Get Better Code Density than 8/16 bit MCU's
NXP LPC1100 Cortex M0

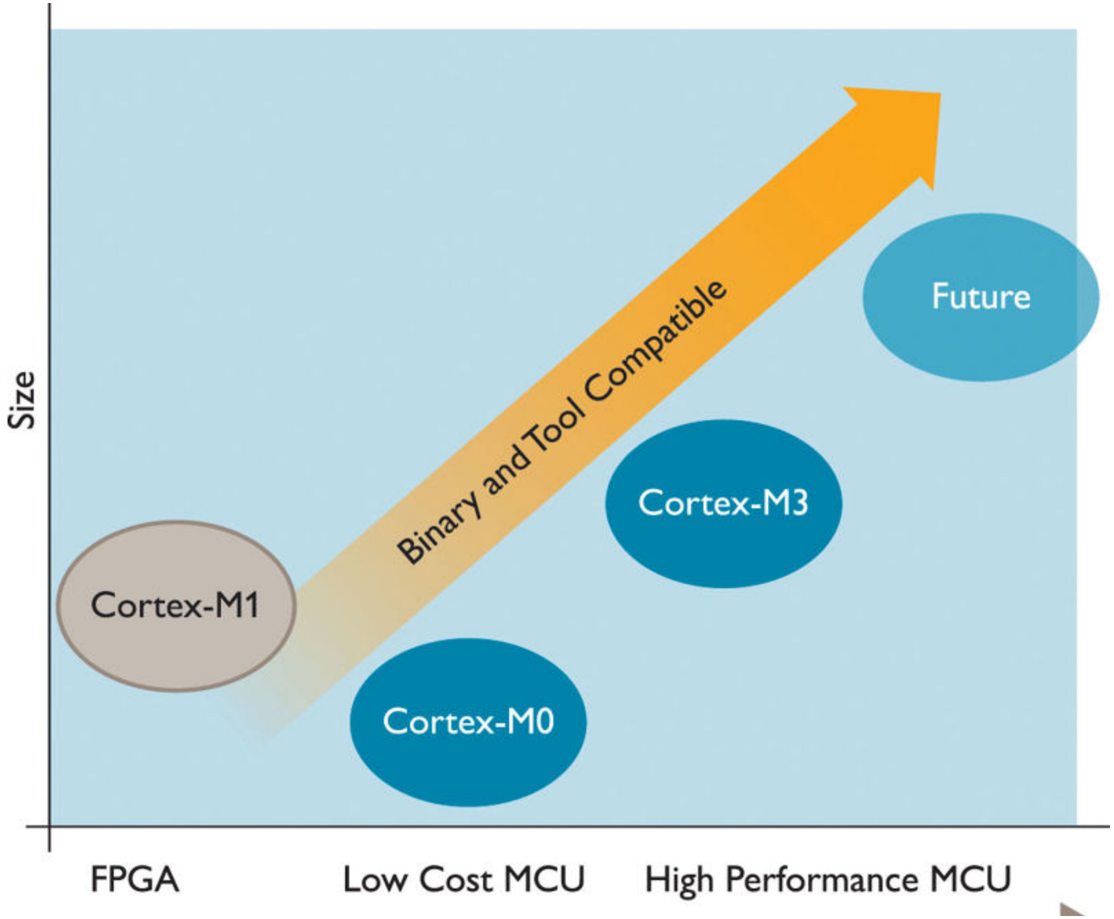
Oct 2009

Outline

- Introduction
- ARM Cortex-M0 processor
- Why processor bit width doesn’t matter
  - Code size
  - Performance
  - Cost
- Conclusions

ARM Cortex-M Processors

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



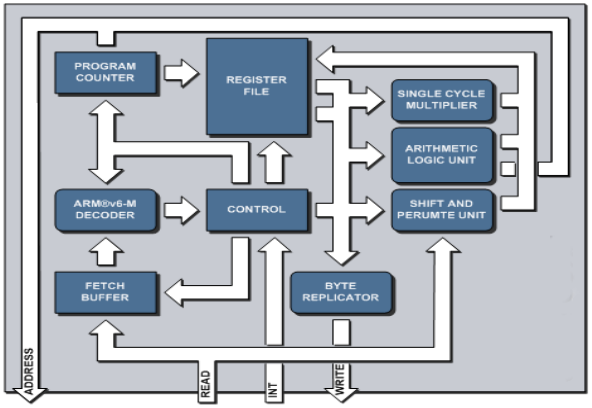
ARM Cortex-A Series:
Applications processors for feature-rich OS and user applications

ARM Cortex-R Series:
Embedded processors for real-time signal processing and control applications

ARM Cortex-M Series:
Deeply embedded processors optimized for microcontroller and low-power applications

ARM Cortex-M0 Processor

- **32-bit ARM RISC processor**
  - Thumb 16-bit instruction set



- **Very power and area optimized**
  - Designed for low cost, low power
- **Automatic state saving on interrupts and exceptions**
  - Low software overhead on exception entry and exit
- **Deterministic instruction execution timing**
  - Instructions always takes the same time to execute*

*Assumes deterministic memory system

Thumb instruction set

- **32-bit operations, 16-bit instructions**
  - Introduced in ARM7TDMI (‘T’ stands for Thumb)
  - Supported in every ARM processor developed since
  - Smaller code footprint

- **Thumb-2**
  - All processor operations can all be handled in ‘Thumb’ state
  - Enables a performance optimised blend of 16/32-bit instructions
  - Supported in all Cortex processors

Instruction set architecture

- **Based on 16-bit Thumb ISA from ARM7TDMI**
  - Just 56 instructions, all with guaranteed execution time
  - 8, 16 or 32-bit data transfers possible in one instruction

Thumb
User assembly code, compiler generated

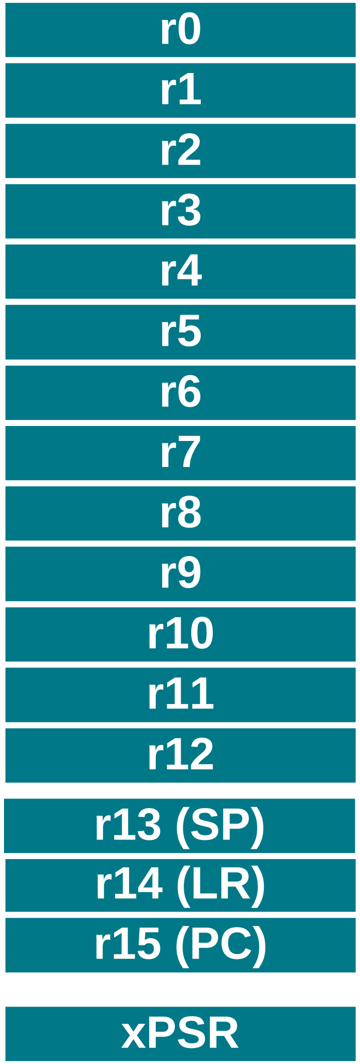
ADC	ADD	ADR	AND	ASR	B
BIC	BL		BX	CMN	CMP
EOR	LDM	LDR	LDRB	LDRH	LDRSB
LDRSH	LSL	LSR	MOV	MUL	MVN
ORR	POP	PUSH	ROR	RSB	SBC
STM	STR	STRB	STRH	SUB	SVC
TST	BKPT	BLX	CPS	REV	REV16
REVSH	SXTB	SXTH	UXTB	UXTH	

