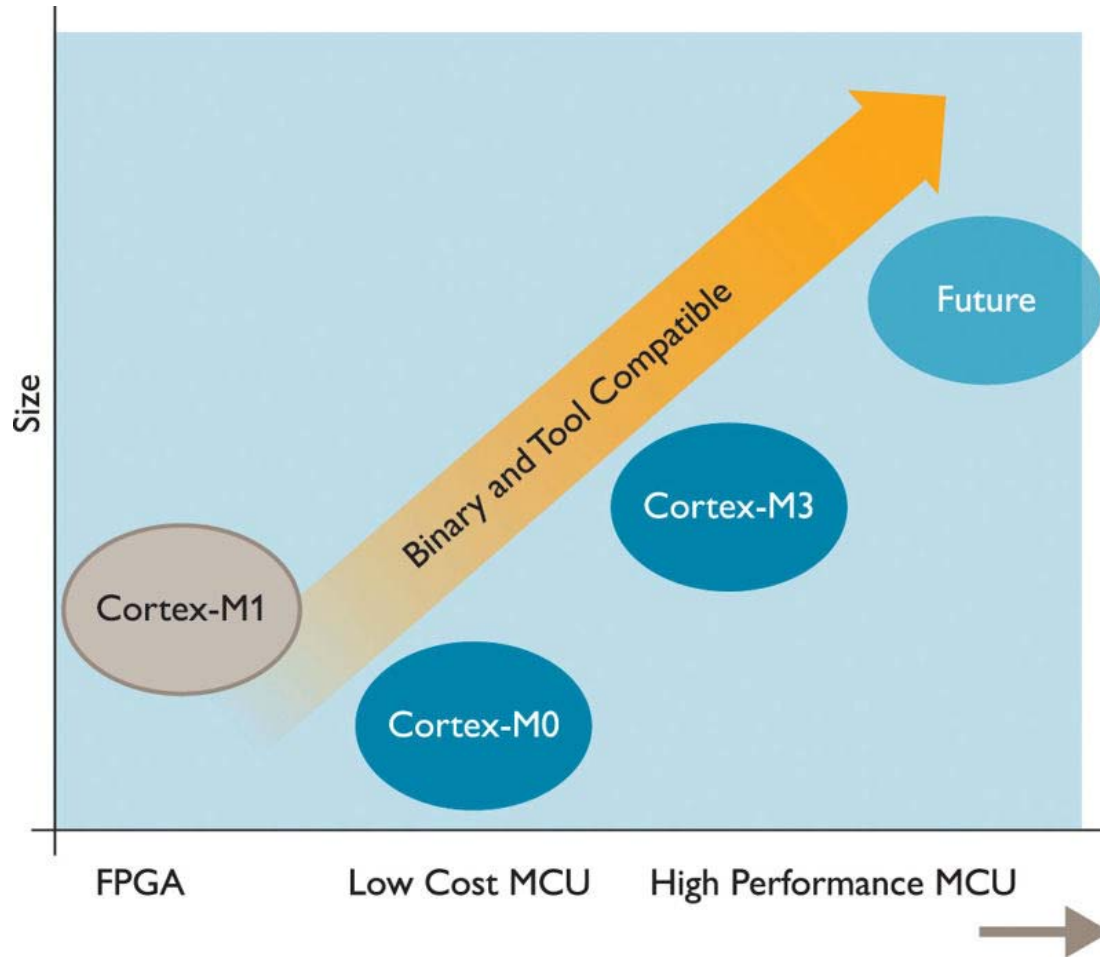
# ARM Cortex-M - Designed for Ease of Use

- Virtually everything can be written in C/C++
- No need for assembler in top level interrupt handlers
- Easy to use atomic bit twiddling
- Fast, fully deterministic ISR entry with hardware stacking on interrupt entry
- Simpler programmer's model with state manipulation handled in hardware
- Memory Map, NMI and SYSTICK defined and integrated enabling better code reuse
  - Eases portability of applications and RTOS



# ARM Cortex-M Processors

- Cortex-M family optimised for deeply embedded
  - Microcontroller and low-power applications







# Processor Mode

- **Handler Mode**

- Used to handle exceptions. The processor returns to Thread mode when it has finished exception processing.

- **Thread Mode**

- Used to execute application software. The processor enters Thread mode when it comes out of reset.
- In Thread mode, the CONTROL register controls whether software execution is privileged or unprivileged, see CONTROL register. In Handler mode, software execution is always privileged.
- RTX is using processor modes

## ■ Unprivileged

The software:

- has limited access to the MSR and MRS instructions, and cannot use the CPS instruction
- cannot access the system timer, NVIC, or system control block
- might have restricted access to memory or peripherals.
- Unprivileged software executes at the unprivileged level

## ■ Privileged

- The software can use all the instructions and has access to all resources.

Not available in Cortex-M0 / Cortex-M1

# Register File

Thread/Handler	Thread
R0	_____
R1	_____
R2	_____
R3	_____
R4	_____
R5	_____
R6	_____
R7	_____
R8	_____
R9	_____
R10	_____
R11	_____
R12	_____
R13 (MSP)	R13 (PSP)
R14 (LR)	_____
PC	_____
PSR	_____
PRIMASK	
FAULTMASK*	
BASEPRI*	
CONTROL	

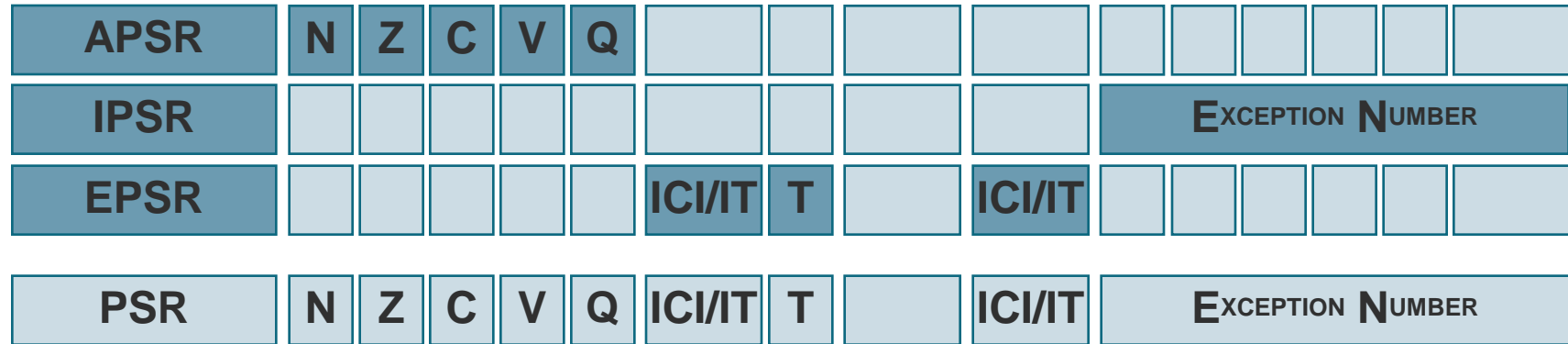
- **All registers are 32-bit wide**
- **13 general purpose registers**
  - R0 - R7 are accessible by any instruction
  - R8 - R12 are accessible to a few 16-bit instructions and to all 32-bit instructions
- **3 registers with special meaning/usage**
  - R13 - Stack Pointer (SP)
    - 2 banked copies - Main and Process
  - R14 - Link Register (LR)
  - R15 - Program Counter (PC)
- **Special-purpose registers**
  - PSR
  - PRIMASK
  - FAULTMASK
  - BASEPRI
  - CONTROL

\*Not available in Cortex-M0 / Cortex-M1



# Program Status Registers

- **PSR - Program Status Register combines**
  - APSR - Application Program Status Register
    - Negative, Zero, Carry, OVerflow, Q-Sticky Saturation Flag
  - IPSR - Interrupt Program Status Register
    - ICI/IT – Interrupt Continuable Instruction, IF-THEN instruction status
    - Thumb – Always 1
  - EPSR - Execution Program Status Register
    - Exception Number – Indicates which exception processor is handling
- **CPU Instructions MSR and MRS allow access (together or separate)**
  - For example:           MSR PSR, r0  
                              MRS r1, IPSR



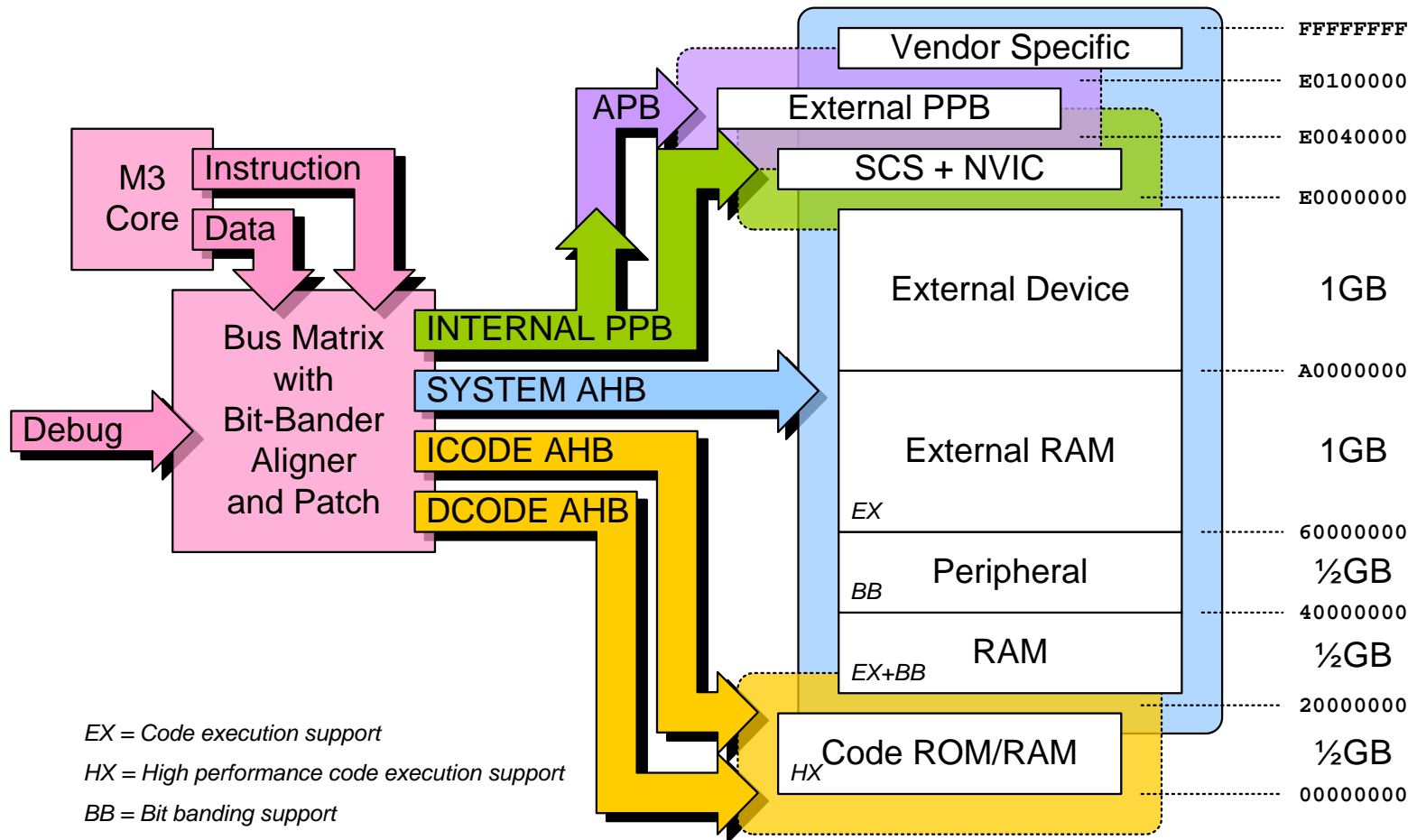


# Memory Access

- **The Cortex-M3 has an internal Harvard architecture**
  - Separate instruction and data interfaces
- **The internal bus matrix converts the internal Harvard to 3 AHB-Lite interfaces**
  - I-Code – Instruction accesses to the Code memory space
  - D-Code – Data accesses to Code memory space
  - System – All accesses to System memory space
- **In a typical system:**
  - Instructions stored in Code space (in Flash)
  - SRAM accessed across System bus
- **This allows the interrupt latency to be minimized**
  - The exception vector is fetched over the ICode bus
  - In parallel, the processor state is saved over the System bus

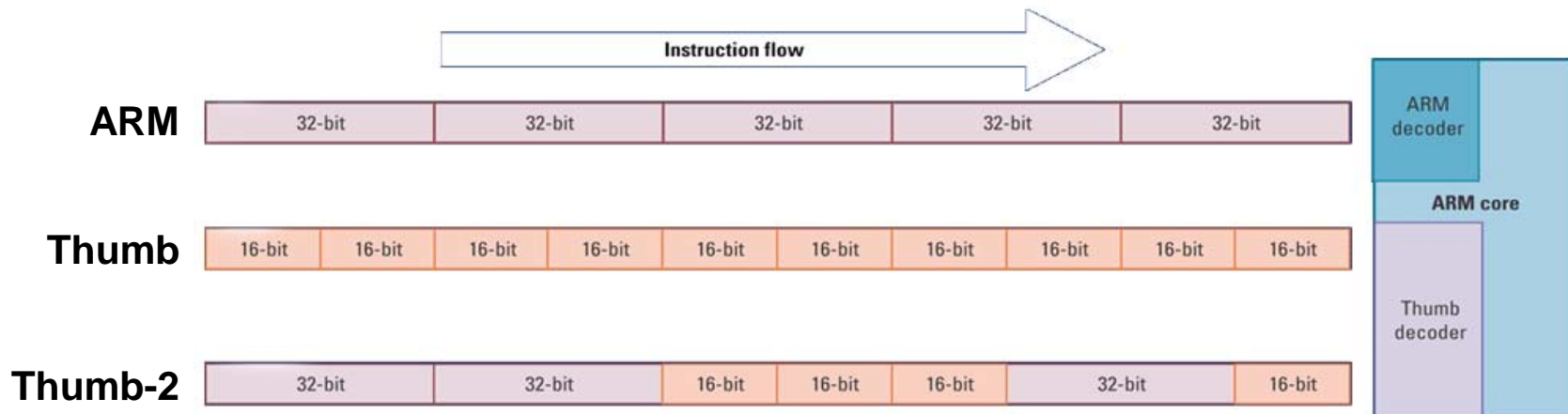
# Memory Map

- Linear 4GB memory map
- Fixed map required to host system components and simplify implementation
- Bus Matrix partitions memory access via the AHB and PPB buses



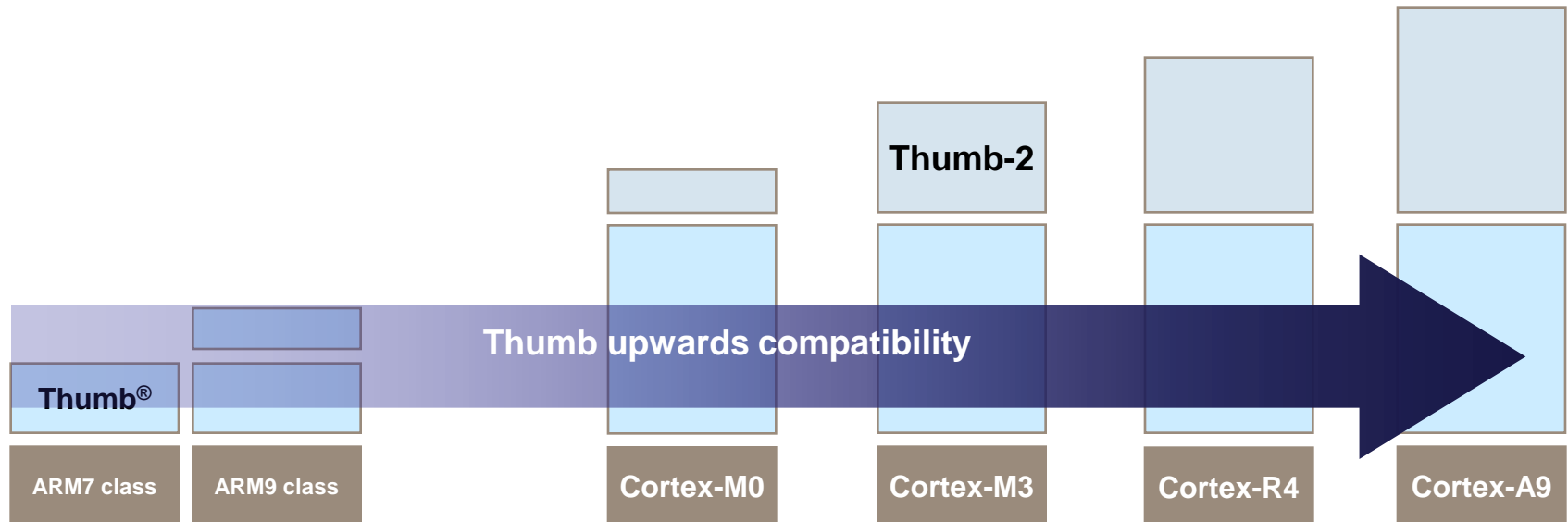
# Thumb-2 Technology

- Thumb-2 ISA was introduced in ARMv7 architecture
  - Original 16-bit Thumb instructions maintain full compatibility with existing code
  - +
  - New 16-bit Thumb instructions for improved program flow
  - +
  - New 32-bit Thumb instructions for improved performance and code size.  
One 32-bit instruction replaces multiple 16-bit opcodes.  
32-bit instructions are handled in the same mode ~ no interworking required



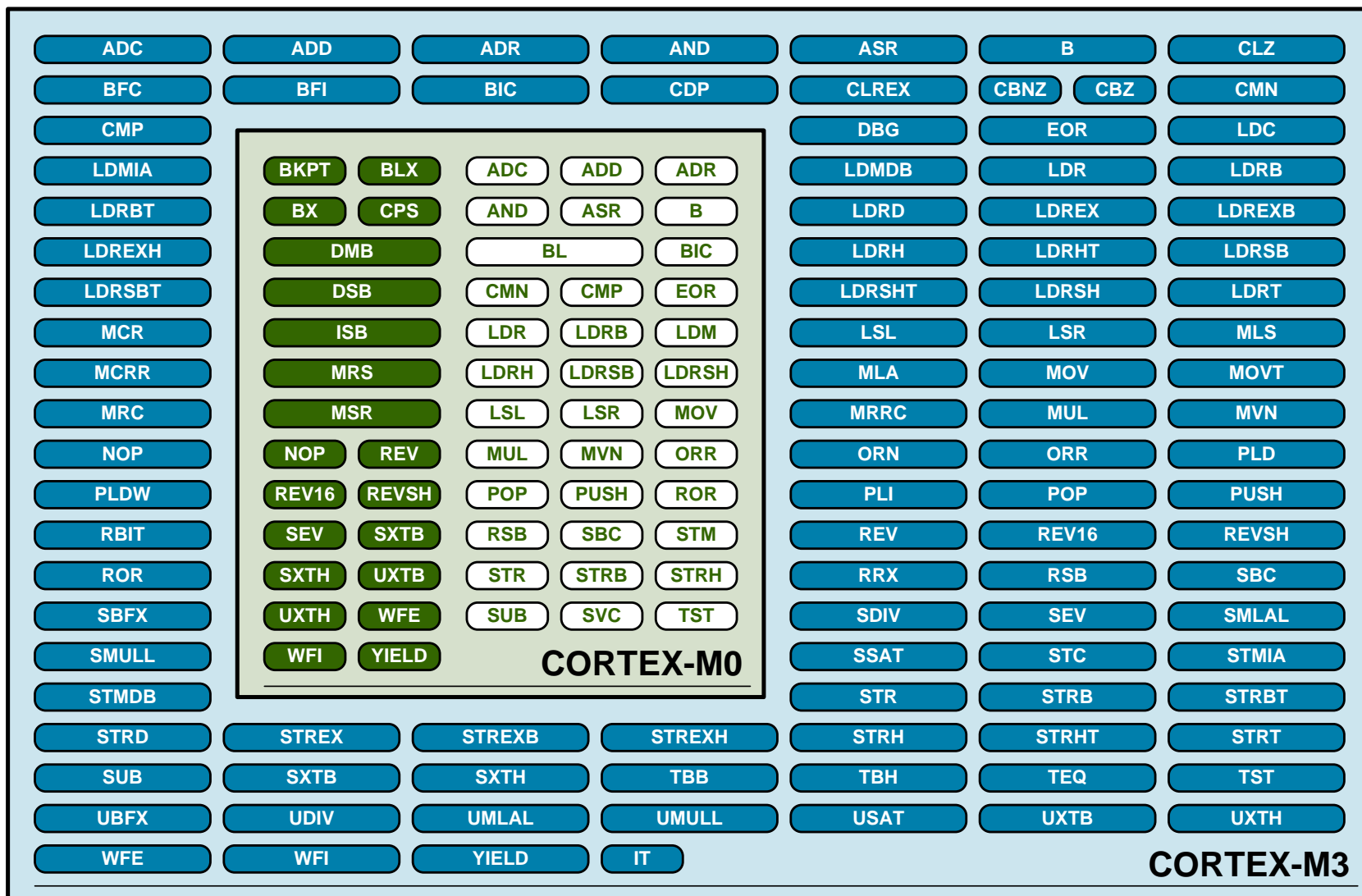
# Software Compatibility

- **Cortex-M0/M1 implements an ISA based primarily on Thumb**
  - The high-density 16-bit instruction set introduced in ARM7TDMI
- **Cortex-M0/M1 includes a few Thumb-2 system instructions**
  - Enables Cortex-M0/M1 to operate in Thumb state only
  - Enables binary upwards compatible with Cortex-M3
- **Cortex-M3 implements Thumb-2**