Thumb-2
System, OS

NOP	
SEV	WFE
WFI	YIELD
DMB	
DSB	
ISB	
MRS	
MSR	

Program registers

- **All registers are 32-bit wide**
  - Instructions exist to support 8/16/32-bit data
- **13 general purpose registers**
  - Registers r0 – r7 (Low registers)
  - Registers r8 – r12 (High registers)
- **3 registers with special meaning/usage**
  - Stack Pointer (SP) – r13
  - Link Register (LR) – r14
  - Program Counter (PC) – r15
- **Special-purpose registers - xPSR**



Instruction behaviour

- **Most instructions occupy 2 bytes of memory**

```
a = a * b;          C code
MUL r0, r1;         Assembler
```

- **When executed, complete in a fixed time**
  - **Data processing (e.g. add, shift, logical OR) take 1 cycle**
  - **Data transfers (e.g. load, store) take 2 cycles**
  - **Branches, when taken, take 3 cycles**

- **The instructions operate on 32-bit data values**
  - **Processor registers and ALU are 32-bit wide!**

Thumb instructions

- **Cortex M0 requires instruction fetches to be half word aligned**
- **Thumb instructions are aligned on a two-byte boundaries**

MSByte	**MSByte -1**	**LSByte + 1**	**LSByte**
Word at Address A			
Halfword at Address A+2		Halfword at Address A	
Byte at Address A+3	Bye at Address A+2	Byte at Address A+1	Byte at Address A

- **32 bit instructions are organized as 2 half words**

32-bit Thumb instruction, hw1				32-bit Thumb instruction, hw2			
15	8	7	0	15	8	7	0
Byte at Address A+1		Byte at Address A		Byte at Address A+3		Byte at Address A+2	

Nested Vectored Interrupt Controller

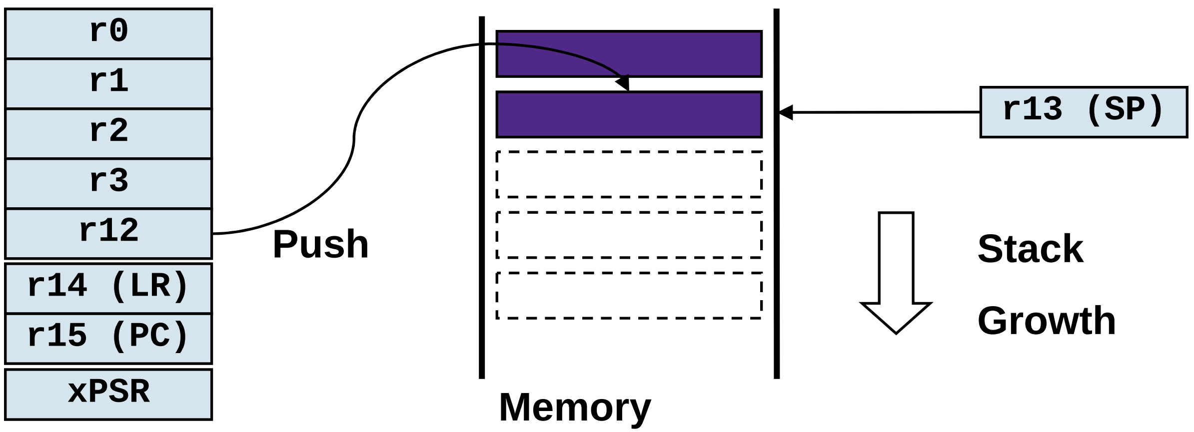
- **NVIC enables efficient exception handling**
  – Integrated within the processor - closely coupled with the core
  – Handles system exceptions & interrupts

- **The NVIC includes support for**
  – Prioritization of exceptions
  – Tail-chaining & Late arriving interrupts

- **Fully deterministic exception handling timing behavior**
  – Always takes the same number of cycles to handle an exception
  – Fixed at 16 clocks for no jitter
  – Register to trade off latency versus jitter

- **Everything can be written in C**

Interrupt behaviour



- On interrupt, hardware automatically stacks corruptible state
- Interrupt handlers can be written fully in C
  - Stack content supports C/C++ ARM Architecture Procedure Calling Standard
- Processor fetches initial stack pointer from 0x0 on reset

Writing interrupt handlers

Traditional approach

- **Exception table**
  - Fetch instruction to branch
- **Top-level handler**
  - Routine handles re-entrancy

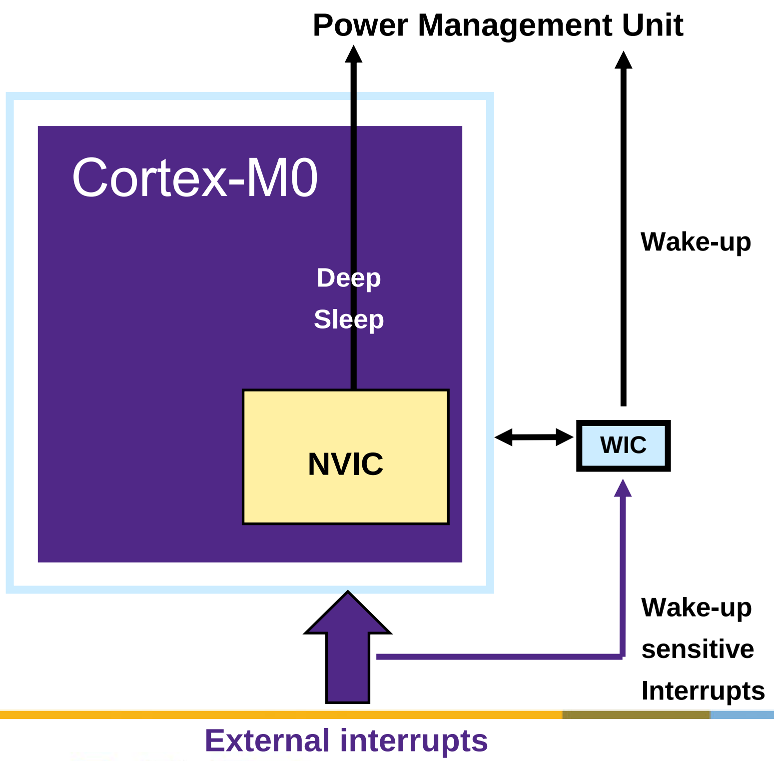
```
IRQVECTOR
        LDR     PC, IRQHandler

IRQHandler PROC
    STMFD sp!,{r0-r4,r12,lr}
    MOV r4,#0x80000000
    LDR r0,[r4,#0]
    SUB sp,sp,#4
    CMP r0,#1
    BLEQ C_int_handler
    MOV r0,#0
    STR r0,[r4,#4]
    ADD sp,sp,#4
    LDMFD sp!,{r0-r4,r12,lr}
    SUBS pc,lr,#4
ENDP
```