# Instruction Set Comparison



Present in ARM7TDMI

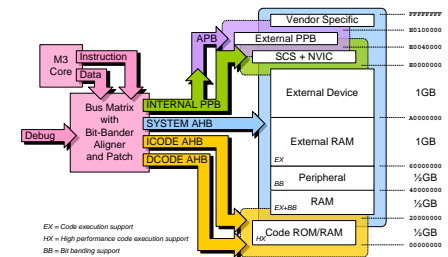
# Nested Vector Interrupt Controller

- **NVIC is a core peripheral**
  - Consistent between Cortex-M cores
  - Tailored towards fast and efficient interrupt handling
- **Number of interrupts can be configured by device manufacturer**
  - 1 ... 240 interrupt channels for M3
  - 1 ... 32 interrupt channels for M0
  - Each peripheral has its own interrupt vector(s)
- **8 – 256 interrupt priorities (1 - 4 Cortex M1/M0)**
- **Configured via memory-mapped control registers**
- **Exceptions/interrupts processed in Handler mode**
  - Supervisor privilege
- **Interruptible LDM/STM (and PUSH/POP) for low interrupt latency**
  - Continued on return from interrupt
- **Non Maskable Interrupt (NMI)**

# Nested Vector Interrupt Controller

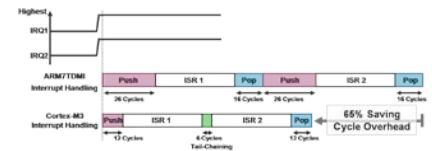
## ■ When an interrupt occurs:

- The exception vector is fetched over the ICODE bus
- In parallel, the processor state is saved over the SYSTEM bus
- Automatic save and restore of processor state
  - {PC, xPSR, R0-R3, R12, R14}
  - Provides low latency interrupt/exception entry and exit
  - Allows handler to be written entirely in 'C'



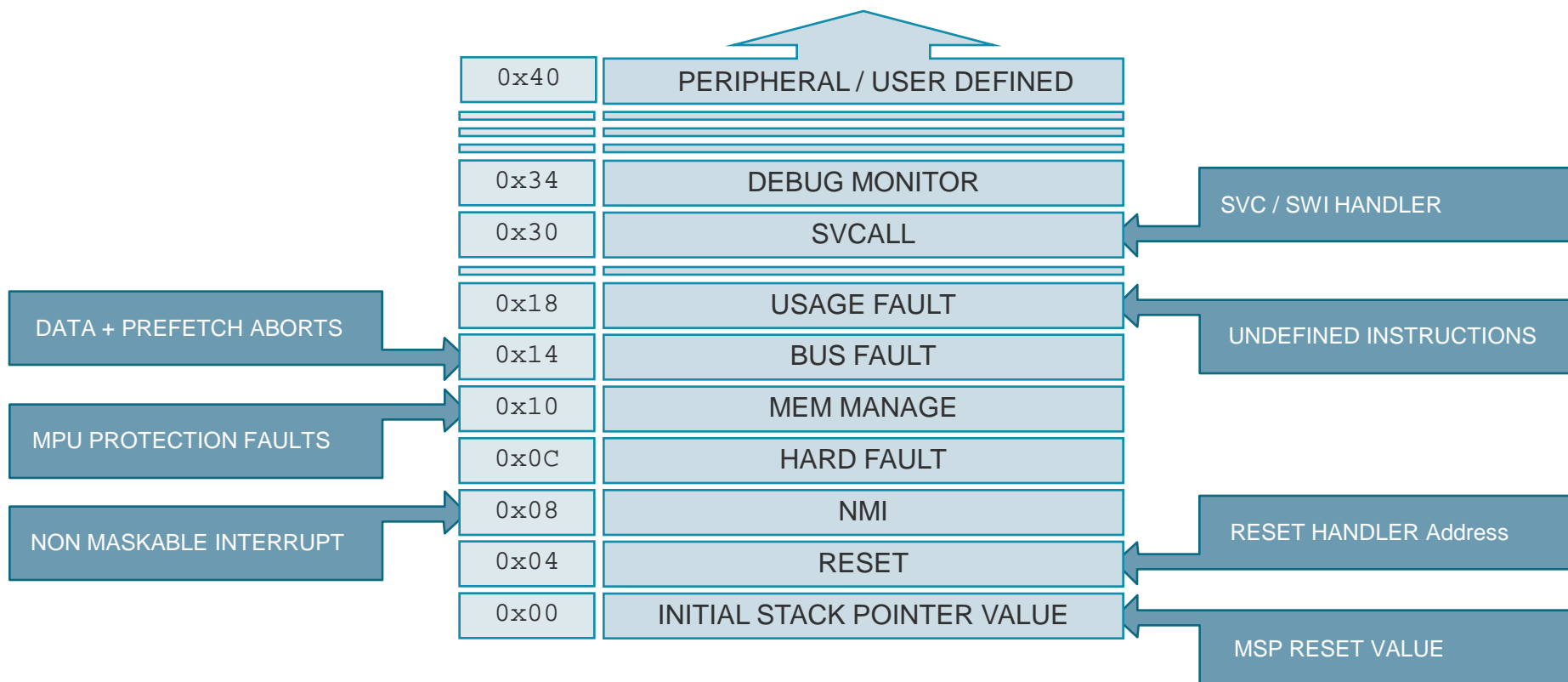
## ■ Interrupt Latency for Cortex M3 max. 12 Cycles

## ■ Interrupt Latency into pending interrupts 6 Cycles



# Vector Table

- **ARMv7M architecture implements a re-locatable vector table**
  - Contains initial stack pointer value
  - Contains the address of RESET handler
  - Contains the address of the function to execute for a particular handler
  - The first sixteen entries are special with the others mapping to specific interrupts



# Exception & Pre-emption Ordering

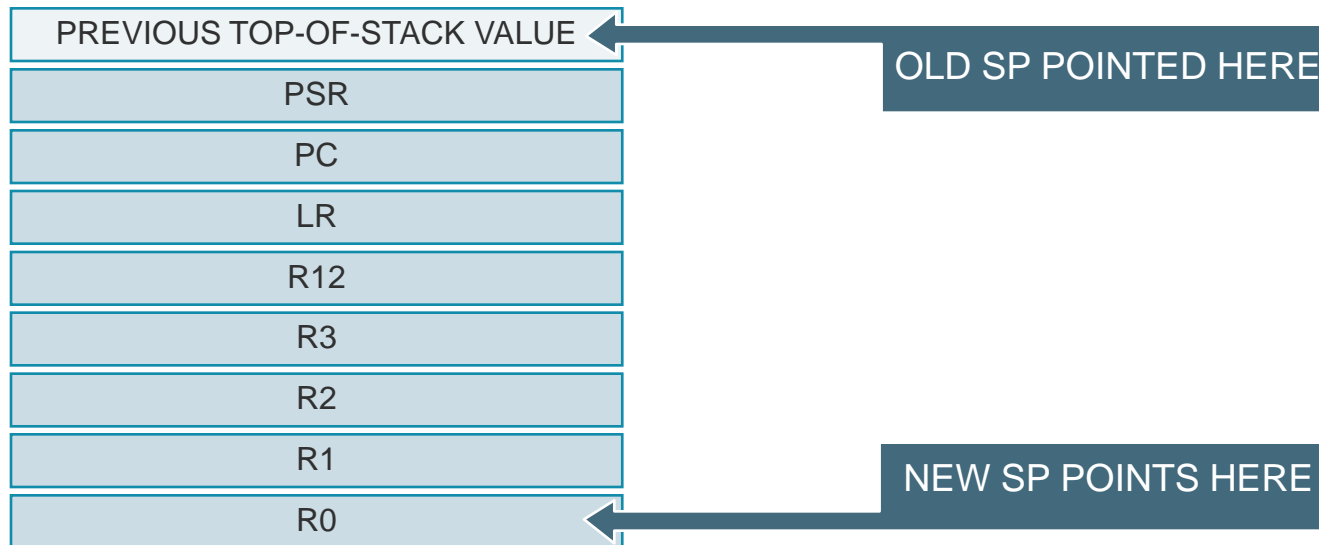
- **Exception handling order is defined by programmable priority**
  - Reset, Non Maskable Interrupt (NMI) and Hard Fault have predefined pre-emption.
  - NVIC catches exceptions and pre-empts current task based on priority

	Exception	Name	Priority	Descriptions
Fault Mode & Start-up Handlers	1	Reset	-3 (Highest)	Reset
	2	NMI	-2	Non-Maskable Interrupt
	3	Hard Fault	-1	Default fault if other handler not implemented
	4	MemManage Fault	Programmable	MPU violation or access to illegal locations
	5	Bus Fault	Programmable	Fault if AHB interface receives error
	6	Usage Fault	Programmable	Exceptions due to program errors
System Handlers	11	SVCcall	Programmable	System Service call
	12	Debug Monitor	Programmable	Break points, watch points, external debug
	14	PendSV	Programmable	Pendable Service request for System Device
	15	Systick	Programmable	System Tick Timer
Custom Handlers	16	Interrupt #0	Programmable	External Interrupt #0
	...	...	...	...
	...	...	...	...
	...	...	...	...
	255	Interrupt #239	Programmable	External Interrupt #239

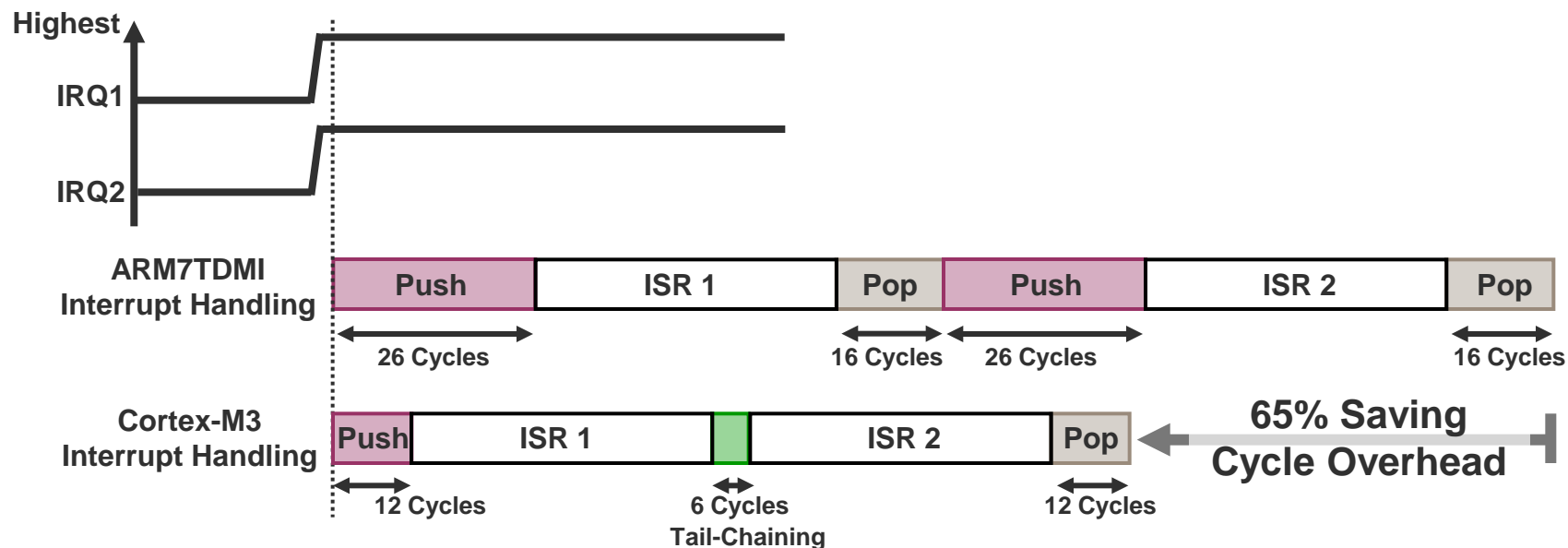
# Exception Model

- **Exceptions cause current machine state to be stacked**
  - Stacked registers conform to Embedded Application Binary Interface (EABI)
- **Exception handlers are trivial as register manipulation carried out in hardware**
  - No assembler code required
  - Simple 'C' interrupt service routines

```
void MY_IRQHandler(void) { /* my handler */ }
```



# Interrupt Response – Tail Chaining



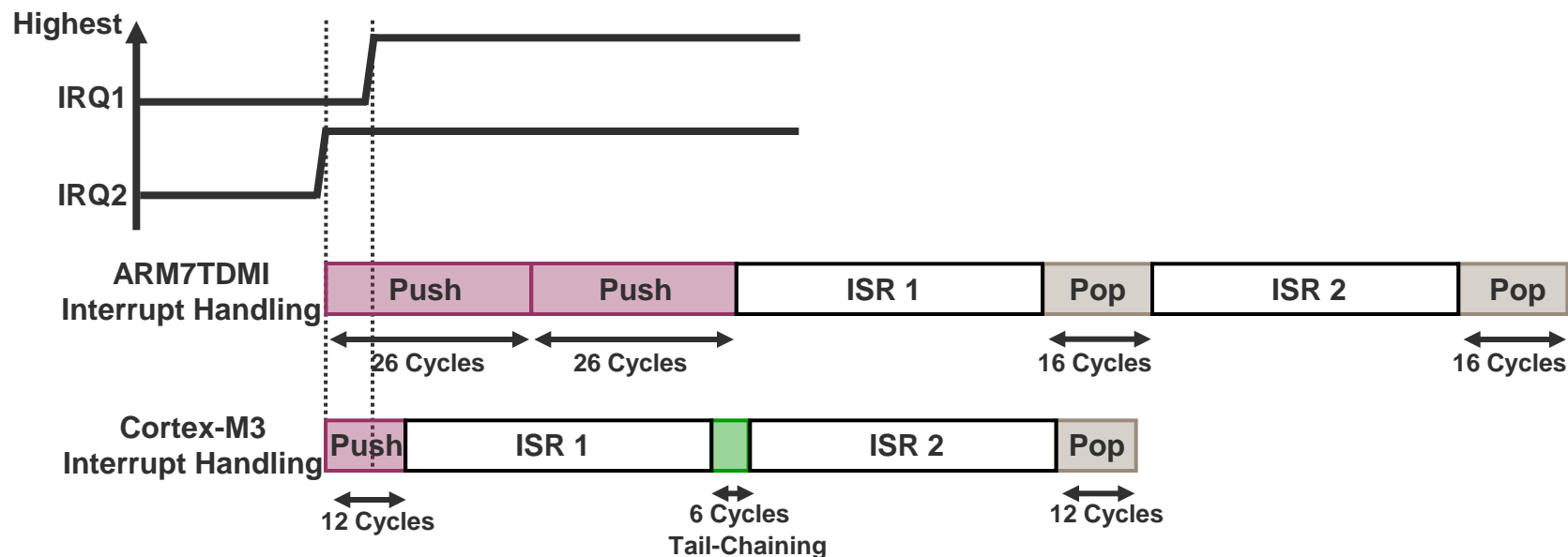
## ARM7TDMI

- 26 cycles from IRQ1 to ISR1  
(up to 42 cycles if in LSM)
- 42 cycles from ISR1 exit to ISR2 entry
- 16 cycles to return from ISR2

## Cortex-M3

- 12 cycles from IRQ1 to ISR1  
(Interruptible/Continual LSM)
- 6 cycles from ISR1 exit to ISR2 entry
- 12 cycles to return from ISR2

# Interrupt Response – Late Arriving



## ARM7TDMI

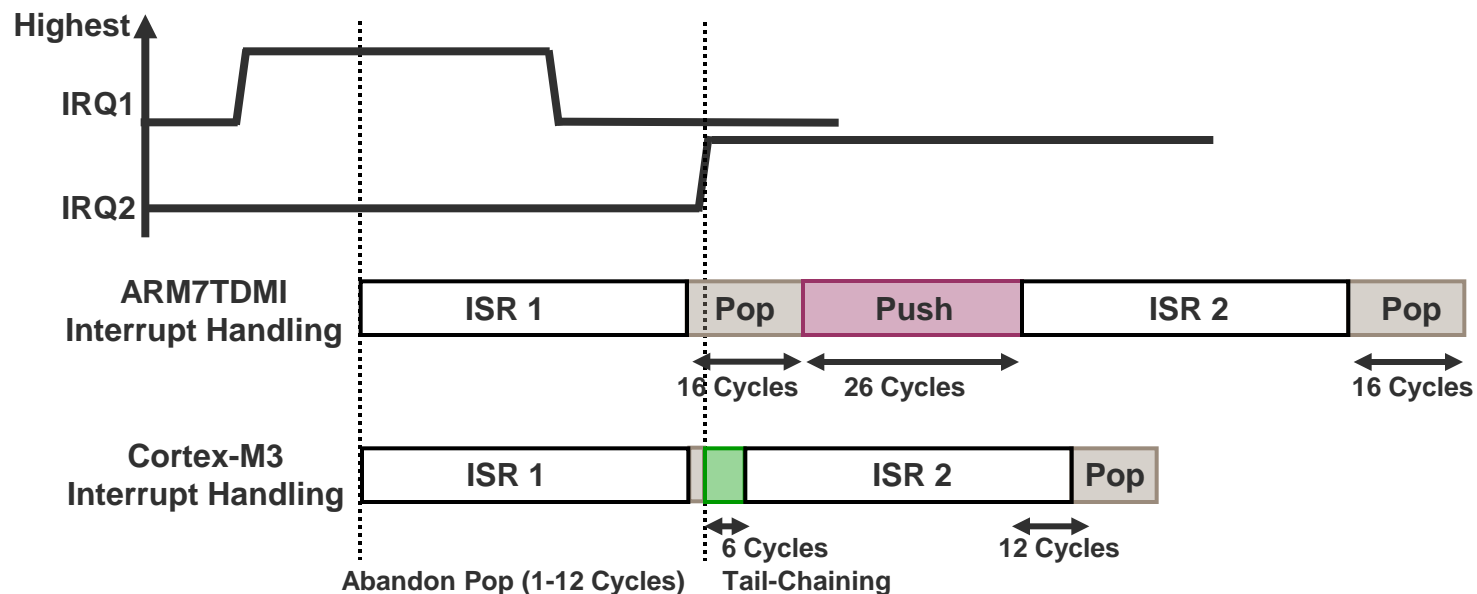
- 26 cycles to ISR2 entered
- Immediately pre-empted by IRQ1  
Additional 26 cycles to enter ISR1.
- ISR 1 completes  
Additional 16 cycles return to ISR2.

## Cortex-M3

- 12 cycles to ISR entry
- Parallel stacking & instruction fetch
- Target ISR may be changed until last cycle (PC is set)
- When IRQ1 occurs new target ISR set



# Interrupt Response – Pop Pre-emption



## ARM7TDMI

- Load multiple not interruptible
- Core must complete the recovery of the stack then re-stack to enter the ISR

## Cortex-M3

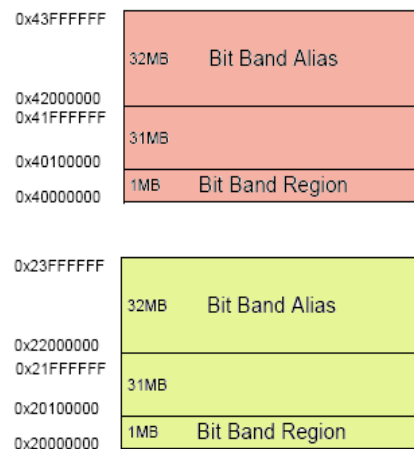
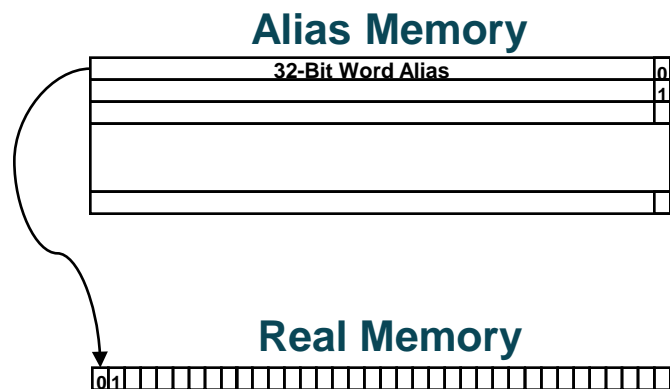
- Hardware un-stacking interruptible
- If interrupted only 6 cycles required to enter ISR2

- Simple 24 Bit down counter
- 4 Registers
  - CTRL - Control and Status
  - LOAD – Reload Value
  - VAL – Current Value
  - CALIB - Calibration Value
- Exception Vector #15
- Accessible from Privileged Mode
  - Can be protected from user application
- Makes porting OS and software easier, because SYSTICK timer will be the same across different Cortex-M products

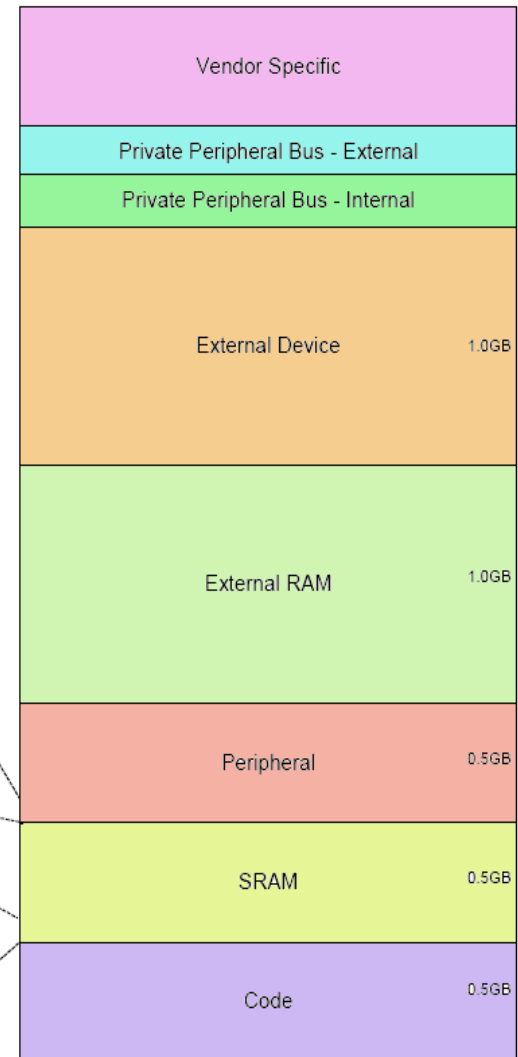


# Atomic Bit Manipulation

- Internal and peripheral data represented as individual bits without the processing overhead normally associated with this type of action.
- For deterministic systems with lots of peripheral data, such as microcontrollers, atomic bit manipulation can compact some types of data by 32 times

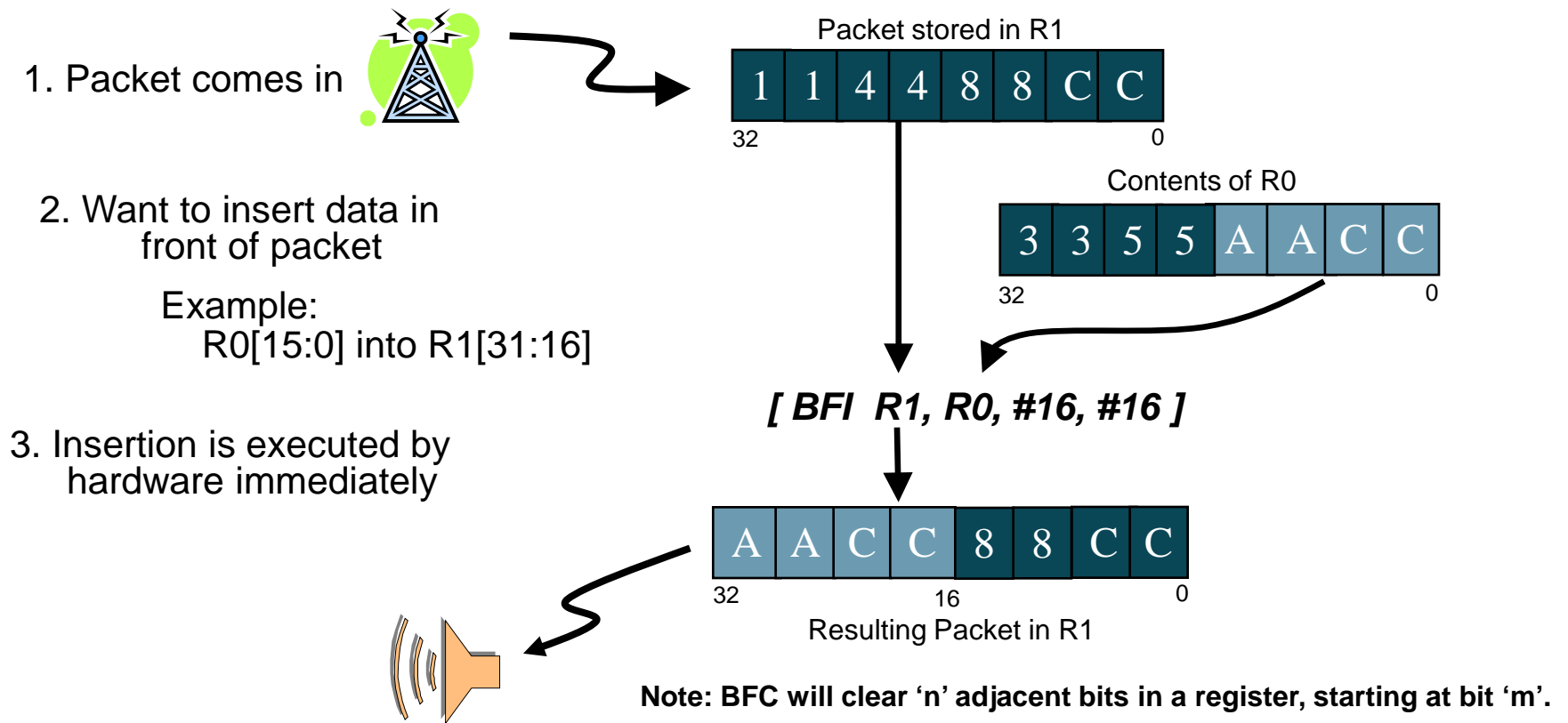


## Cortex-M3 Memory Map



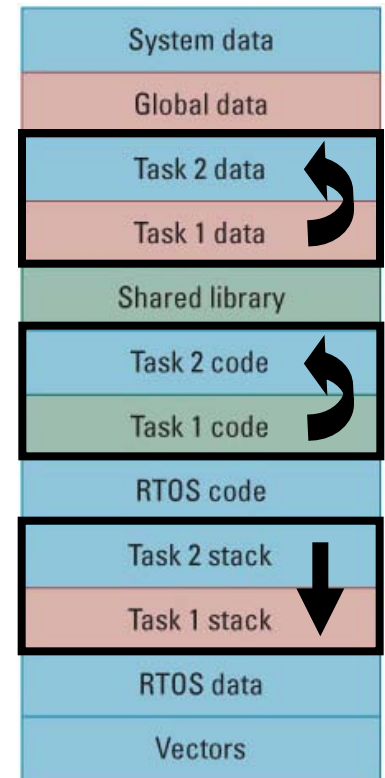
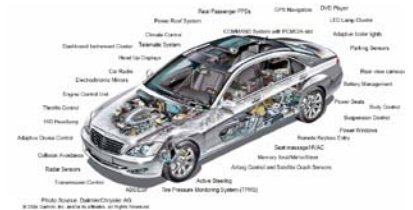
# Bit Manipulation – BFI / BFC

- Insert or clear any number of adjacent bits anywhere in a register
  - Ideal for modifying or stripping packet headers
  - BFI - Bit Field Insert
  - BFC - Bit Filed Clear



# Memory Protection Unit (MPU)

- MPU provides access control for various memory regions
- Zero Latency Memory Protection
  - 8 register-stored regions
  - Same regions used for instructions and data
  - Minimum region size 32 Bytes (max 4GB)
  - No address translation or Page Tables
- Configured via memory-mapped control registers



Not available in Cortex-M0 / Cortex-M1

---

Introduction to the ARM® Cortex™-M Architecture

# CORESIGHT™

# Verification Challenges

## ■ Debug a running system

- Many embedded systems cannot be stopped for debug
  - Brushless DC Motor control
  - Communication protocols lose their handshaking
  - ...
- Analyze dynamic system behaviour
  - Not just a snapshot
- Optimize performance bottlenecks of real-time systems



## ■ System verification

- Many applications require certification
  - Automotive, medical, aero-space, security, and ...
- Code coverage is standard to prove testing





# CoreSight™ Introduction

---

- **ARM CoreSight is a complete on-chip debug and real-time trace solution for the entire system-on-chip (SoC)**
- **Configurable to adapt for market requirements**
  - Debug components can be configured or even removed (check your device manufacturers data sheet)
- **On-the-fly debugging**
  - Debug application while the processor is running
    - Set breakpoints, read/write memory locations
  - Direct debug access to memory (no software overhead)
  - Increased number of breakpoints and watchpoints
- **Flexible trace options**
  - Integrated Data Trace (Cortex-M3)
  - Optional Instruction Trace (ETM)
- **Reduced pin count interface**
  - 2-pin Serial Wire Debug (SWD)
  - 1-pin Serial Wire Viewer (SWV)
  - Standard JTAG mode



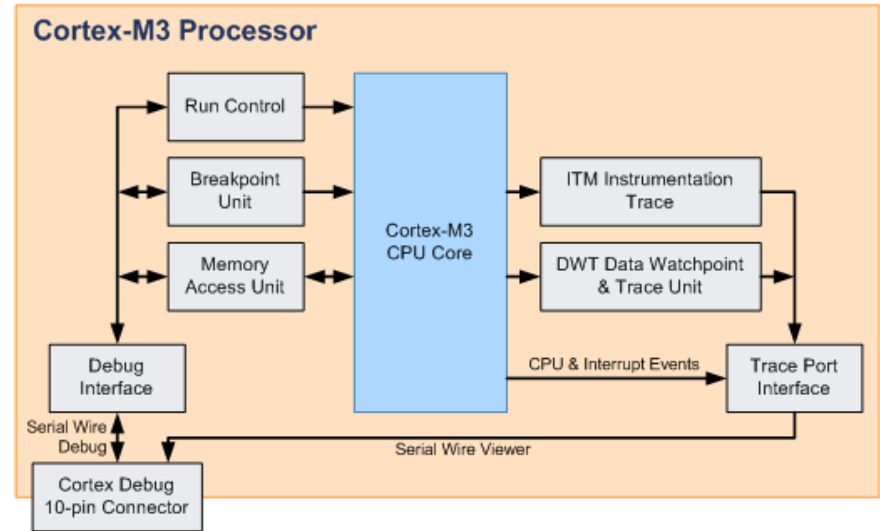
# Cortex-M3 CoreSight Debug

## ■ Real-Time Debug

- While processor is running
- Run, stop and single-step
- Read/Write registers & memory
- Set 8 breakpoints/watchpoints
- Flash download and verify

## ■ Multiple interfaces

- Debug via reduced pin-count interface
  - 2-pin Serial Wire Debug (SWD) interface
- CoreSight debug also available via standard JTAG interface



# Cortex-M3 Data Trace via SWV

## ■ Integrated trace capability

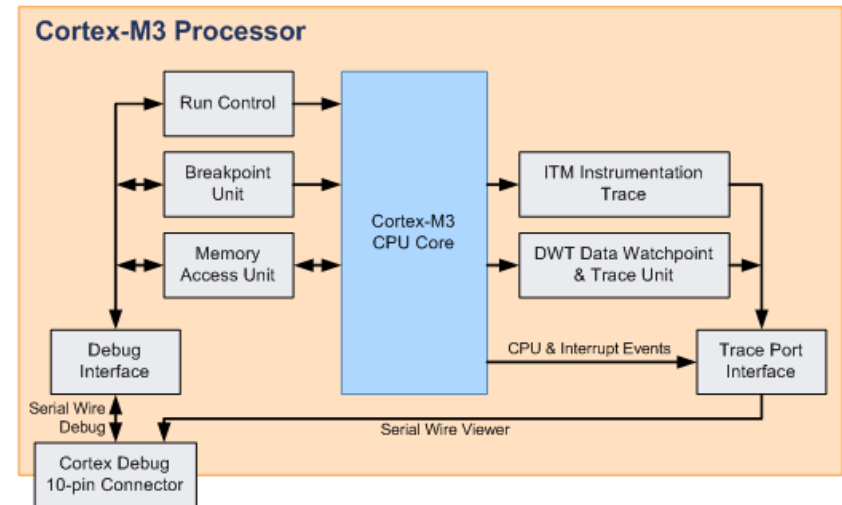
- Instrumentation Trace Macrocell (ITM)
- Data Watchpoint and Trace unit (DWT)

## ■ Data Trace provides

- PC (Program Counter) sampling
- Event counters showing CPU cycle statistics
- Exception and Interrupt execution with timing statistics
- Trace data used for timing analysis or simple *printf*-style debugging

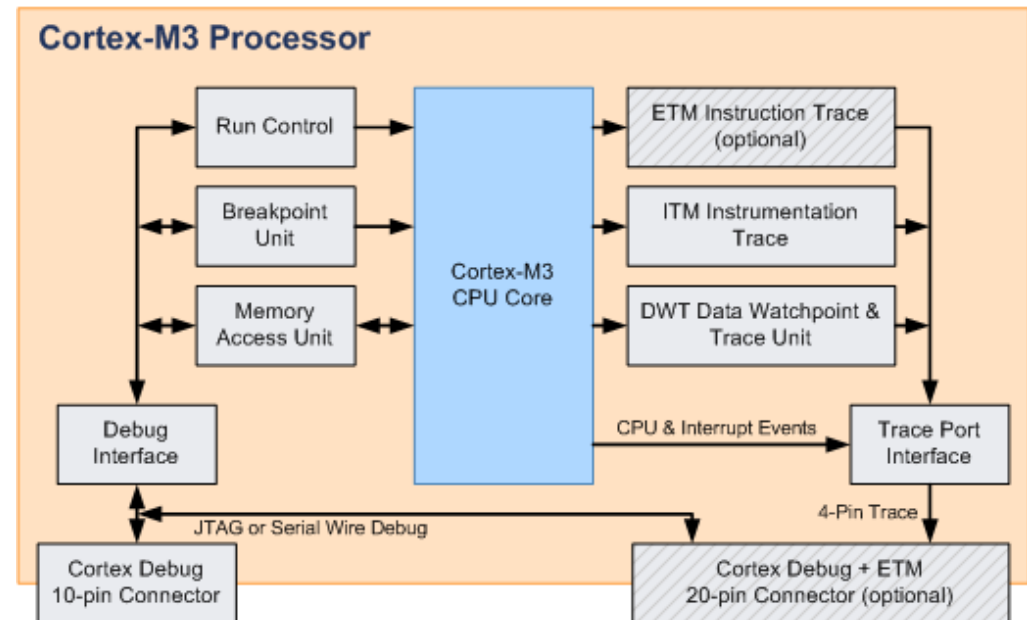
## ■ Serial Wire Viewer (SWV)

- 1-pin output for data trace information in Serial Wire mode
- Minimal overhead
  - Instrument code to output debug messages
  - Configure DWT unit



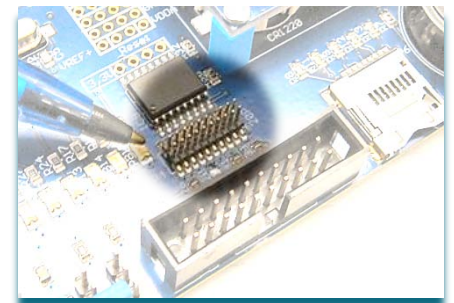
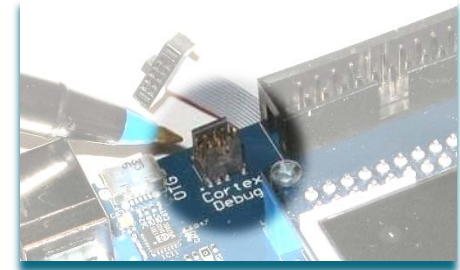
# Cortex-M3 Instruction Trace (ETM)

- **Instruction Trace via Embedded Trace Macrocell (ETM)**
  - Instruction by instruction execution history
- **Enables**
  - Analysis of execution history leading up to an event
  - Code coverage
  - Performance analysis
  - Application optimization
- **ETM Interface**
  - 4 pins for trace information



# CoreSight Connectors

- **20-pin 0.1” Standard JTAG connector**
  - 5 JTAG signals, plus power and GND pins
- **10-pin 0.05” Cortex Debug Connector**
  - Includes traditional 5-pin JTAG interface signals
  - Plus CoreSight interface
    - 2-pin **Serial Wire Debug (SWD)**
    - 1-pin **Serial Wire Viewer (SWV) Output**
      - for Data Trace at minimum cost
- **20-pin 0.05” Cortex Debug + ETM Connector**
  - Same as 10-pin 0.05” **Cortex Debug Connector**
  - Plus 4 ETM trace pins
    - for Data and Instruction Trace



---

Introduction to the ARM® Cortex™-M Architecture

# **CORESIGHT™ IN MDK ARM**

# CoreSight Support in MDK-ARM

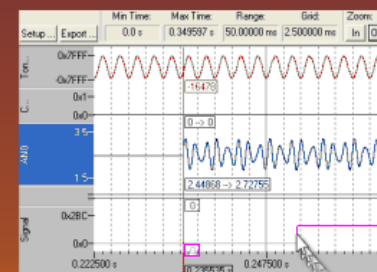
## ■ Debug Windows for

- Trace records
- Event counters
- Exceptions and interrupts
- ITM debug messages
- ETM instruction trace

## ■ Analysis and Optimization Tools

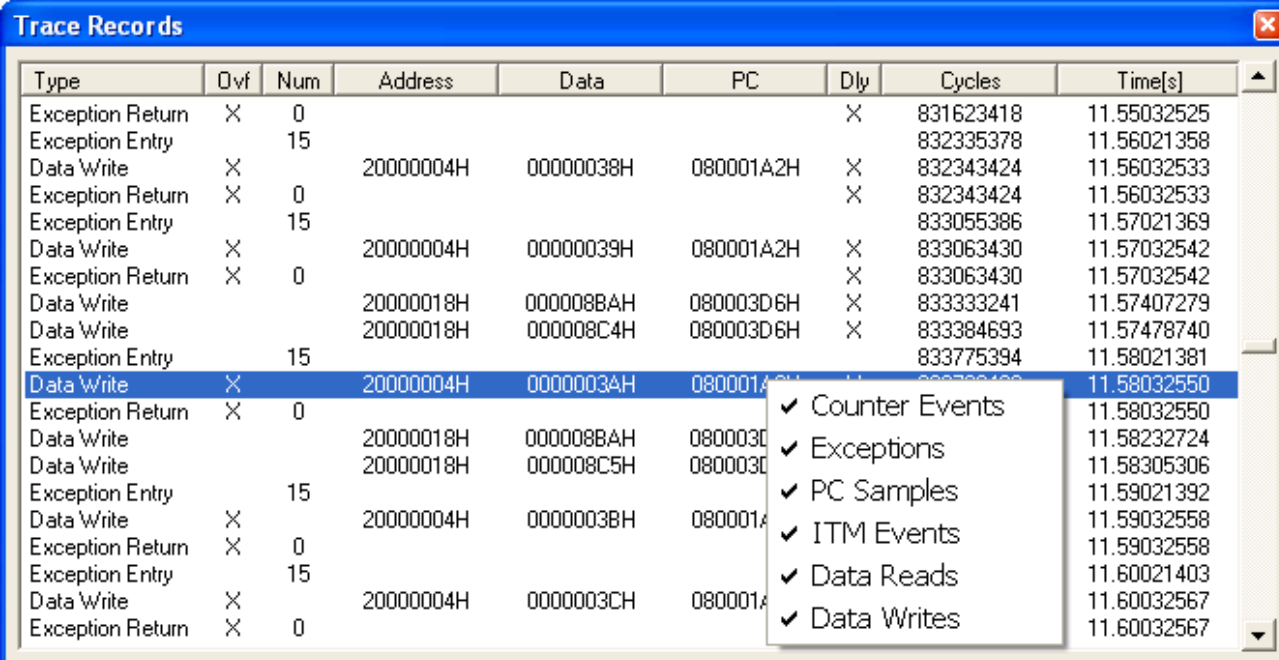
- Logic Analyzer displays state changes and behavior of variables
- Performance analysis of functional blocks
- Execution profiling for finding and resolving bottlenecks
- Code coverage improves testing and verification

Real-Time Trace - Serial Wire Viewer



# Trace Records

- **Trace Records display program flow**
  - Capture timestamp, PC sample, and Read/Write accesses
  - Time delay and lost cycles are noted
- **Raw trace data from all trace sources**
  - Filter window to refine the view
  - Can be updated while CPU is running



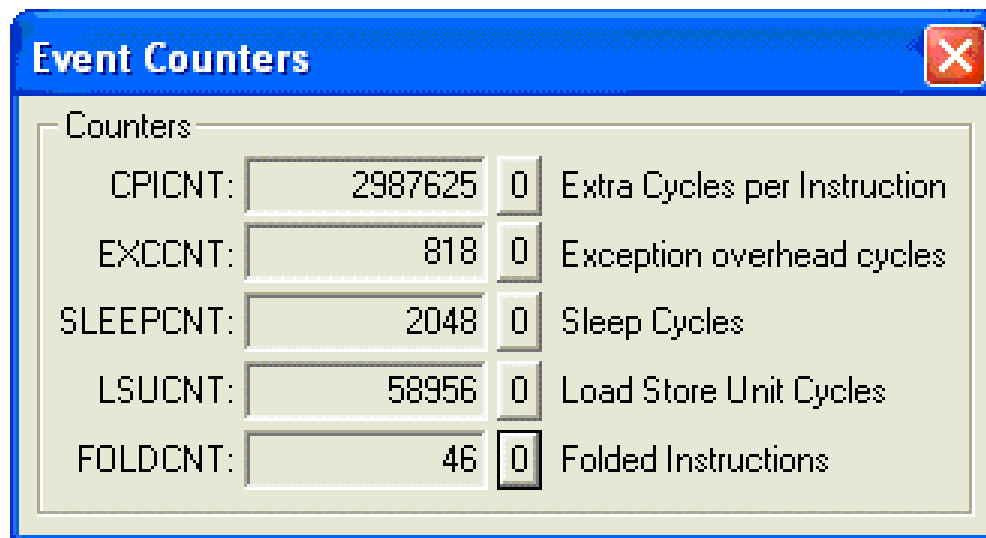
The screenshot shows a window titled "Trace Records" with a table of trace data. The table has columns for Type, Dly, Num, Address, Data, PC, Dly, Cycles, and Time[s]. A context menu is open over the table, listing various trace sources with checkmarks next to them.