ARM Cortex-M family

- **NVIC automatically handles**
  - Saving corruptible registers
  - Exception prioritization
  - Exception nesting
- **ISR can be written directly in C**
  - Pointer to C routine at vector
  - ISR is a C function
- **Faster interrupt response**
  - With less software effort
- **WFI, sleep on exit**

Software support for sleep modes

- **ARM Cortex-M family has 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-M0 can be put into state retention
- **WIC signals wake-up to PMU**
  - Core can be woken almost instantaneously
  - React to critical external events

Instruction set comparison

ADC ADD ADR AND ASR B CLZ
BFC BFI BIC CDP CLREX CBNZ CBZ CMN
CMP DBG EOR LDC
LDMIA LDMDB LDR LDRB
LDRBT LDRD LDREX LDREXB
LDREXH LDRH LDRHT LDRSB
LDRSBT LDRSHT LDRSH LDRT
MCR LSL LSR MLS
MCRR MLA MOV MOVT
MRC MRRC MUL MVN
NOP ORN ORR PLD
PLDW PLI POP PUSH
RBIT REV REV16 REVSH
ROR RRX RSB SBC
SBFX SDIV SEV SMLAL
SMULL SSAT STC STMIA
STMDB STR STRB STRBT
STRD STREX STREXB STREXH STRH STRHT STRT
SUB SXTB SXTH TBB TBH TEQ TST
UBFX UDIV UMLAL UMULL USAT UXTB UXTH
WFE WFI YIELD IT

BKPT BLX ADC ADD ADR
BX CPS AND ASR B
DMB BL BIC
DSB CMN CMP EOR
ISB LDR LDRB LDM
MRS LDRH LDRSB LDRSH
MSR LSL LSR MOV
NOP REV MUL MVN ORR
REV16 REVSH POP PUSH ROR
SEV SXTB RSB SBC STM
SXTH UXTB STR STRB STRH
UXTH WFE SUB SVC TST
WFI YIELD

CORTEX-M0

CORTEX-M3

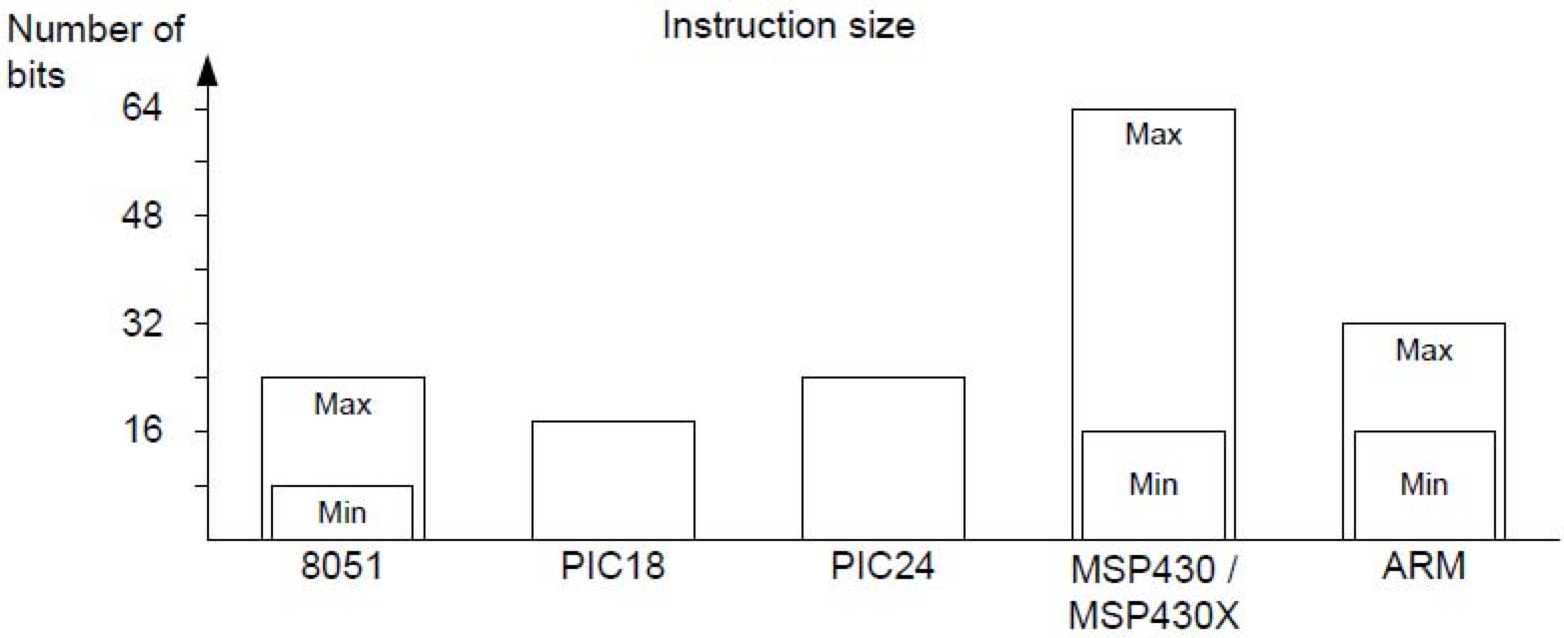
Present in ARM7TDMI

Code Size

Code size of 32 bits versus 16/8bit MCU’s

- **The instruction size of 8 bit MCU’s is not 8 bits**
  - 8051 is 8 to 24 bits
  - PIC18 is 18 bits
  - PIC16 is 16 bits
- The instruction size of 16 bit MCU’s is not 16 bits
  - MSP430 can be up to 32bits and the extended version can be up to 64 bits
  - PIC24 is 24 bits
- **<u>The instruction size for M0 is mostly 16 bits</u>**

Code size of 32 bits versus 16/8bit MCU's



16-bit multiply example

- **Consider an device with a 10-bit ADC**
  – Basic filtering of data requires a 16-bit multiply operation
  – 16-bit multiply operation is compared below