Type	Dly	Num	Address	Data	PC	Dly	Cycles	Time[s]
Exception Return	X	0				X	831623418	11.55032525
Exception Entry		15					832335378	11.56021358
Data Write	X		20000004H	00000038H	080001A2H	X	832343424	11.56032533
Exception Return	X	0				X	832343424	11.56032533
Exception Entry		15					833055386	11.57021369
Data Write	X		20000004H	00000039H	080001A2H	X	833063430	11.57032542
Exception Return	X	0				X	833063430	11.57032542
Data Write			20000018H	000008BAH	080003D6H	X	833333241	11.57407279
Data Write			20000018H	000008C4H	080003D6H	X	833384693	11.57478740
Exception Entry		15					833775394	11.58021381
Data Write	X		20000004H	0000003AH	080001A2H	X	833784188	11.58032550
Exception Return	X	0				X	833784188	11.58032550
Data Write			20000018H	000008BAH	080003D6H	X	833792928	11.58232724
Data Write			20000018H	000008C5H	080003D6H	X	833800960	11.58305306
Exception Entry		15					833800960	11.59021392
Data Write	X		20000004H	0000003BH	080001A2H	X	833800960	11.59032558
Exception Return	X	0				X	833800960	11.59032558
Exception Entry		15					833800960	11.60021403
Data Write	X		20000004H	0000003CH	080001A2H	X	833800960	11.60032567
Exception Return	X	0				X	833800960	11.60032567

- ✓ Counter Events
- ✓ Exceptions
- ✓ PC Samples
- ✓ ITM Events
- ✓ Data Reads
- ✓ Data Writes

# Event Counters

- Display real-time values of event counters
- Provide performance indications
  - Extra cycles taken to execute instructions
    - May be due to memory contentions (Flash waitstates)
  - Cycles of overhead caused by handling exceptions
  - Cycles spent in sleep mode
  - Number of cycles spent performing memory accesses
  - Number of folded branch instructions



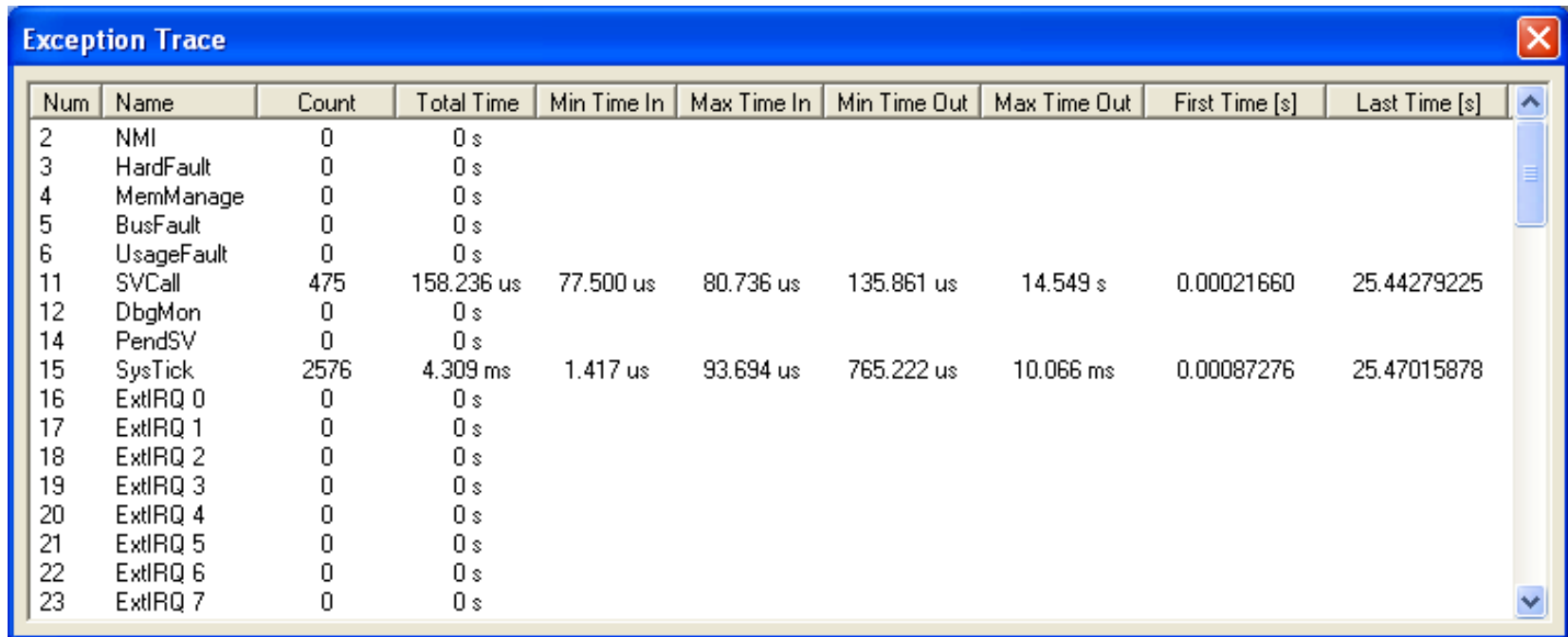
The screenshot shows a window titled "Event Counters" with a close button in the top right corner. The window contains a table of performance metrics. The table has five rows, each representing a different counter. The first column is the counter name, the second is the current value, the third is a reset button (a square with '0'), and the fourth is the description of the counter.

Counter	Value	Reset	Description
CPICNT:	2987625	0	Extra Cycles per Instruction
EXCCNT:	818	0	Exception overhead cycles
SLEEP CNT:	2048	0	Sleep Cycles
LSUCNT:	58956	0	Load Store Unit Cycles
FOLDCNT:	46	0	Folded Instructions



# Exception and Interrupt Trace

- Statistical information about exceptions and interrupts
- Captures detailed information
  - Name and number of exception; number of times entered
  - Max and Min time spent in and out of exceptions

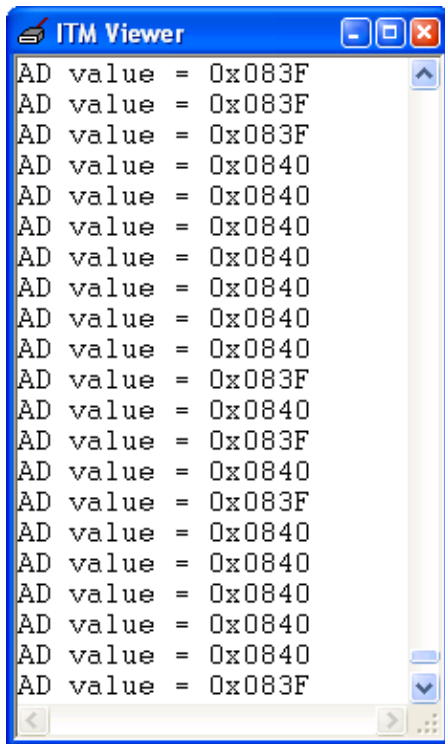


The screenshot shows a window titled "Exception Trace" with a table of exception statistics. The table has 10 columns: Num, Name, Count, Total Time, Min Time In, Max Time In, Min Time Out, Max Time Out, First Time [s], and Last Time [s]. The data is as follows:

Num	Name	Count	Total Time	Min Time In	Max Time In	Min Time Out	Max Time Out	First Time [s]	Last Time [s]
2	NMI	0	0 s						
3	HardFault	0	0 s						
4	MemManage	0	0 s						
5	BusFault	0	0 s						
6	UsageFault	0	0 s						
11	SVCall	475	158.236 us	77.500 us	80.736 us	135.861 us	14.549 s	0.00021660	25.44279225
12	DbgMon	0	0 s						
14	PendSV	0	0 s						
15	SysTick	2576	4.309 ms	1.417 us	93.694 us	765.222 us	10.066 ms	0.00087276	25.47015878
16	ExtIRQ 0	0	0 s						
17	ExtIRQ 1	0	0 s						
18	ExtIRQ 2	0	0 s						
19	ExtIRQ 3	0	0 s						
20	ExtIRQ 4	0	0 s						
21	ExtIRQ 5	0	0 s						
22	ExtIRQ 6	0	0 s						
23	ExtIRQ 7	0	0 s						

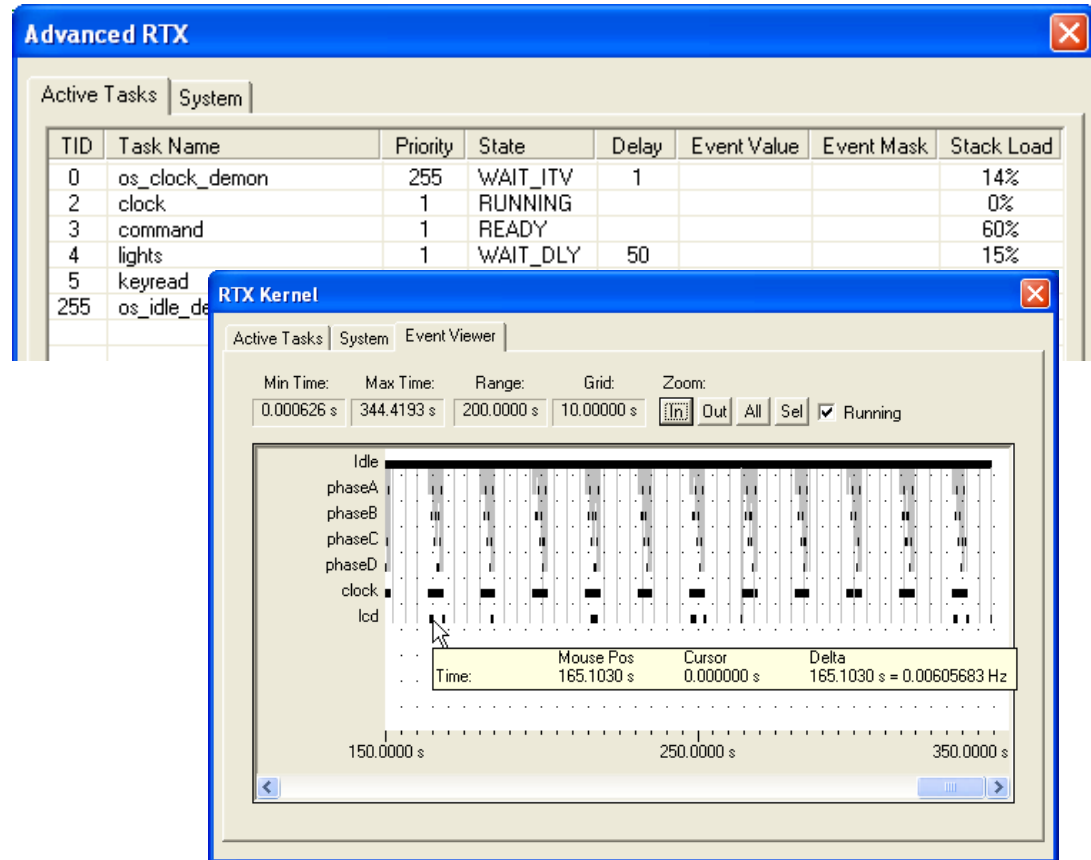
# ITM (Instrumented) Trace

- Displays *printf*-style debug information
  - 32 user-defined channels
- ITM Viewer window
  - RTX event viewer
  - Real-time Update



ITM Viewer

```
AD value = 0x083F
AD value = 0x083F
AD value = 0x083F
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x083F
AD value = 0x0840
AD value = 0x083F
AD value = 0x0840
AD value = 0x083F
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x0840
AD value = 0x083F
```



Advanced RTX

Active Tasks | System

TID	Task Name	Priority	State	Delay	Event Value	Event Mask	Stack Load
0	os_clock_demon	255	WAIT_ITV	1			14%
2	clock	1	RUNNING				0%
3	command	1	READY				60%
4	lights	1	WAIT_DLY	50			15%
5	keyread						
255	os_idle_de						

RTX Kernel

Active Tasks | System | Event Viewer

Min Time: 0.000626 s    Max Time: 344.4193 s    Range: 200.0000 s    Grid: 10.00000 s    Zoom: In Out All Sel  Running

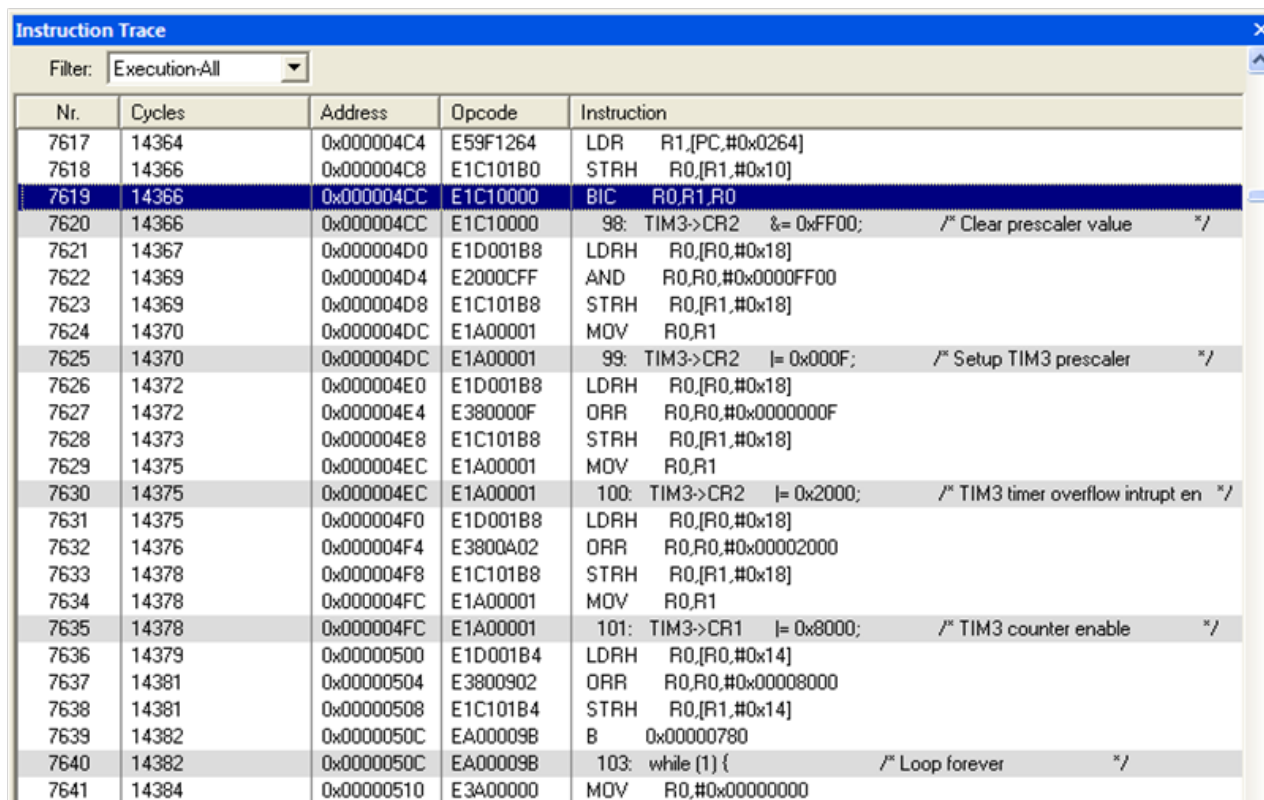
Idle  
phaseA  
phaseB  
phaseC  
phaseD  
clock  
lcd

Time: 165.1030 s    Mouse Pos: 165.1030 s    Cursor: 0.000000 s    Delta: 165.1030 s = 0.00605683 Hz

150.0000 s    250.0000 s    350.0000 s

# ETM Instruction Trace

- **Cycle-by-cycle record of execution history**
  - Captures all executed instructions and data accesses
- **Instruction Trace window displays**
  - Cycle count, Opcode, and Assembly code



The screenshot shows the 'Instruction Trace' window with a filter set to 'Execution-All'. The table below represents the data shown in the window.

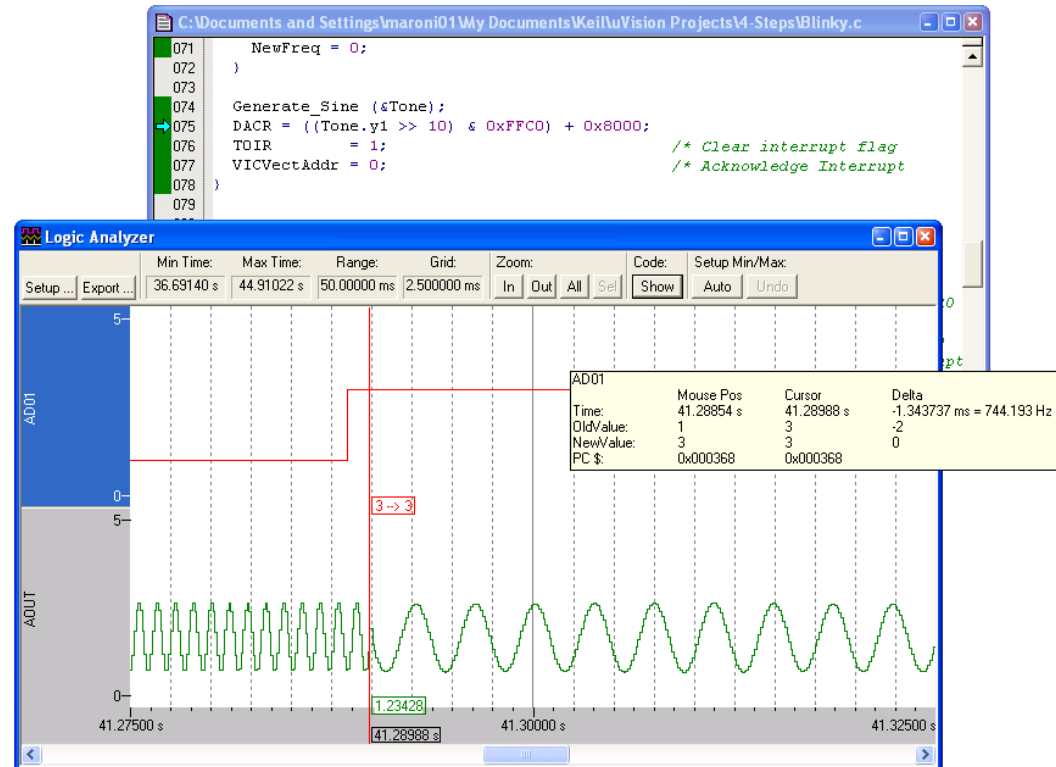
Nr.	Cycles	Address	Opcode	Instruction
7617	14364	0x000004C4	E59F1264	LDR R1,[PC,#0x0264]
7618	14366	0x000004C8	E1C101B0	STRH R0,[R1,#0x10]
7619	14366	0x000004CC	E1C10000	BIC R0,R1,R0
7620	14366	0x000004CC	E1C10000	98: TIM3->CR2 &= 0xFF00; /* Clear prescaler value */
7621	14367	0x000004D0	E1D001B8	LDRH R0,[R0,#0x18]
7622	14369	0x000004D4	E2000CFF	AND R0,R0,#0x0000FF00
7623	14369	0x000004D8	E1C101B8	STRH R0,[R1,#0x18]
7624	14370	0x000004DC	E1A00001	MOV R0,R1
7625	14370	0x000004DC	E1A00001	99: TIM3->CR2  = 0x000F; /* Setup TIM3 prescaler */
7626	14372	0x000004E0	E1D001B8	LDRH R0,[R0,#0x18]
7627	14372	0x000004E4	E380000F	ORR R0,R0,#0x0000000F
7628	14373	0x000004E8	E1C101B8	STRH R0,[R1,#0x18]
7629	14375	0x000004EC	E1A00001	MOV R0,R1
7630	14375	0x000004EC	E1A00001	100: TIM3->CR2  = 0x2000; /* TIM3 timer overflow intrupt en */
7631	14375	0x000004F0	E1D001B8	LDRH R0,[R0,#0x18]
7632	14376	0x000004F4	E3800A02	ORR R0,R0,#0x00002000
7633	14378	0x000004F8	E1C101B8	STRH R0,[R1,#0x18]
7634	14378	0x000004FC	E1A00001	MOV R0,R1
7635	14378	0x000004FC	E1A00001	101: TIM3->CR1  = 0x8000; /* TIM3 counter enable */
7636	14379	0x00000500	E1D001B4	LDRH R0,[R0,#0x14]
7637	14381	0x00000504	E3800902	ORR R0,R0,#0x00008000
7638	14381	0x00000508	E1C101B4	STRH R0,[R1,#0x14]
7639	14382	0x0000050C	EA00009B	B 0x00000780
7640	14382	0x0000050C	EA00009B	103: while (1) { /* Loop forever */
7641	14384	0x00000510	E3A00000	MOV R0,#0x00000000

Supported by:

ULINKPro, J-Trace, JTAGjet-Trace

# Logic Analyzer

- **Allows signals to be monitored graphically**
  - Intuitive view of signal behaviour
- **Accurate timing**
  - Easy, fast analysis of signal timing with access to source code
  - View delta changes from cursor to current location
- **Code analysis**
  - View instruction that caused variable change

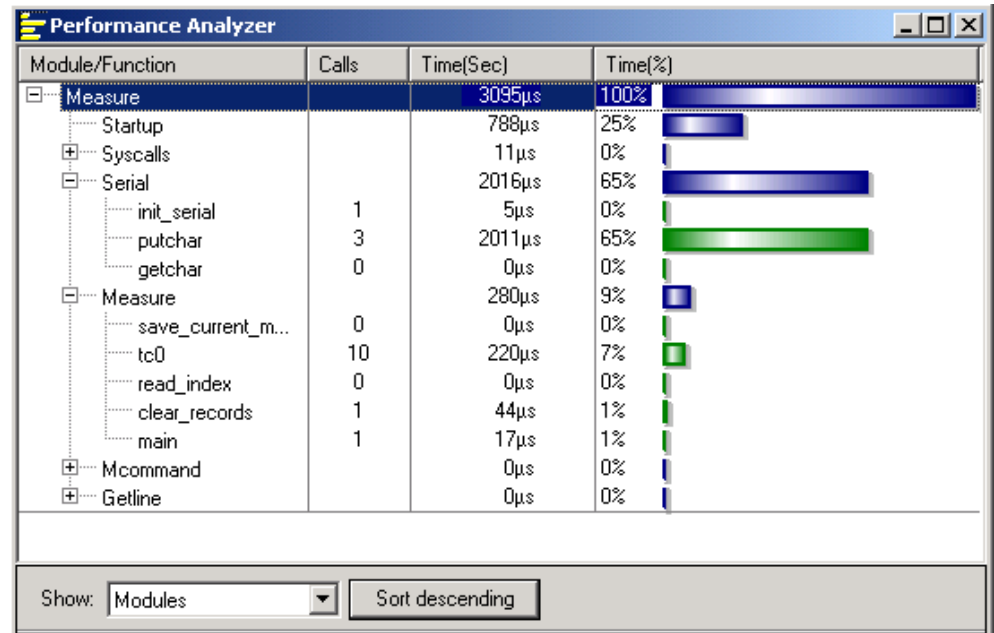


Supported by:

ULINKPro, J-Trace, JTAGjet-Trace

# Performance Analyzer

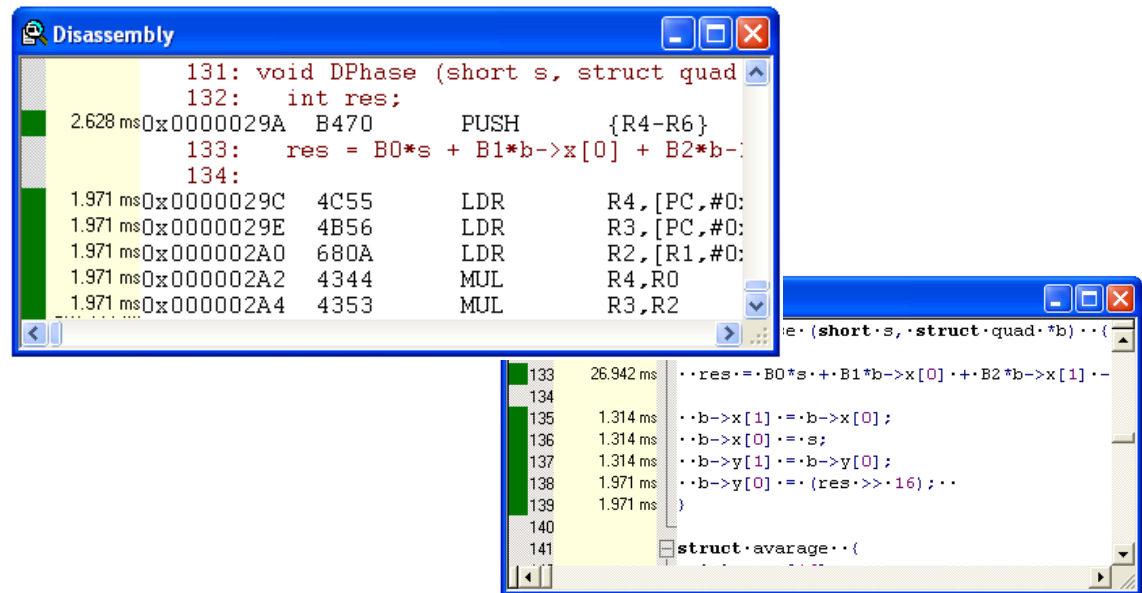
- Analysis of functions and program blocks
- Records and displays execution times for functions
  - Number of calls for each function and program block
  - Time spent executing that block
  - Bar graph display
- Optimize applications
  - See where the program is spending the most time



Supported by:  
ULINKPro

# Execution Profiling

- **Instruction by instruction execution analysis**
- **Generates detailed profiling records**
  - How many times each line of code was called
  - Execution time for each instruction
- **Analyze your code's performance**
  - Which instructions are taking longer than expected to execute



The screenshot shows a disassembly window with the following data:

Address	Hex	Instruction	Time
131	void DPhase (short s, struct quad		
132	int res;		
2.628 ms	0x0000029A B470	PUSH {R4-R6}	
133	res = B0*s + B1*b->x[0] + B2*b-		
134			
1.971 ms	0x0000029C 4C55	LDR R4,[PC,#0:	
1.971 ms	0x0000029E 4B56	LDR R3,[PC,#0:	
1.971 ms	0x000002A0 680A	LDR R2,[R1,#0:	
1.971 ms	0x000002A2 4344	MUL R4,R0	
1.971 ms	0x000002A4 4353	MUL R3,R2	

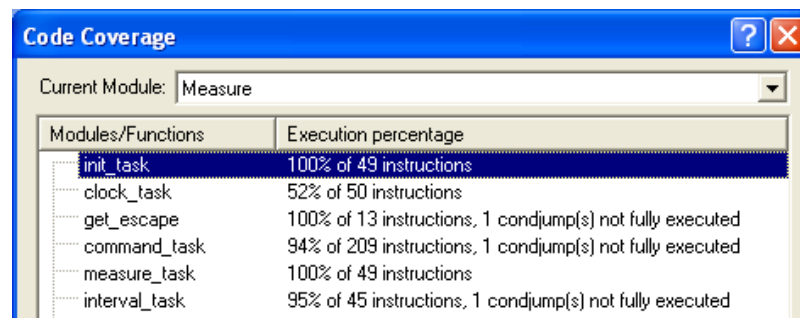
The second window shows the following assembly code with execution times:

133	26.942 ms	..res += B0*s + B1*b->x[0] + B2*b->x[1] --
134		
135	1.314 ms	..b->x[1] += b->x[0];
136	1.314 ms	..b->x[0] += s;
137	1.314 ms	..b->y[1] += b->y[0];
138	1.971 ms	..b->y[0] += (res >> 16); ..
139	1.971 ms	}
140		
141		struct avarage .. {

Supported by:  
ULINKPro

# Code Coverage

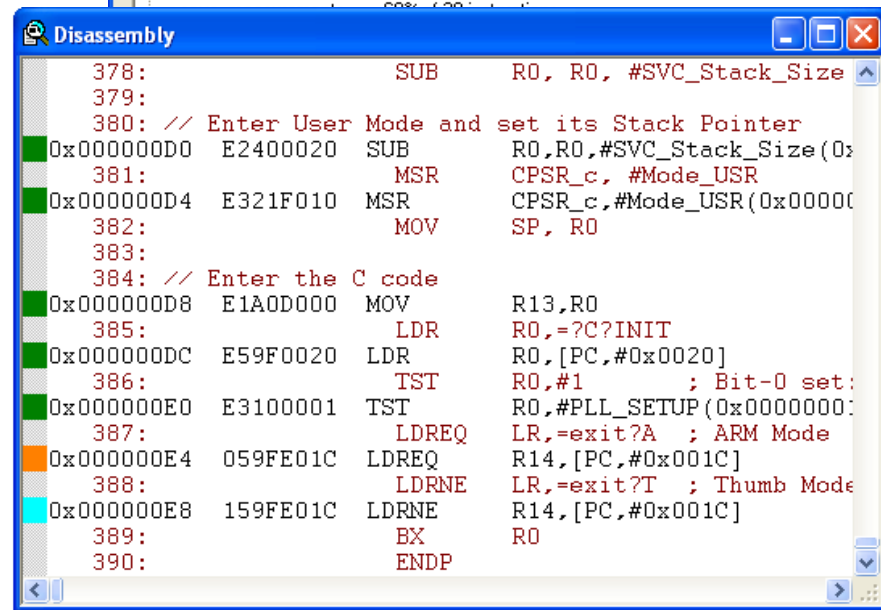
- **Helps tune testing to ensure all lines of code are executed**
  - Often used for certification
- **Color-coded instruction status**
  - Whether a line has executed
  - Taken and skipped branches
- **Ensures complete testing**
  - Save current coverage
  - Update testing and re-run code coverage



Code Coverage

Current Module: Measure

Modules/Functions	Execution percentage
init_task	100% of 49 instructions
clock_task	52% of 50 instructions
get_escape	100% of 13 instructions, 1 condjump(s) not fully executed
command_task	94% of 209 instructions, 1 condjump(s) not fully executed
measure_task	100% of 49 instructions
interval_task	95% of 45 instructions, 1 condjump(s) not fully executed



Disassembly

```
378:          SUB     R0, R0, #SVC_Stack_Size
379:
380: // Enter User Mode and set its Stack Pointer
0x000000D0 E2400020 SUB     R0,R0,#SVC_Stack_Size(0x
381:          MSR     CPSR_c, #Mode_USR
0x000000D4 E321F010 MSR     CPSR_c,#Mode_USR(0x00000
382:          MOV     SP, R0
383:
384: // Enter the C code
0x000000D8 E1A0D000 MOV     R13,R0
385:          LDR     R0,=?C?INIT
0x000000DC E59F0020 LDR     R0,[PC,#0x0020]
386:          TST     R0,#1 ; Bit-0 set:
0x000000E0 E3100001 TST     R0,#PLL_SETUP(0x0000000
387:          LDREQ  LR,=exit?A ; ARM Mode
0x000000E4 059FE01C LDREQ  R14,[PC,#0x001C]
388:          LDRNE  LR,=exit?T ; Thumb Mode
0x000000E8 159FE01C LDRNE  R14,[PC,#0x001C]
389:          BX     R0
390:          ENDP
```

Supported by:  
ULINKPro

---

Introduction to the ARM® Cortex™-M Architecture

# CMSIS

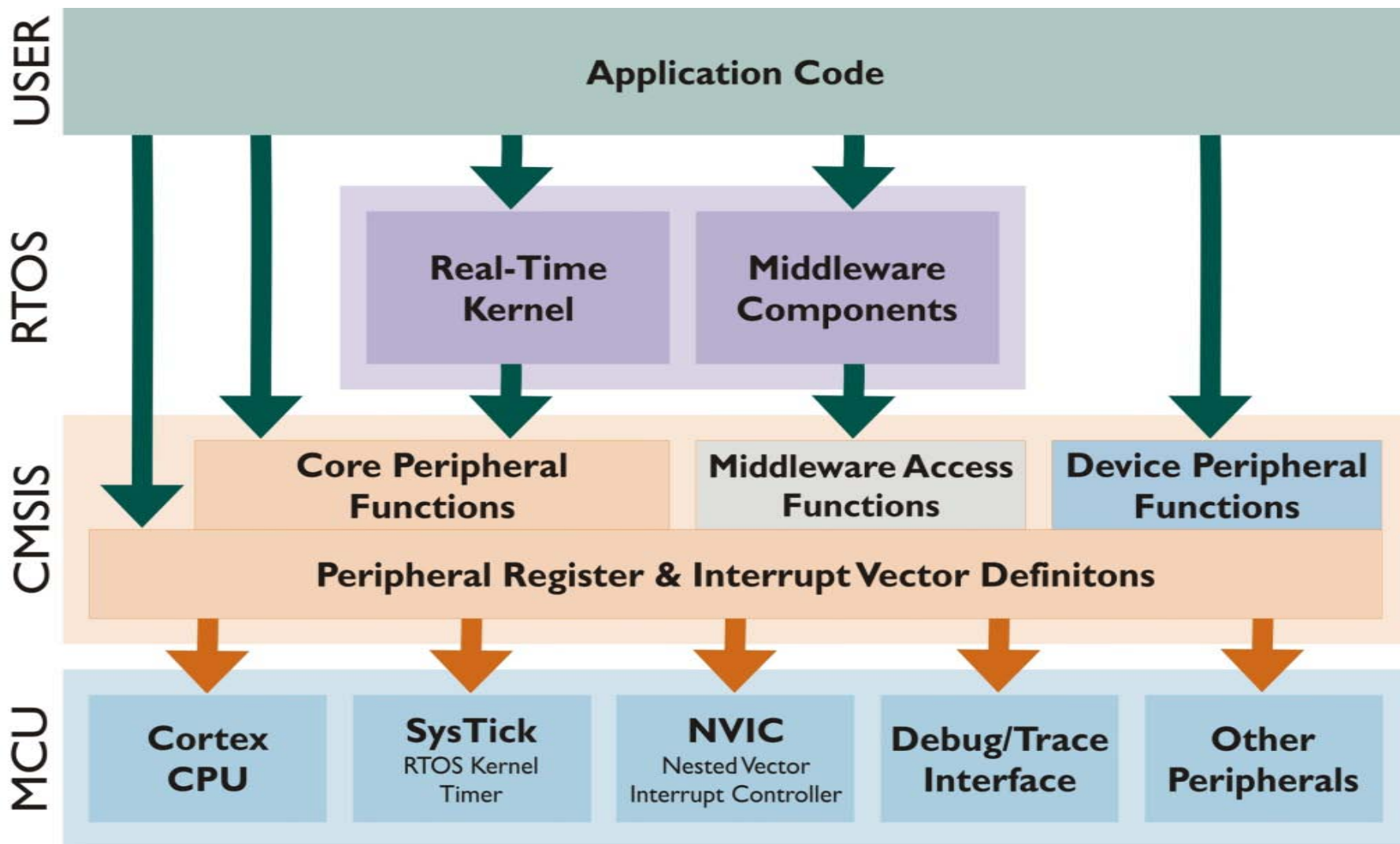


# What is CMSIS

- **CMSIS - Cortex Microcontroller Software Interface Standard**
  - Abstraction layer for all Cortex-M processor based devices
  - Developed in conjunction with silicon, tools and middleware Partners
- **CMSIS Peripheral Access Layer defines**
  - Consistent layout of all peripheral registers
  - Vector definitions for all exceptions and interrupts
  - Functions to access core registers and core peripherals
  - Device independent interface for RTOS Kernels
  - Debug channel (for printf-style + RTOS Kernel)
- **CMSIS Middleware Access Layer provides**
  - Common methods of accessing communication peripherals
- **CMSIS compliant software components allow**
  - Easy reuse of example applications or template code
  - Combination of software components from multiple vendors



# CMSIS - Structure



# Goal of CMSIS

---

- **Reduce Complexity**
  - Only a small number of files and functions are provided
  - Register Interface to access simple peripheral functions
  - Ensure that efficient code can be generated by C compilers
- **Provide Industry-Wide Programming Standards**
  - Ensure a consistent method for accessing peripherals
  - Adapt Object-Oriented programming techniques
  - Conform to safety requirements (MISRA)
- **Support Partnerships and Innovation**
  - Support a wide range of silicon, tool, and middleware Partners
  - Allow differentiation and innovation in peripherals
  - Simple Licensing Terms (freely available to everyone)
  - Potential for new software business models



# CMSIS – Files for Peripheral Access Layer

## Compiler Vendor-Independent Files:

- **Cortex-Mx Core Files** ([provided by ARM](#))
  - core\_cm3.h+core\_cm3.c      core\_cm0.h+core\_cm0.c
- **Device-specific Files** ([provided by Silicon Vendors](#))
  - Register Header File (*device.h*)
  - System Startup File (*system\_device.c*)
- **Compatible with all supported Compilers (IAR, RealView, GNU, ...)**

## Compiler-Vendor + Device-Specific Startup File:

- **Device Specific Compiler Startup Code** ([provided by Silicon Vendors](#))
  - *startup\_device.s*

## CMSIS Files are available via [www.onARM.com](http://www.onARM.com):

- **Device Database that lists all available devices**
  - CMSIS Files can be downloaded

# CMSIS – Example

```
#include <device.h> // file name depends on device

void SysTick_Handler (void) { // SysTick Interrupt Handler
    ;
}

void TIM1_UP_IRQHandler (void) { // Timer Interrupt Handler
    ;
}

void timer1_init(int frequency) { // set up Timer (device specific)
    NVIC_SetPriority (TIM1_UP_STM_IRQn, 1); // Set Timer priority
    NVIC_EnableIRQ (TIM1_UP_STM_IRQn); // Enable Timer Interrupt
}

void main (void) {
    SystemInit (); // global system setup

    if (SysTick_Config (SystemFrequency / 1000)) { // SysTick 1mSec
        : // Handle Error
    }
    timer1_init (); // setup device specific timer
}
```

# Standards Make Sense & CMSIS

## Standards Make Sense

Today's electronics industry is based upon standards. These standards have enabled the development of ever more advanced technology which has been adopted and applied by system designers around the globe.

The ARM Cortex™-M processors are rapidly setting the standard for advanced microcontroller applications, a fact illustrated by a growing number of leading MCU vendors each providing a differentiated product, but based upon the same standard ARM architecture.

## Cortex-M Microcontrollers

For more information visit  
[www.onARM.com](http://www.onARM.com)

**Cortex**  
Intelligent Processors by ARM

**ARM** The Architecture for the Digital World®

© ARM Ltd. ADI173 | 03.09

## ARM® Cortex™ Microcontroller Software Interface Standard

The ARM® Cortex™ Microcontroller Software Interface Standard (CMSIS) is a vendor-independent hardware abstraction layer for the Cortex-M processor series. The CMSIS enables consistent and simple software interfaces to the processor for interface peripherals, real-time operating systems, and middleware, simplifying software re-use, reducing the learning curve for new microcontroller developers and reducing the time to market for new devices.

Creation of software is acknowledged as a major cost factor by the embedded industry. By standardizing the software interfaces across all Cortex silicon vendor products, this cost is significantly reduced, especially when creating projects for new devices or migrating existing software to a Cortex processor-based microcontroller from other silicon vendors.

The creation of the CMSIS enables silicon vendors to focus their resources on the differentiating peripheral features of their product, and eliminates the need to maintain their own individual and incompatible standards for programming a microcontroller.

The standard is fully scalable to ensure that it is suitable for all Cortex-M processor series microcontrollers from the smallest 8KB device up to devices with sophisticated communication peripherals such as Ethernet or USB-OTG. (The CMSIS memory requirement for the Core Peripheral Access Layer is less than 1KB code, less than 10 bytes RAM).

The CMSIS answers the challenges that are faced when software components are deployed to physical microcontroller devices based on a Cortex-M0, Cortex-M1 or Cortex-M3 processor. The CMSIS will be also expanded to future Cortex-M processor cores.





# The complete offering for ARM MCU applications

**IAR Embedded Workbench**  
The complete set of development tools for building and debugging embedded applications with 1400 example projects

Support for ARM7™, ARM9™, ARM9E™, ARM10™, ARM11, Cortex-M0, Cortex-M1, Cortex-M3



## IAR visualSTATE

State machine development tool that provides advanced verification and validation utilities and generates very compact C/C++ code that is 100% consistent with your system design.