8-bit example	16-bit example	ARM Cortex-M0

`MOV A, XL ; 2 bytes`
`MOV B, YL ; 3 bytes`
`MUL AB; 1 byte`
`MOV R0, A; 1 byte`
`MOV R1, B; 3 bytes`
`MOV A, XL ; 2 bytes`
`MOV B, YH ; 3 bytes`
`MUL AB; 1 byte`
`ADD A, R1; 1 byte`
`MOV R1, A; 1 byte`
`MOV A, B ; 2 bytes`
`ADDC A, #0 ; 2 bytes`
`MOV R2, A; 1 byte`
`MOV A, XH ; 2 bytes`
`MOV B, YL ; 3 bytes`
`MUL AB; 1 byte`
`ADD A, R1; 1 byte`
`MOV R1, A; 1 byte`
`MOV A, B ; 2 bytes`
`ADDC A, R2 ; 1 bytes`
`MOV R2, A; 1 byte`
`MOV A, XH ; 2 bytes`
`MOV B, YH ; 3 bytes`
`MUL AB; 1 byte`
`ADD A, R2; 1 byte`
`MOV R2, A; 1 byte`
`MOV A, B ; 2 bytes`
`ADDC A, #0 ; 2 bytes`
`MOV R3, A; 1 byte`
	`MOV R1,&MulOp1`
	`MOV R2,&MulOp2`
	`MOV SumLo,R3`
	`MOV SumHi,R4`
		`MULS r0,r1,r0`

Time: 48 clock cycles* **Code size:** 48 bytes	**Time:** 8 clock cycles **Code size:** 8 bytes	**Time:** 1 clock cycle **Code size:** 2 bytes

* 8051 need at least one cycle per instruction byte fetch as they only have an 8-bit interface

What about Data ?

- **8 bit microcontrollers do not just process 8 bit data**
  - Integers are 16 bits
  - 8 bit microcontroller needs multiple instructions integers
  - C libraries are inefficient
  - Stack size increases
  - Interrupt latency is affected
- **Pointers take multiple Bytes.**
- **M0 can handle Integers in one instruction**
- **M0 can efficiently process 8 and 16 bit data**
  - Supports byte lanes
  - Instructions support half words and bytes. LDR, LDRH, LDRB
- **M0 has efficient Library support**
  - Optimized for M0

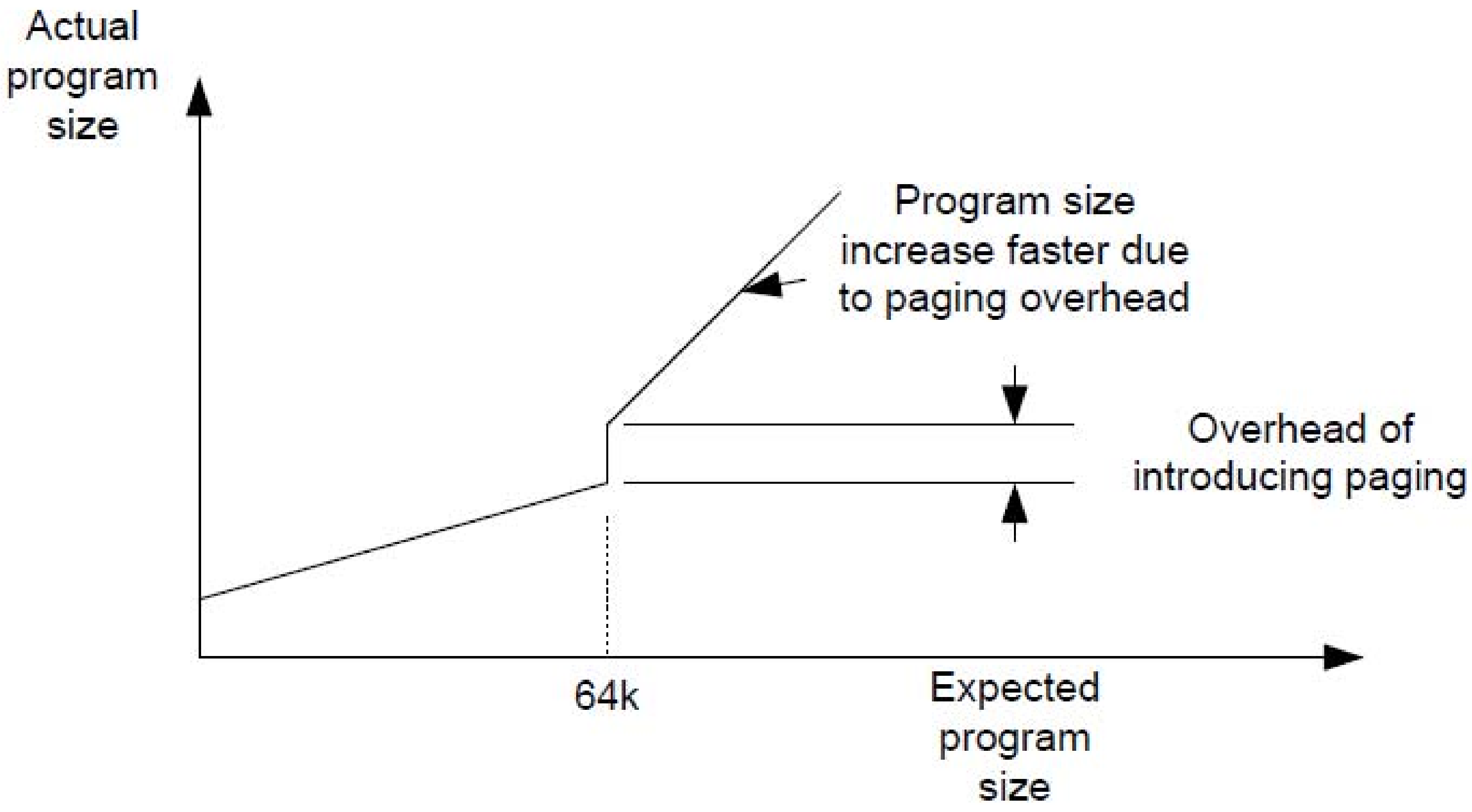
What about Data ?

- For 16 bit processors have issues with
  - Long integers
  - Floating point types
  - Data transfers between processor registers and memory
- 16 bit processors have 16 bit registers
  - Two registers required for 32 bit transfers
  - Increased stack requirements
- M0 has 32 bit registers and 32 bit memories
  - Less cycles for long integers
  - Good floating point performance
  - Less cycles for data transfers

What addressing modes?