## IAR Development kits

30+ development kits with all you need to get started in 30 minutes! Includes development tools, PCB, J-Link probe and example projects

**IAR PowerPac**  
The tightly integrated RTOS and middleware alternative for your ARM application with support for USB and TCP/IP connectivity



## Debug probes

Fastest debug probes on the market

- IAR J-Link
- IAR J-Trace
- IAR J-Trace CM3



## Microcontroller Development Kit

Complete software development environment for Cortex-M1/M3 and ARM7/9 microcontrollers

Easy to learn and use, yet powerful enough for the most demanding embedded ARM application

### RealView® Microcontroller Development Kit

RealView C/C++ Compiler

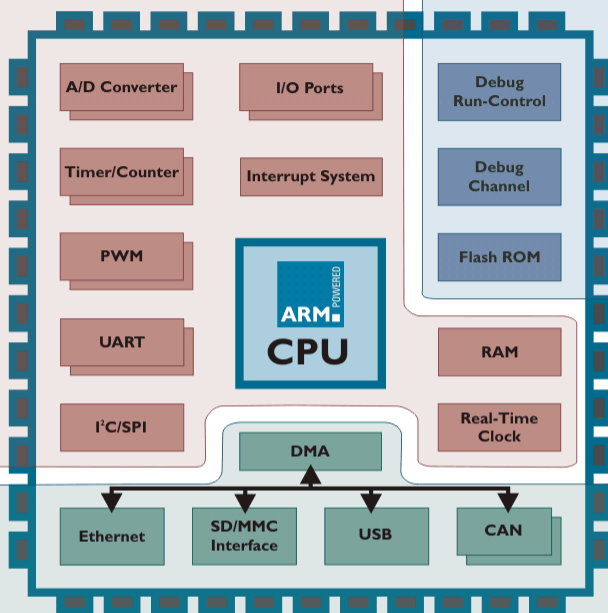
RTX RTOS Kernel Library

µVision Device Database & IDE

µVision Debugger & Analysis Tools

Complete Development Kit

Examples and Templates



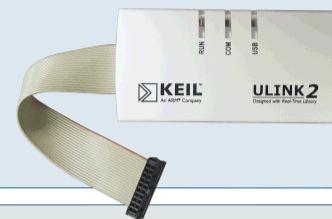
## RTX and Real-Time Library

Fully featured real-time kernel

Library of middleware components to speed up software development and solve real-time and communication challenges

## ULINK2 USB Adapter

On-the-fly debugging and Flash programming via JTAG or serial interface



### RealView® Real-Time Library

RTX RTOS Source Code

TCPnet Networking Suite

Flash File System

USB Device Interface

CAN

Examples and Templates





---

Introduction to the ARM® Cortex™-M Architecture

# CORTEX-M MCU OVERVIEW

# Cortex-M - Success in Market



Over 20 licensees of Cortex-M3 processor  
Over 150 Cortex-M3 processor-based devices  
Over 400 ARM processor-based MCUs  
140% CAGR units shipped by vendors



---

Introduction to the ARM® Cortex™-M Architecture

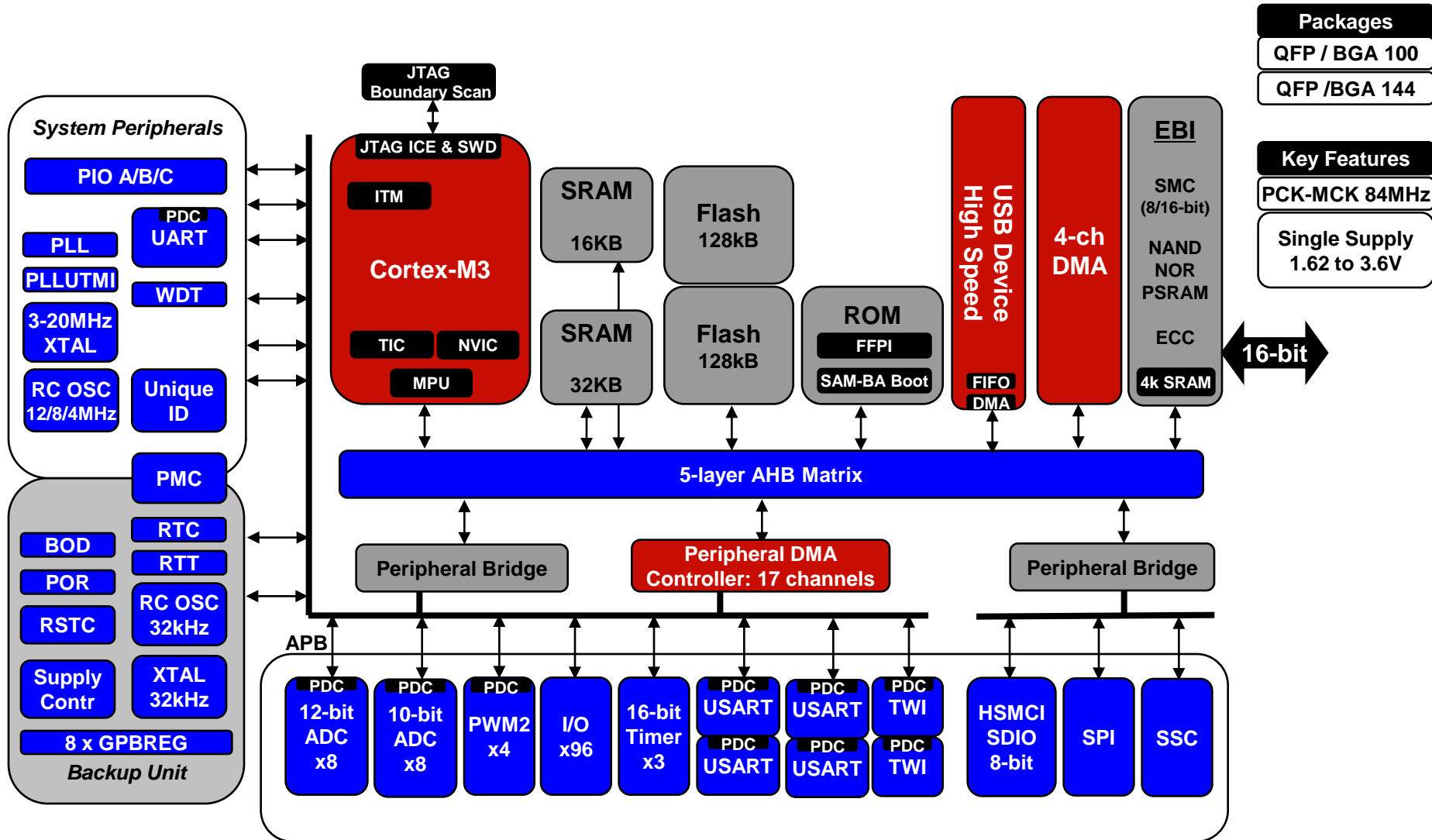
**ATMEL**



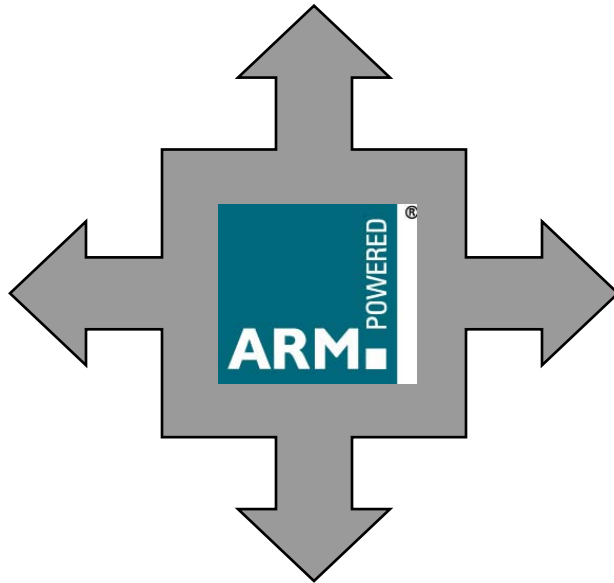
- **World's First Cortex M3 Flash MCU with High Speed USB device**
  - Optimized for applications requiring USB High Speed
- **First CM3 MCU with Dual bank Flash with boot bank select**
- **Lowest CM3 MCU operating supply voltage at 1.62V**

- **Many applications require more than 12Mbps:**
  - USB Dongle: HDTV, Wireless, Memory Stick
  - Dataloggers: Fast data download
  - PC Application: Webcam, Printer, External HDD
  - Replacement of inter-PCB links like high-speed SPI and parallel wires by USB
  
- **High Speed USB 480 Mbps is the solution**





- **High Speed peripherals**
  - USB HS Device
  - SPI
  - SD/MMC
  - SDIO
  - Memory mapped FPGA or ASSP



High Speed Peripheral	Maximum bandwidth
USB Device	425 Mbps
External Bus Interface	>500 Mbps out of 7ns SRAM
MMC interface	384 Mbps
SDIO/SDCard	192 Mbps
SPI	48 Mbps



# Key Features and Benefits

SAM3U Key Feature	Customer Benefit
First CM3 MCU with USB High speed Device and integrated transceiver	Fast down and up loading of datafiles. USB is a robust, EMI tolerant, plug and play and low cost high speed serial interconnect
96 MHz max operating frequency	Processing power for today and tomorrow
DMA and distributed SRAM, High Speed SDIO, SD/MMC Card, SPI and EBI	Ability to sustain high data transfer rates between multiple high-speed peripherals
First CM3 MCU with Dual bank Flash with boot bank select	Save In Application Programming (IAP) including for the boot program
Lowest CM3 MCU operating supply voltage at 1.62V	True 1.8V operation (except ADC and USB transceiver) Extended operation when running from two AA alkaline batteries
Memory Protection Unit (MPU)	Improved code protection and secures multi-application/ task execution
Differential input and Programmable Gain Amplifier (PGA) on 12-bit 1Msps ADC	Reduced BOM cost and board space



---

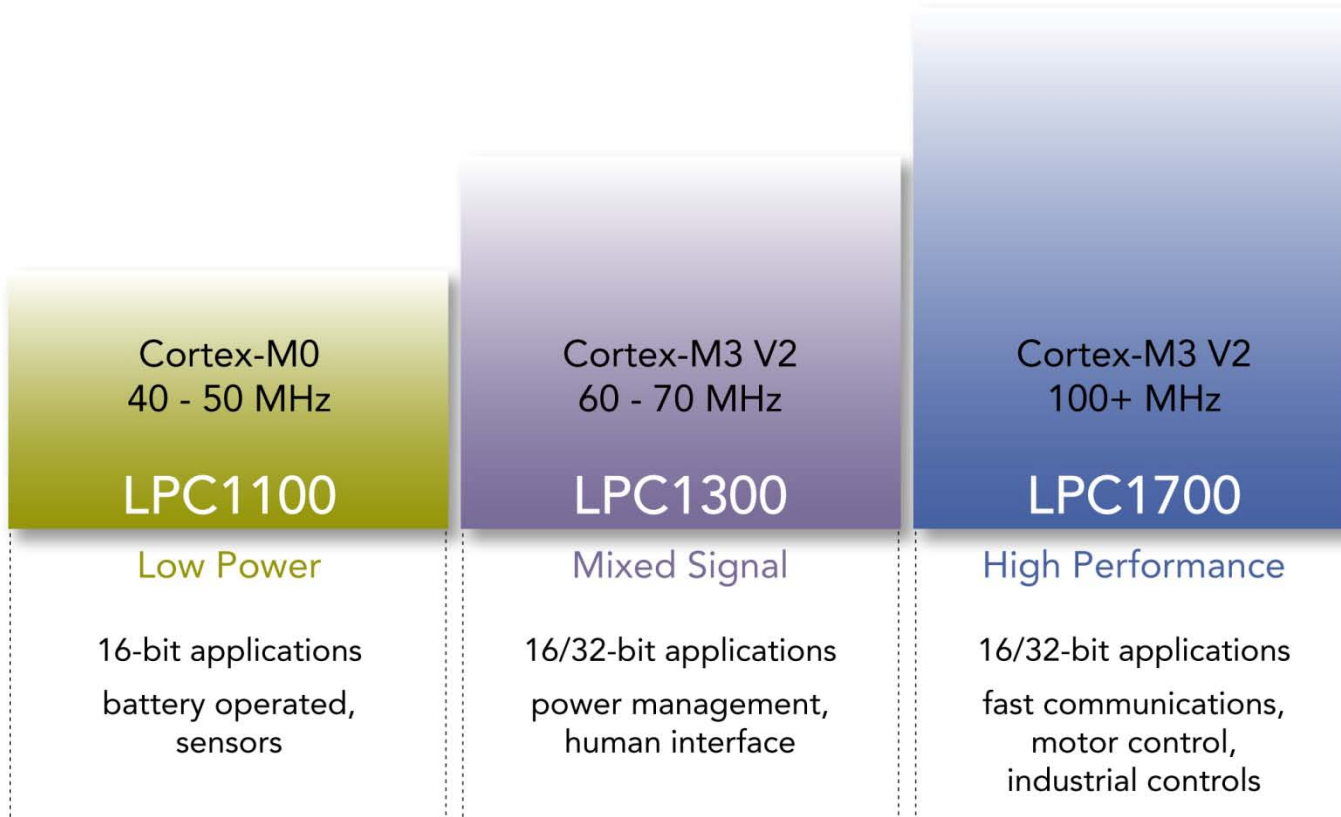
Introduction to the ARM® Cortex™-M Architecture

**NXP**

# NXP ARM Cortex™ Microcontroller Product Series Overview



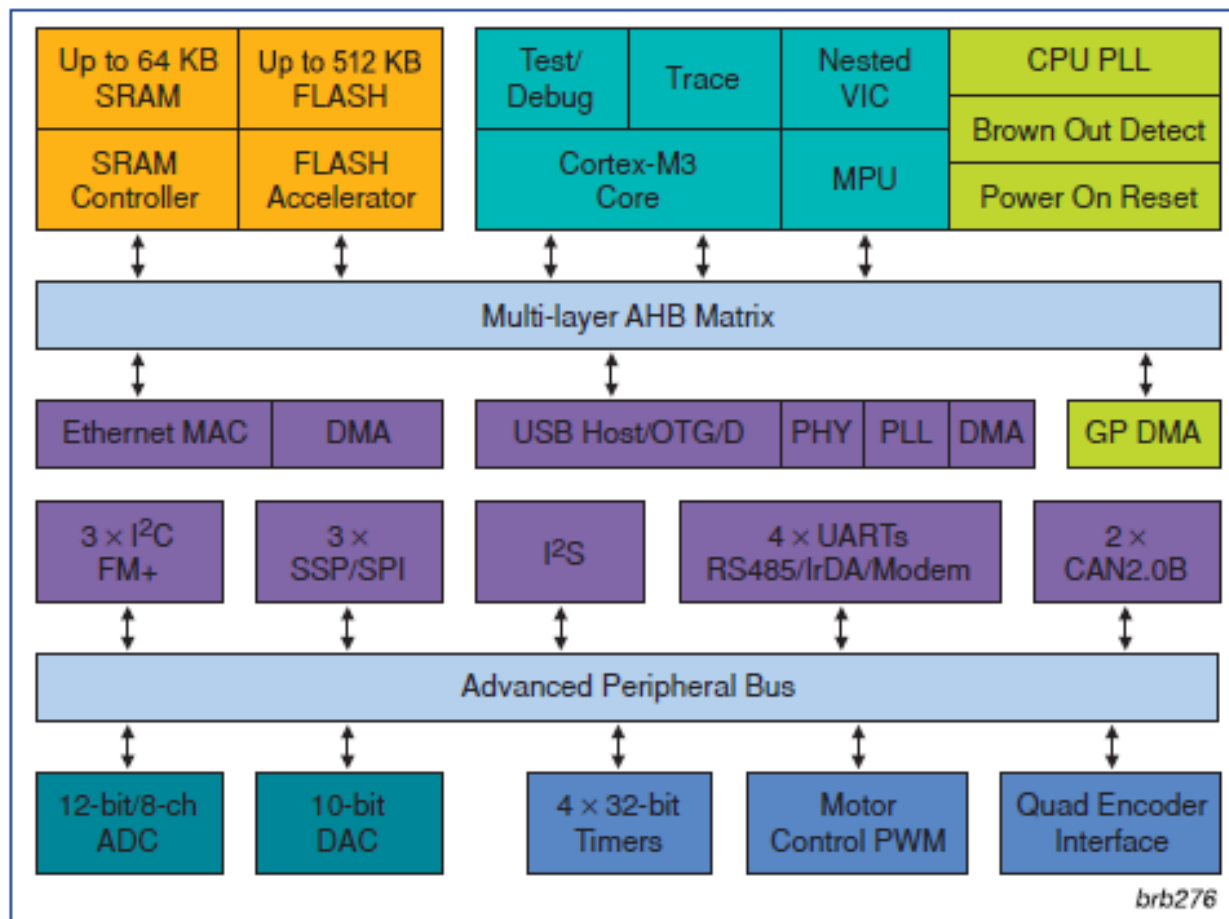
▲  
performance / functionality



# LPC1700 Block Diagram



LPC1700 Block Diagram







# LPC1700 Variations

Part Number	Flash (KB)	SRAM (KB)	Ethernet	USB	CAN	I <sup>2</sup> S	DAC	Package
LPC1768	512	64	Y	Device/Host/OTG	2	Y	Y	LQFP100
LPC1766	256	64	Y	Device/Host/OTG	2	Y	Y	LQFP100
LPC1765	256	64	N	Device/Host/OTG	2	Y	Y	LQFP100
LPC1764	128	32	Y	Device	2	N	N	LQFP100
LPC1758	512	64	Y	Device/Host/OTG	2	Y	Y	LQFP80
LPC1756	256	32	N	Device/Host/OTG	2	Y	Y	LQFP80
LPC1754	128	32	N	Device/Host/OTG	1	N	Y	LQFP80
LPC1752	64	16	N	Device	1	N	N	LQFP80
LPC1751	32	8	N	Device	1	N	N	LQFP80

# LPC1700 Features

- ▶ Highest bandwidth Ethernet
- ▶ USB On-The-Go/Host/Device adding to the industry's widest choice of USB options
- ▶ Quadrature Encoder Interface and Motor Control Pulse Width Modulator (PWM) for flexible, motor control with power to spare
- ▶ Two CAN interfaces
- ▶ True 12-bit analog-to-digital converter (ADC) and 10-bit digital-to-analog converter (DAC)
- ▶ Fast-Mode Plus (1 Mb/s) I<sup>2</sup>C bus, in addition to 4 UARTs, 3 SPI/SSP buses and an I<sup>2</sup>S bus
- ▶ Real-Time Clock operating at less than 1 uA
- ▶ Pin compatibility with the NXP LPC2300 ARM7 microcontrollers series

# NXP's Portfolio Strengths

Technology	NXP offers ...
<b>USB</b> 	<ul style="list-style-type: none"><li>▶ Widest choice<ul style="list-style-type: none"><li>– Leads the industry with more than 45 USB-equipped ARM MCUs</li><li>– Complete USB functionality: device, host, On-The-Go (OTG)</li><li>– Pin- and software-compatible options across NXP families</li></ul></li></ul>
<b>LCD</b> 	<ul style="list-style-type: none"><li>▶ Unique functions, low-cost integration<ul style="list-style-type: none"><li>– Fully integrated, color LCD controllers supporting 24-bpp color and up to 1024 x 768 pixels</li><li>– Simplifies design, reduces board space, lowers cost</li><li>– Software-compatible across NXP families</li></ul></li></ul>
<b>Ethernet</b> 	<ul style="list-style-type: none"><li>▶ Highest bandwidth &amp; Easy to use<ul style="list-style-type: none"><li>– 10/100 Ethernet MAC with multilayered bus structure</li><li>– Optimized performance with DMA hardware acceleration</li><li>– Pin- and software-compatible options across NXP families</li></ul></li></ul>
<b>Motor Control</b> 	<ul style="list-style-type: none"><li>▶ Sophisticated features &amp; Performance to spare<ul style="list-style-type: none"><li>– Dedicated motor-control features support even advanced functions</li><li>– Low overhead extends MCU's capacity, enabling a richer feature set</li></ul></li></ul>