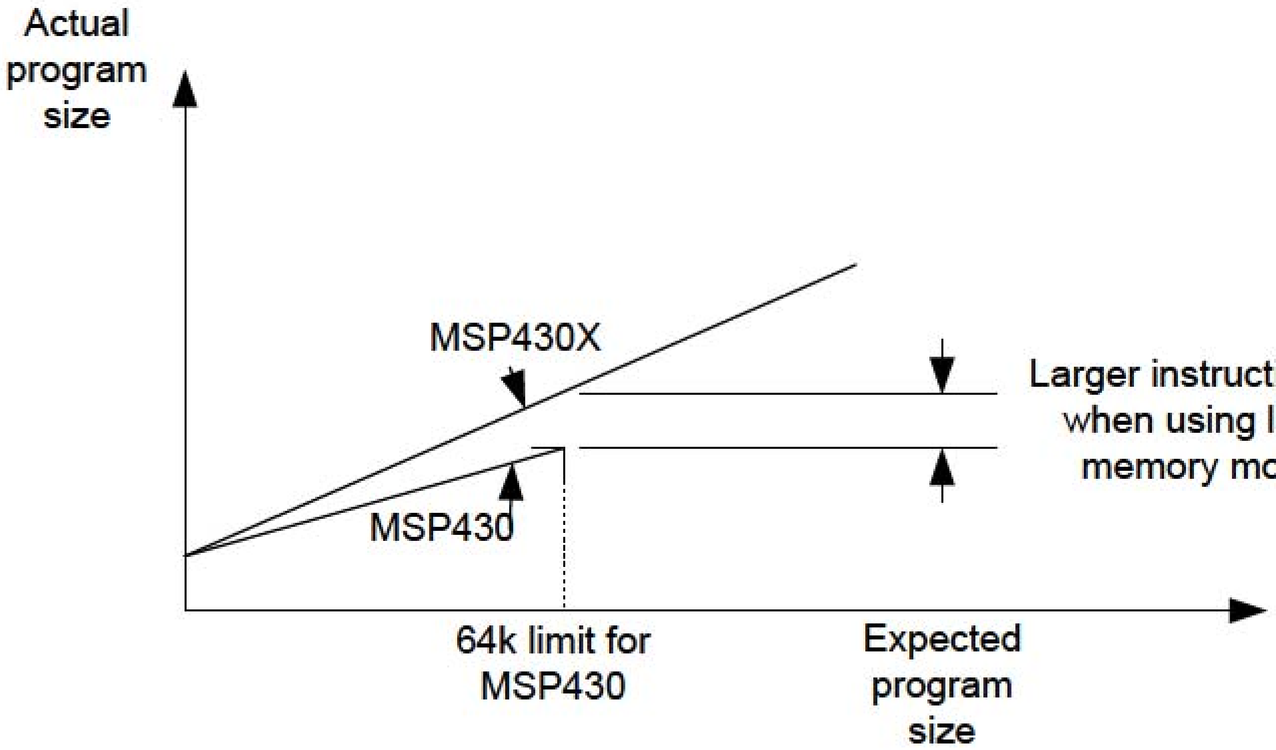
- 16/8 bit processors are limited to 64K of space
  - Data memory limited and segmented
  - Requires banking or extensions to instruction set
  - Memory pointers are extended
    Require multiple instructions and registers
- All cause increased code space
- M0 has a linear 1G address space
  - 32-bit pointers
  - unsigned or signed 32-bit integers
  - unsigned 16-bit or 8-bit integers
  - signed 16-bit or 8-bit integers
  - unsigned or signed 64-bit integers held in two registers.

Code size increase due to paging

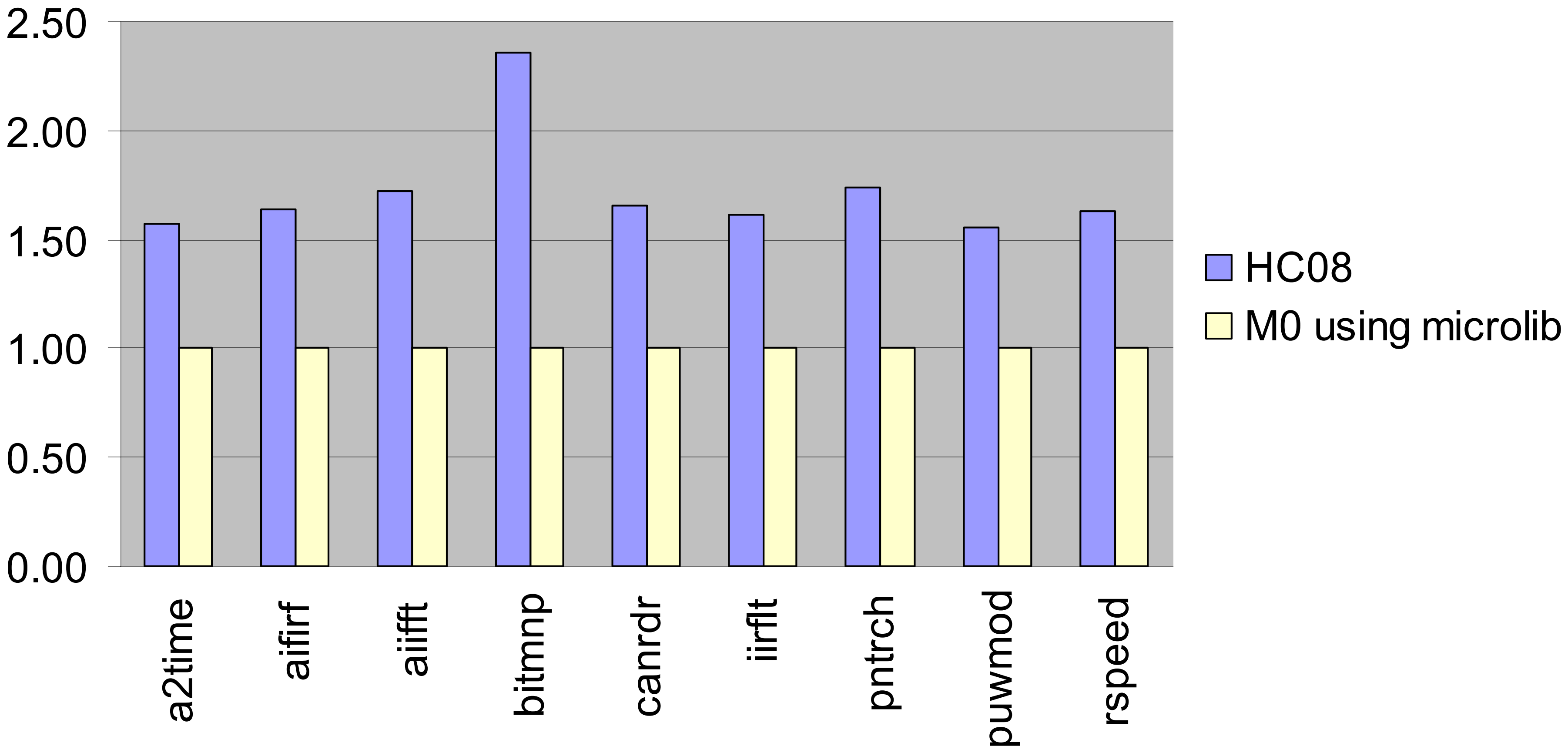


Code size increase for large memory model

(Extended program counter and Registers)

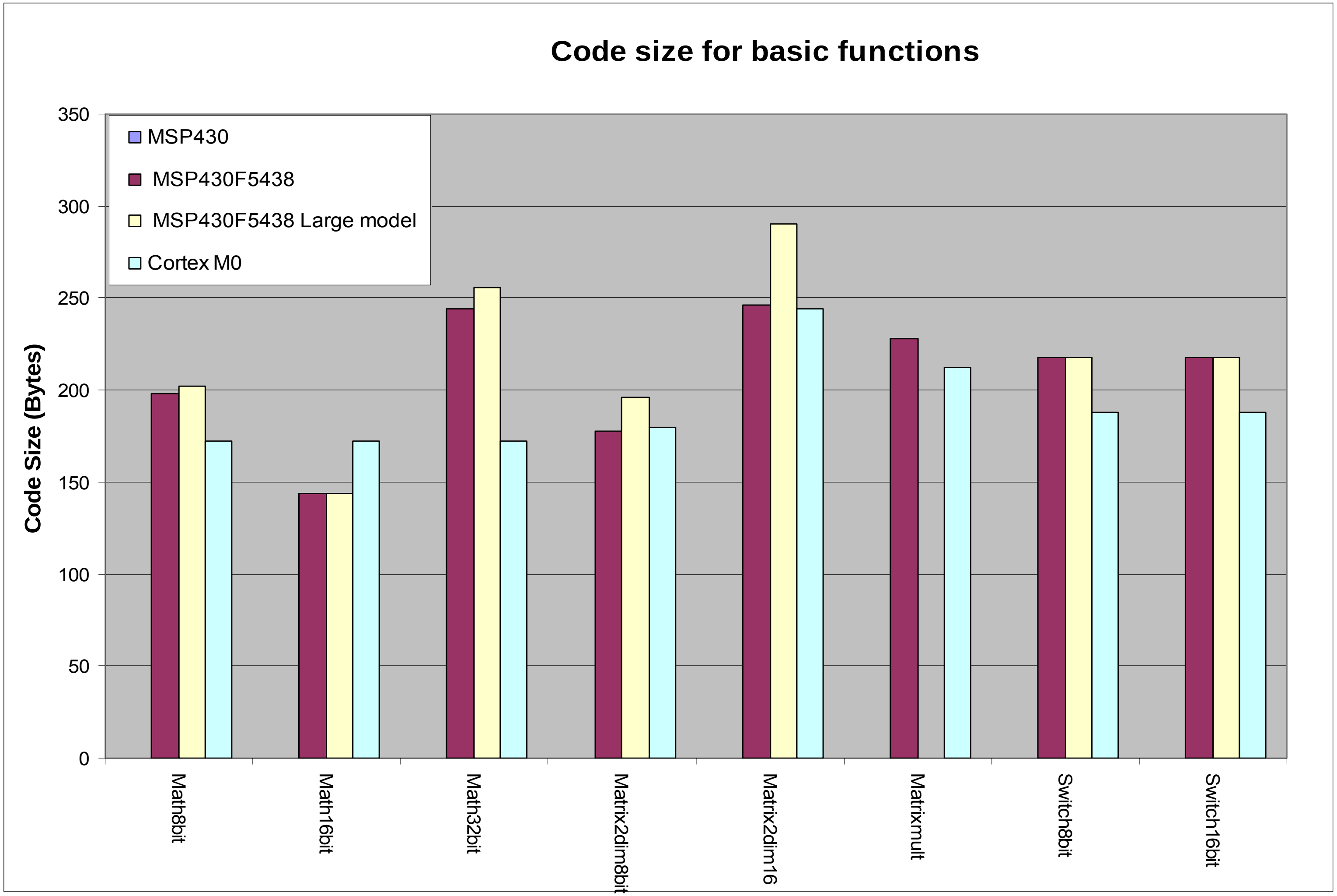


Code Size Performance



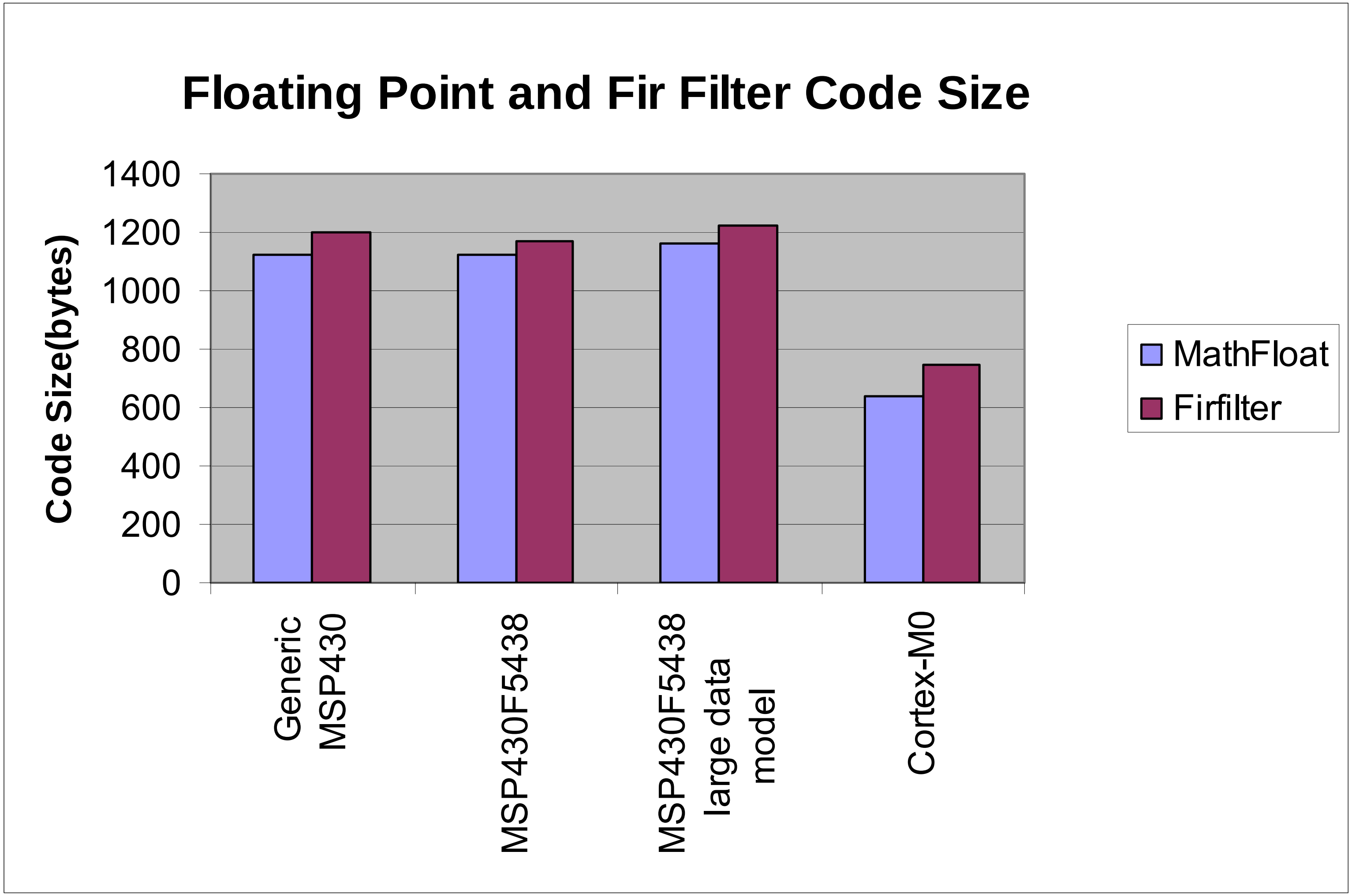
Code Size Performance

- M0 code size is on average 10% smaller than best MSP430 average



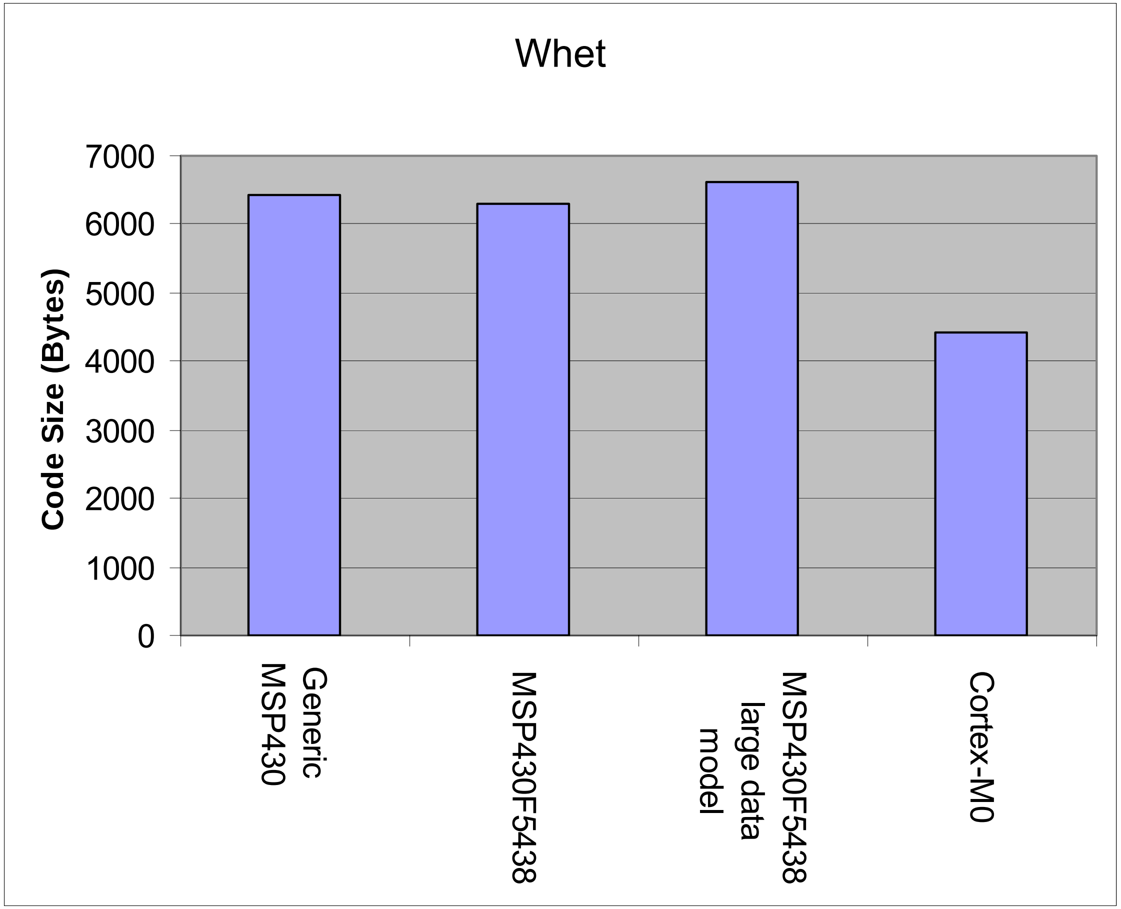
Code Size Performance

- M0 code size is 42% and 36% smaller than best MSP430 generic



Code Size Performance

- M0 code size is 30% smaller than MSP430F5438



What is CoreMark?

- Simple, yet sophisticated
  - Easily ported in hours, if not minutes
  - Comprehensive documentation and run rules
- Free, but not cheap
  - Open C code source download from EEMBC website
  - Robust CPU core functionality coverage
- Dhrystone terminator
  - The benefits of Dhrystone without all the shortcomings
    - Free, small, easily portable
    - CoreMark does real work