---

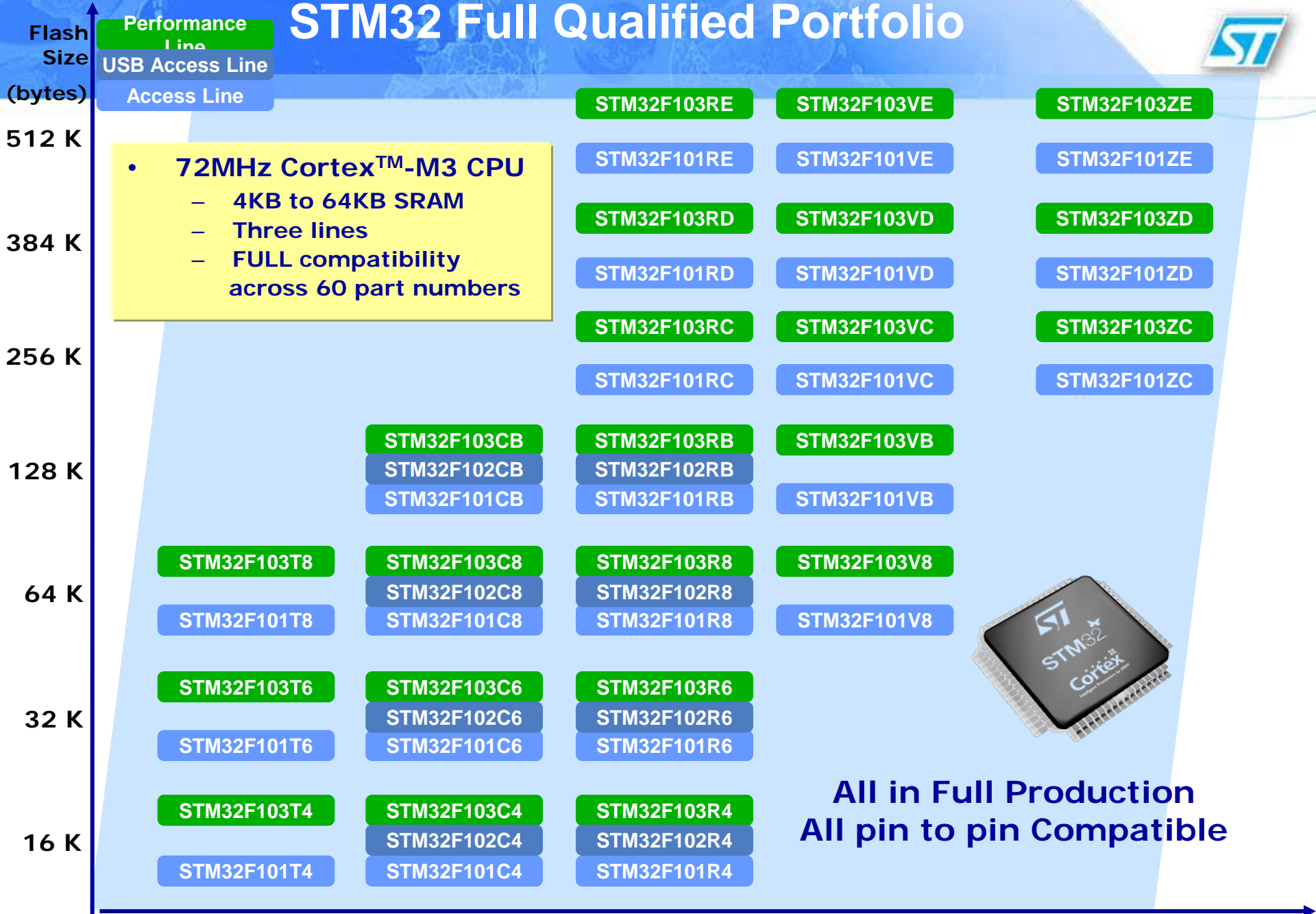
Introduction to the ARM® Cortex™-M Architecture

# ST MICROELECTRONICS



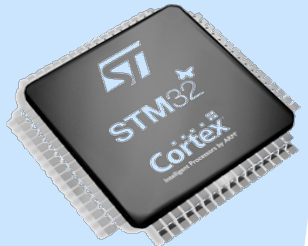


# STM32 Full Qualified Portfolio



• **72MHz Cortex™-M3 CPU**

- 4KB to 64KB SRAM
- Three lines
- FULL compatibility across 60 part numbers



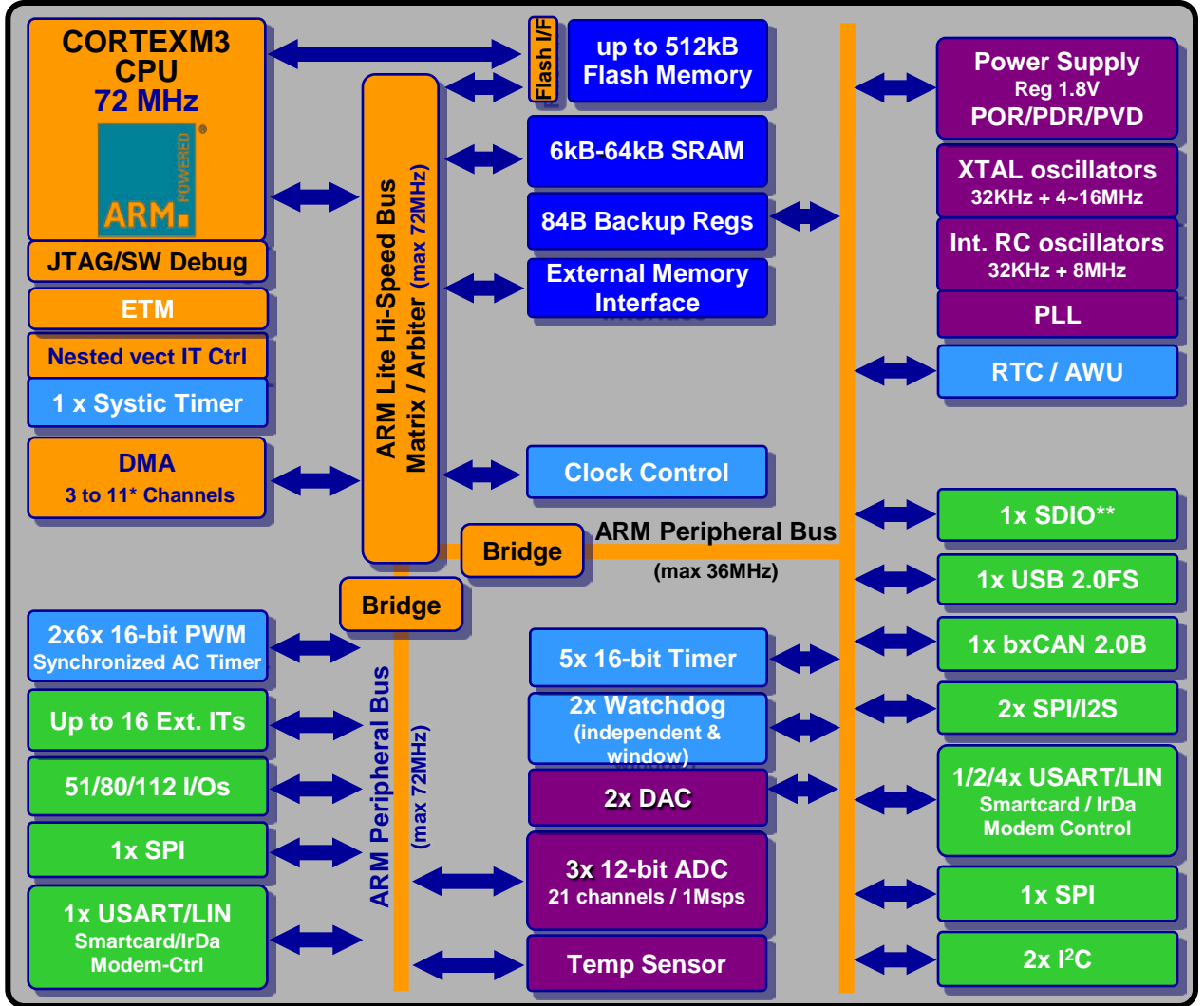
**All in Full Production**  
**All pin to pin Compatible**

- Family Concept:
  - Software & Peripheral compatibility across all STM32.
  - 100% pin-to-pin compatibility per package
- Precise Analog:
  - up to 3x12-bit ADCs @ 1Ms/sec (up to 21channels)
  - Dual-channel 12-bit DAC with buffered outputs
- Optimized Consumption & Battery Operation
  - Run @  $\sim 0.5\text{mA/MHz}$  with half of peripherals ON
  - RTC Mode down to  $1\mu\text{A}$  and  $1.8\text{V}$  (VBAT)
  - Full operation starting  $2.0\text{V}$  (except ADC starting  $2.4\text{V}$ )

- **3-Phase Motor Control**
  - Up to 2 dedicated Timers, i.e. drive 2 motors in parallel
  - Incl. dead-time generation & HW synchro with ADCs
  - Full library covering all types of BLDC motors
- **Safety Support**
  - Dual Watchdog with separated clocks
  - Automatic switch to internal RC if crystal fails
  - CRC module, e.g. for memory check
  - Pre-Certified Library for IEC60335 Class B
- **Exhaustive Software libraries (8) & Application Notes (25) available.**

# STM32F103 Performance 512K

- 2V-3.6V Supply
- 5V tolerant I/Os
- Excellent safe clock modes
- Low-power modes with wake-up
- Internal RC
- Embedded reset
- 36 pins to 100 pins (BGA 5x5, QFN 6x6 available)
- -40/+105°C



# STM32 Connectivity Line STM32F105/107



Both lines include up

Up to 256KB FLASH

Multiple com.  
Peripherals

USART, SPI, I2C

Multiple 16-bit  
TIMERS

Dual DAC

ETM

Main Osc 3-16MHz

Internal 8 MHz RC  
and 40 kHz RC

Real Time Clock

2 x Watchdogs

Reset circuitry

2 x 12-bit ADC 1µs  
Temp sensor

PWM Timer

Up to 12 channels DMA

80% GPIO ratio



## STM32F107

72MHz  
CPU

Up to  
64KB  
SRAM

USB 2.0  
OTG FS

2xCAN  
2.0B

2x  
Audio  
Class  
I<sup>2</sup>S

Ethernet  
IEEE1588

## STM32F105

72MHz  
CPU

Up to  
64KB  
SRAM

USB 2.0  
OTG FS

2xCAN  
2.0B

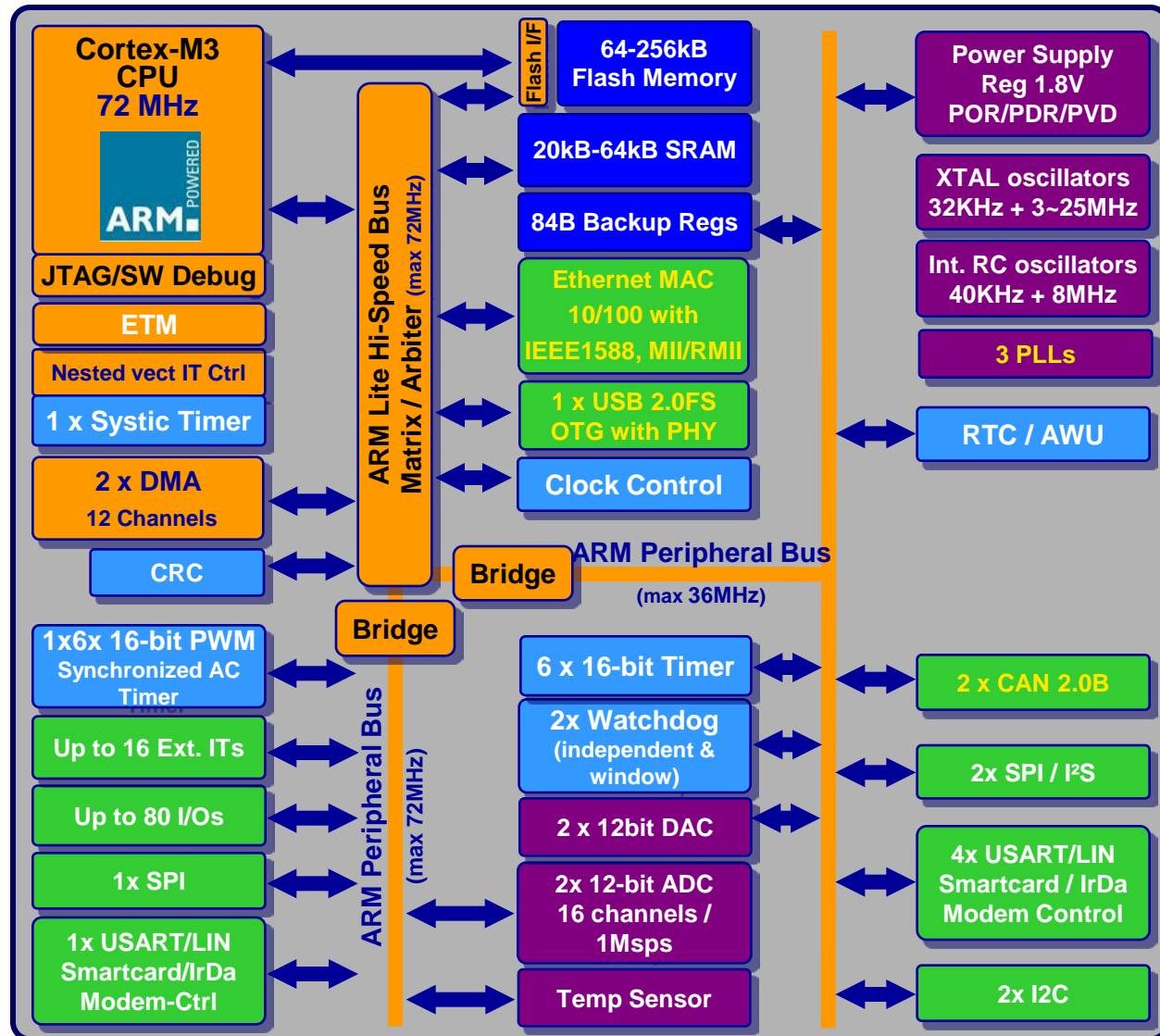
2x  
Audio  
Class  
I<sup>2</sup>S

New features

# STM32F105/7 Connectivity Line



- Up to 256KB Flash / up to 64KB SRAM
- Ethernet 10/100 MAC with IEEE1588, MII & RMII
- USB 2.0 FS OTG w/ OTG PHY
- Preloaded bootloader:
  - USART+CAN+USB DFU
- 2x Audio Classe I<sup>2</sup>S interfaces
- 2x CAN 2.0B with dedicated 512B buffer
- LQFP64, LQFP100
- -40/+105°C



# STM32 Evolution



## NOW

**STM32F10x**  
16K-512K Flash  
Access/Performance  
lines

## NEXT

**HIGH PERFORMANCE**  
MIPS/MHz/MEMORY  
**HIGH BANDWIDTH**  
PERIPHS

**Q2 10**

**STM32  
Plus !**

Alarm & Security  
Factory automation  
Audio, Metering

**CONNECTIVITY**  
USB OTG , ETHERNET ,  
STORAGE MEDIA ,

**June**

**STM32  
Connectivity**

Factory automation  
Audio, Medical  
Gaming, Metering

**ULTRA LOW POWER**

**09**

**LCD**  
Ultra low static leakage  
Ultra low uA/MHz  
Low power analog

**Q1 10**

**STM32L**

GlucO meter  
Gas & power meter  
Portable GPS, DAB  
Road Tolling

**COST REDUCTIONS**  
Lower MHz  
Simple Peripherals

**Q4 09**

**STM32  
Value Line**

Printer, Home audio  
Toys, 3D remote  
Appliance



---

Introduction to the ARM® Cortex™-M Architecture

# **TEXAS INSTRUMENTS LUMINARY**





# Luminary Micro, Inc.

LUMINARY MICRO

---

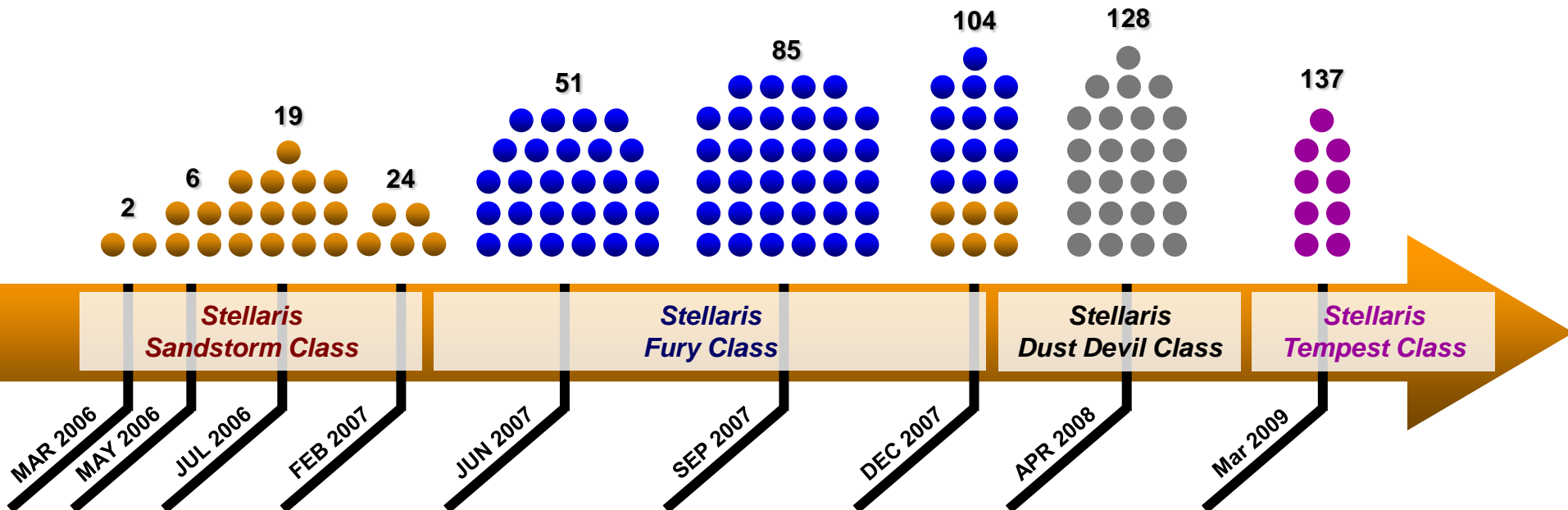


**The Frost & Sullivan  
2008 Global Entrepreneurial  
Company of the Year**



# Luminary Micro's Stellaris Family of Industrial-Grade 32-bit MCUs

- ... the first 32-bit ARM® microcontroller offered for \$1.00
- ... the first available ARM Cortex™-M3-based microcontrollers in production
- ... the first to bring serious motion control capability to the ARM architecture
- ... the only to integrate 10/100 Ethernet MAC and PHY in an ARM architecture
- ... the largest ARM portfolio in the world
- ... now featuring over 100 (and growing!) compatible family members – all available today!
- ... a *complete* and *fully functional* StellarisWare™ Peripheral Driver Library
- ... extensive world-class third-party tools support
- ... optimized for battery-backed applications





# Stellaris® Family Technology

## ARM® Cortex™-M3 v7-M Processor Core

Up to 100 MHz  
Up to 125 MIPS (at 100 MHz)

## On-chip Memory

256 KB Flash; 96 KB SRAM  
ROM loaded with Stellaris DriverLib, BootLoader,  
AES tables, and CRC

## External Peripheral Interface (EPI)

32-bit dedicated parallel bus for external peripherals  
Supports SDRAM, SRAM/Flash, M2M

## Advanced Serial Integration

10/100 Ethernet MAC and PHY  
3 CAN 2.0 A/B Controllers  
USB (full speed) OTG / Host / Device  
3 UARTs with IrDA and ISO 7816 support\*  
2 I<sup>2</sup>Cs  
2 Synchronous Serial Interfaces (SSI)  
Integrated Interchip Sound (I<sup>2</sup>S)

## System Integration

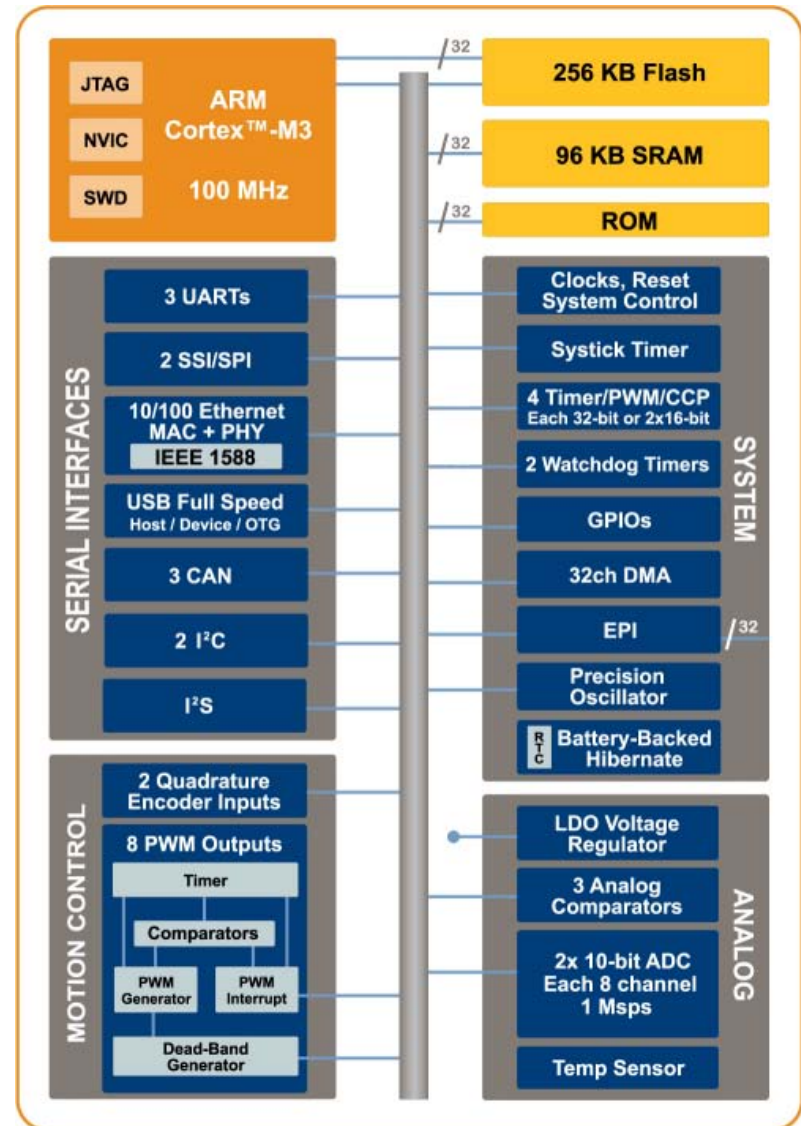
32-channel DMA Controller  
Internal Precision 16MHz Oscillator  
Two watchdog timers with separate clock domains  
ARM Cortex Systick Timer  
4 32-bit timers (up to 8 16-bit) with RTC capability  
Lower-power battery-backed hibernation module  
Flexible pin-muxing capability

## Advanced Motion Control

8 advanced PWM outputs for motion and energy applications  
2 Quadrature Encoder Inputs (QEI)

## Analog

2x 8-ch 10-bit ADC (for a total of 16 channels)  
3 analog comparators  
On-chip voltage regulator (1.2V internal operation)

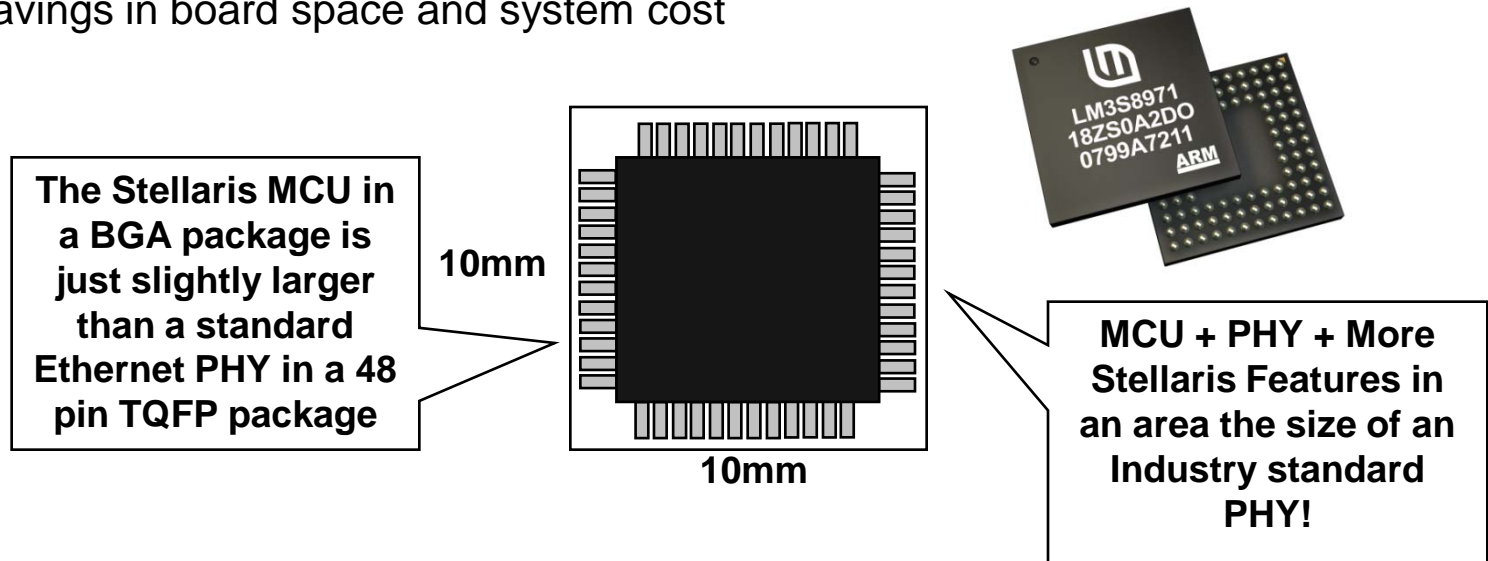




# Stellaris Family: Unique Value Proposition

## The only ARM MCU with 10/100 Ethernet MAC / PHY

- Enables network connectivity and embedded web servers
- Lower external power budget requirements than solutions using an external PHY
- Savings in board space and system cost



And now even more value in the same small package:

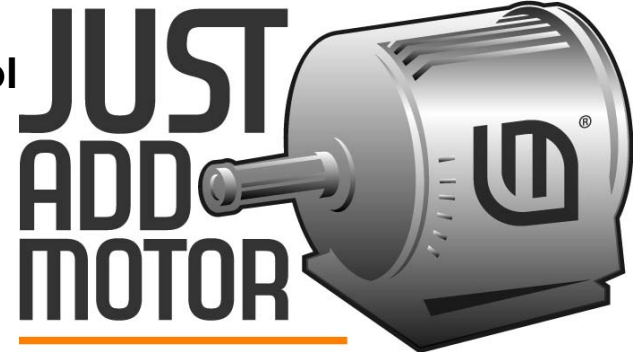
- Hardware support for Precision Time Protocol (IEEE 1588 PTP)



# Stellaris Family: Unique Value Proposition

## Stellaris Motion Control Advantage!

- **Most competitors do not even have motion-control Pulse Width Modulators (PWMs)! (e.g. NXP)**
- **Stellaris supports up to 8 general-purpose PWMs and up to 8 channels of motion control PWMs.**
- **General-purpose PWMs**
  - **Stellaris 16-bit timer simple PWM mode with programmable output negation.**
- **Motion-control PWM Module**
  - **Can generate simple PWM signals for a simple charge pump.**
  - **Can generate paired PWM signals with dead-band delays for a half-H bridge driver.**
  - **Can generate the full six channels of gate controls for a 3-Phase inverter bridge.**
  - **Dead-band generator providing shoot-through protection.**
  - **Synchronization of timers enables precise alignment of *all* edges.**
- **Stellaris Exclusive! Up to 4 fault-condition handling inputs in hardware quickly provide low-latency shutdown.**
- **Stellaris Exclusive! Up to 2 Quadrature Encoder Inputs provide accurate positioning for closed-feedback control.**








# Stellaris Family: Unique Value Proposition

## Stellaris features single-cycle Flash memory up to 50MHz!

- Some competitors claim faster core speeds with ARM7 and Cortex-M3, but the flash is not single-cycle!
- Some competitors claim single-cycle, but the max core speed is very limited

Vendor	MCU Line	Flash Access Time 20MHz CPU	Flash Access Time 25MHz CPU	Flash Access Time 50MHz CPU	Unit of Measure
Luminary Micro	Stellaris				<b>Cycle</b>
ST Micro	STM32	1	2	3	Cycles
Atmel	AVR8	1	n/a	n/a	Cycles
TI	MSP430	n/a	n/a	n/a	Cycles

*Flash access specifications from published datasheets*



# Stellaris Family: Unique Value Proposition

---

## StellarisWare™

**Free license and royalty-free source code:**

- **Peripheral Driver Library**
- **Graphics Library**
- **USB Library**
- **Boot Loader**
- **IEC 60730 Library**

**Enabling our customers with the ability to rapidly develop and deploy their products at competitive costs yielding a higher overall value for the Stellaris solution!**

---

Introduction to the ARM® Cortex™-M Architecture

**TOSHIBA**



# TOSHIBA

Leading Innovation >>>

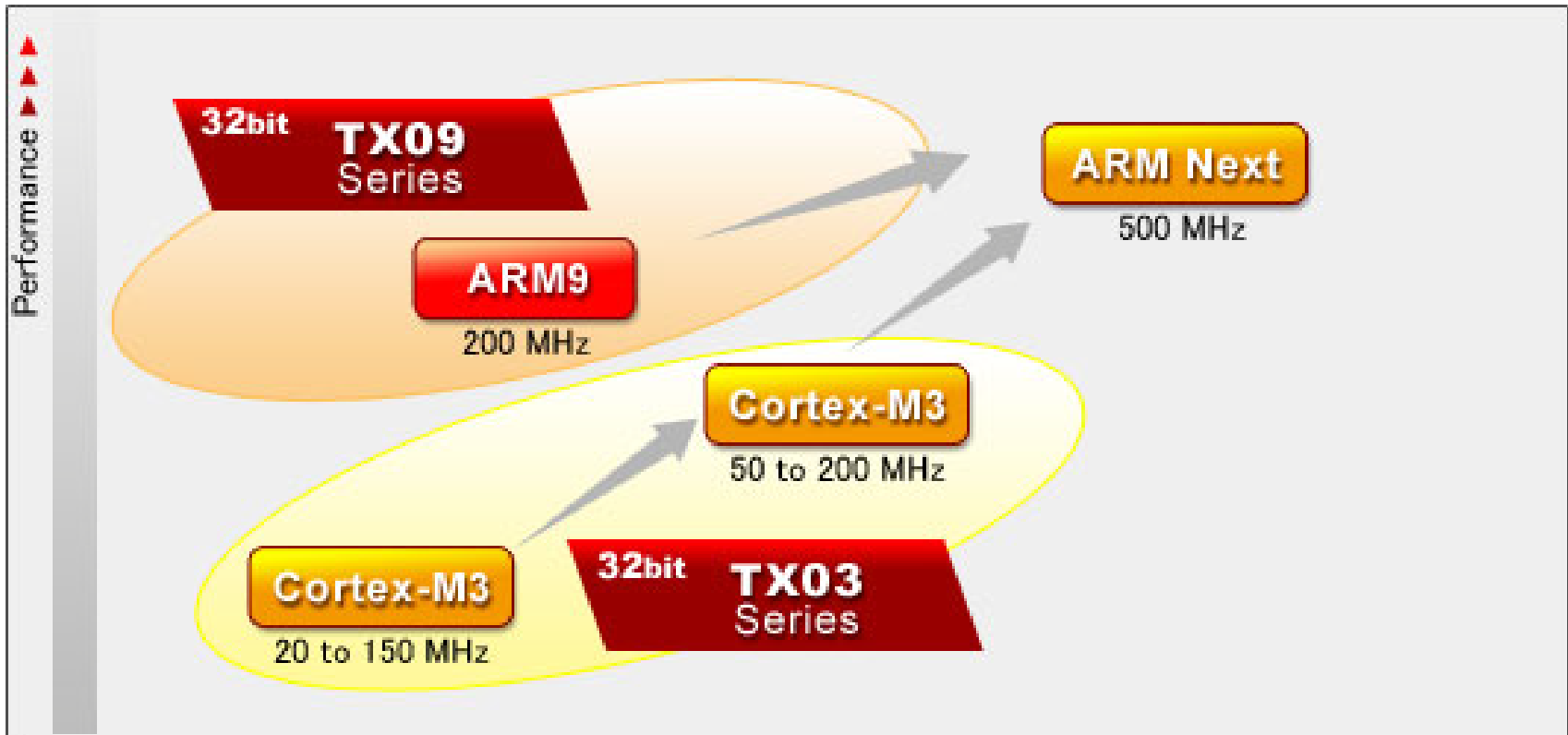


## > TOSHIBA TX03 series MCU

ARM® Cortex™-M3 based MCU

**Cortex™**  
Intelligent Processors by ARM®

# TOSHIBA MCU CPU core roadmap



# Cortex-M3 family concept



**M330**

General purpose application; Low cost derivatives

**M340**

For consumer application like digital camera and camcorder.  
High volume memory;  $\Delta\Sigma$ ADC; ...

**M350**

Automotive devices; IECQ100; PPAP; CAN;...

**M360**

For consumer application like TV and video.  
CEC and remote interface; High volume memory (up to 2MB Flash)

**M370**

OFD

Motor control with vector engine: For industrial and HA market.  
Vop=5V, POR, LVD, OFD; analoge circuit incl. Op-AMPs;

**M380**

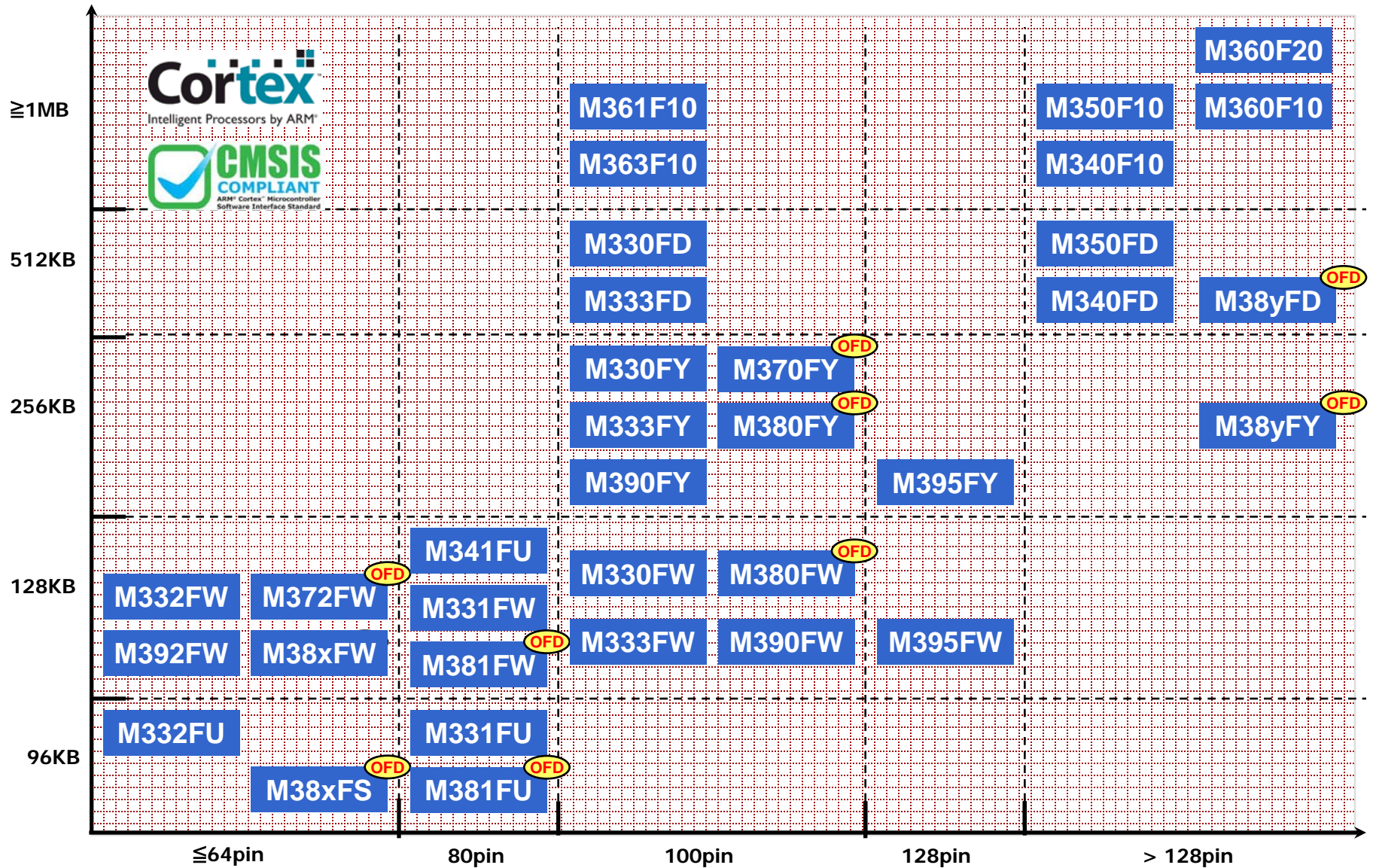
OFD

For industrial and HA market. Vop=5V, POR, LVD, int. Osc., OFD,  
wide product line up (48pin to 144pin package; up to 512kB Flash) ...

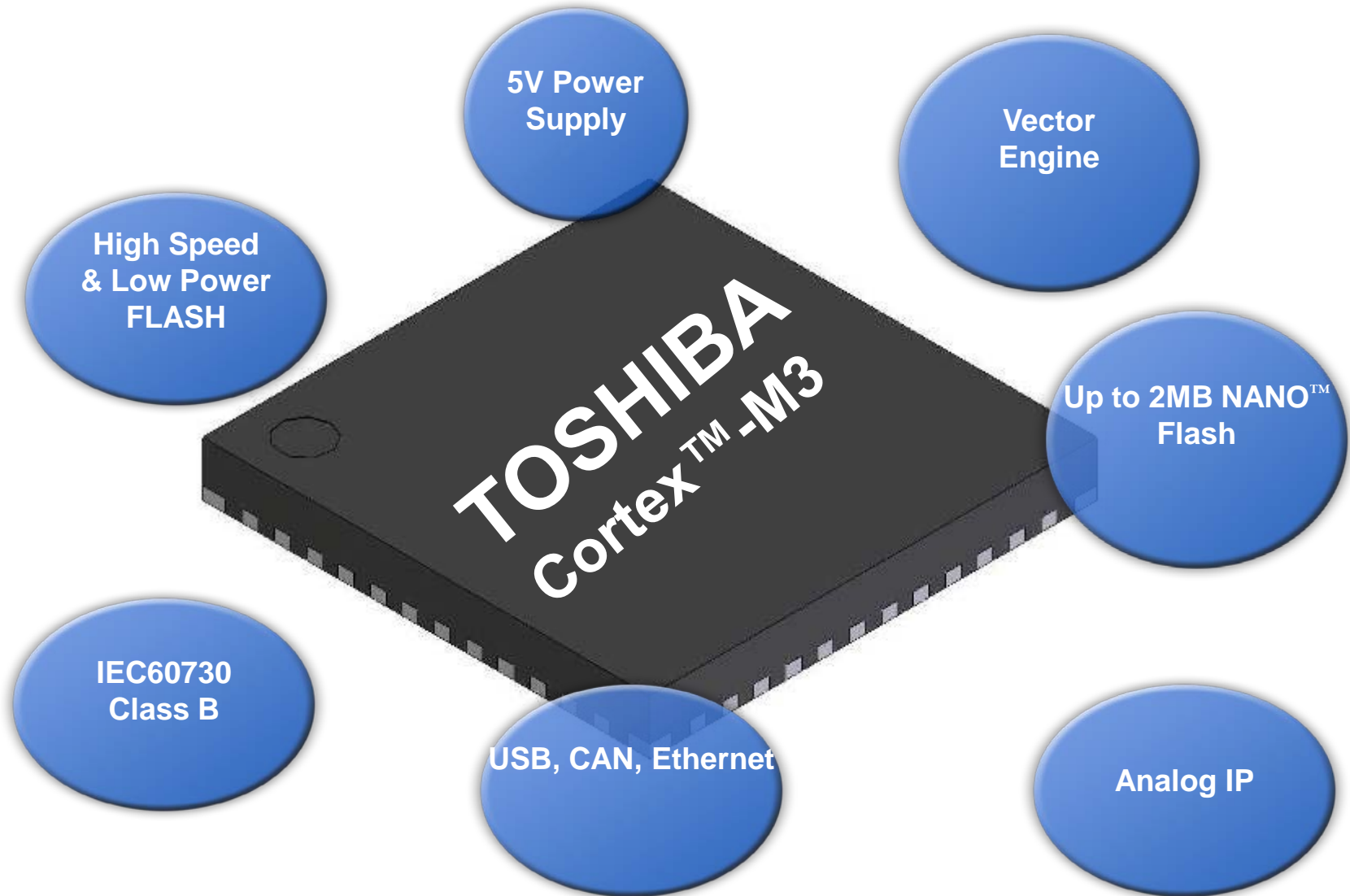
**M390**

For industrial application. Vop=3.3V, POR, LVD, int. Osc., ...

# TX03 series Product Road Map



# TOSHIBA Key Features



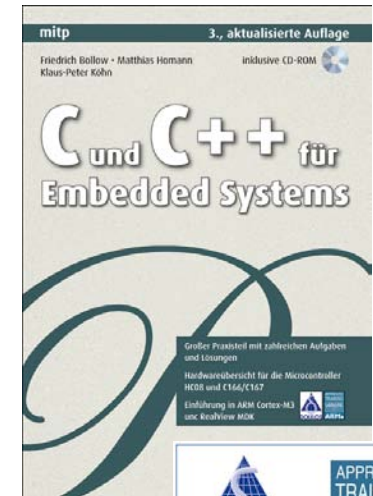
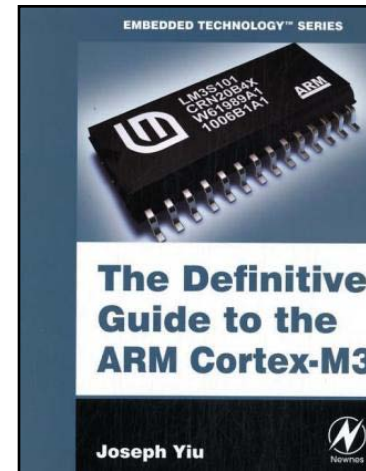
---

Introduction to the ARM® Cortex™-M Architecture

# MORE INFORMATION

# More Information

- **ARM website [www.arm.com](http://www.arm.com)**
  - Technical Reference Manuals
  - Connected Community
- **[www.OnARM.com](http://www.OnARM.com)**
  - Vendor List
  - CMSIS 1.20
  - Tools Overview
- **Books**
  - Joseph Yiu – The Definitive Guide to the ARM Cortex M3
    - ISBN: 978-0-7506-8534-4
  - C und C++ für Embedded Systems
    - ISBN: 978-3-8266-5949-2
- **Training**
  - <http://www.arm.com/support/training.html>
  - <http://www.doulos.com>



# END

Thank you