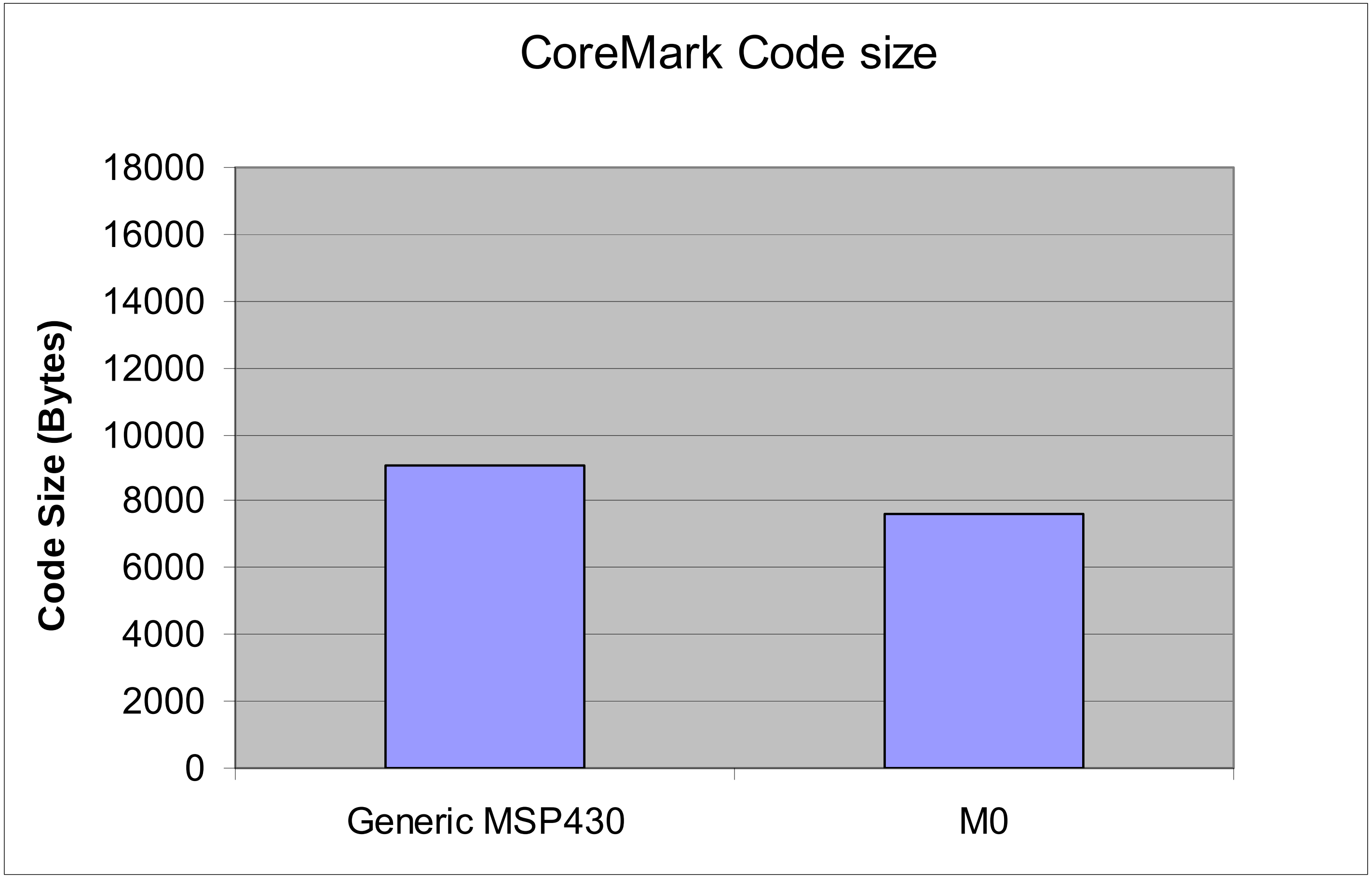
CoreMark Workload Features

- Matrix manipulation allows the use of MAC and common math ops
- Linked list manipulation exercises the common use of pointers
- State machine operation represents data dependent branches
- Cyclic Redundancy Check (CRC) is very common embedded function

- Testing for:
  - A processor's basic pipeline structure
  - Basic read/write operations
  - Integer operations
  - Control operations

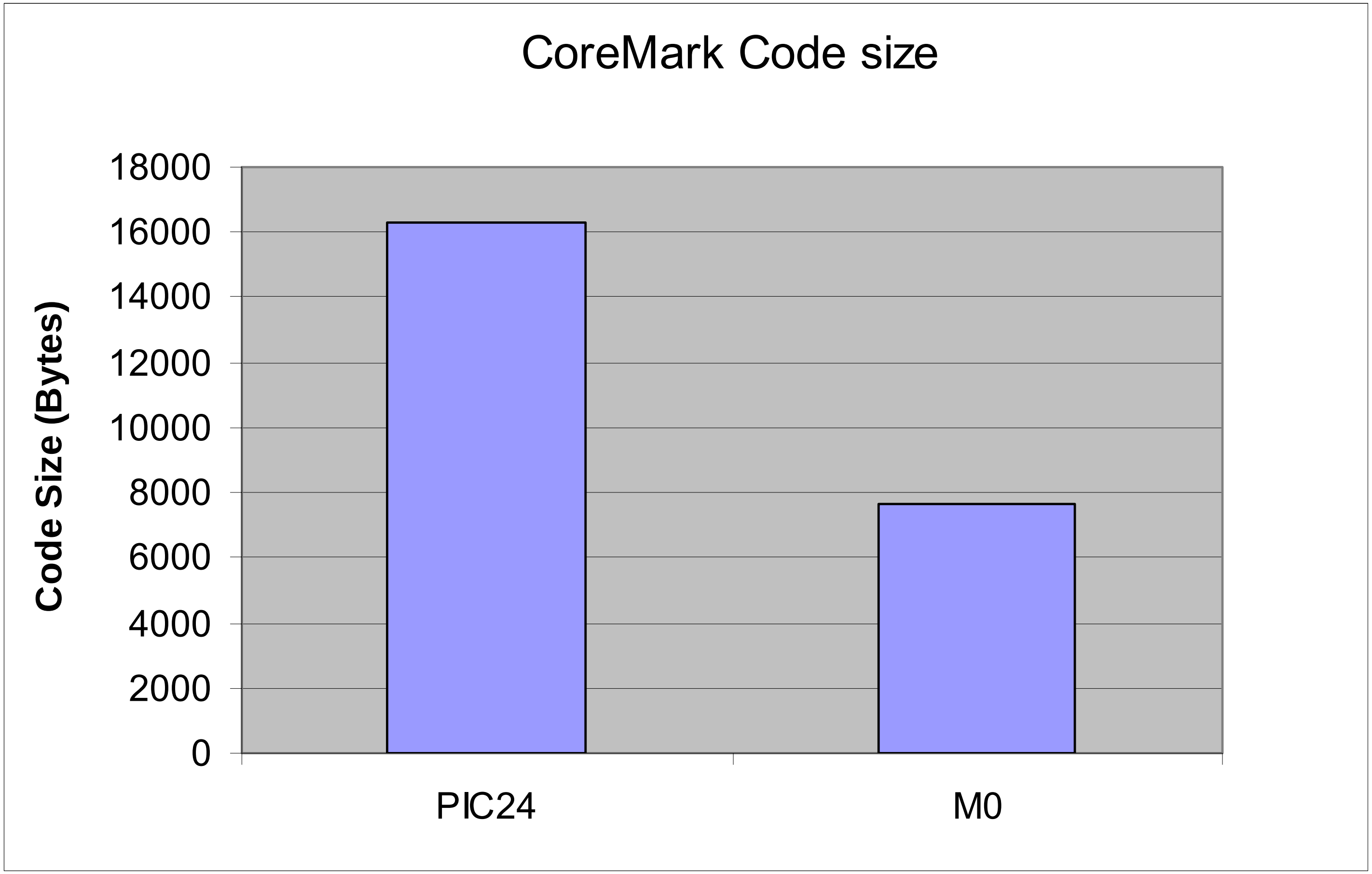
Code Size Performance (CoreMark)

- M0 code size is 16% smaller than generic MSP430



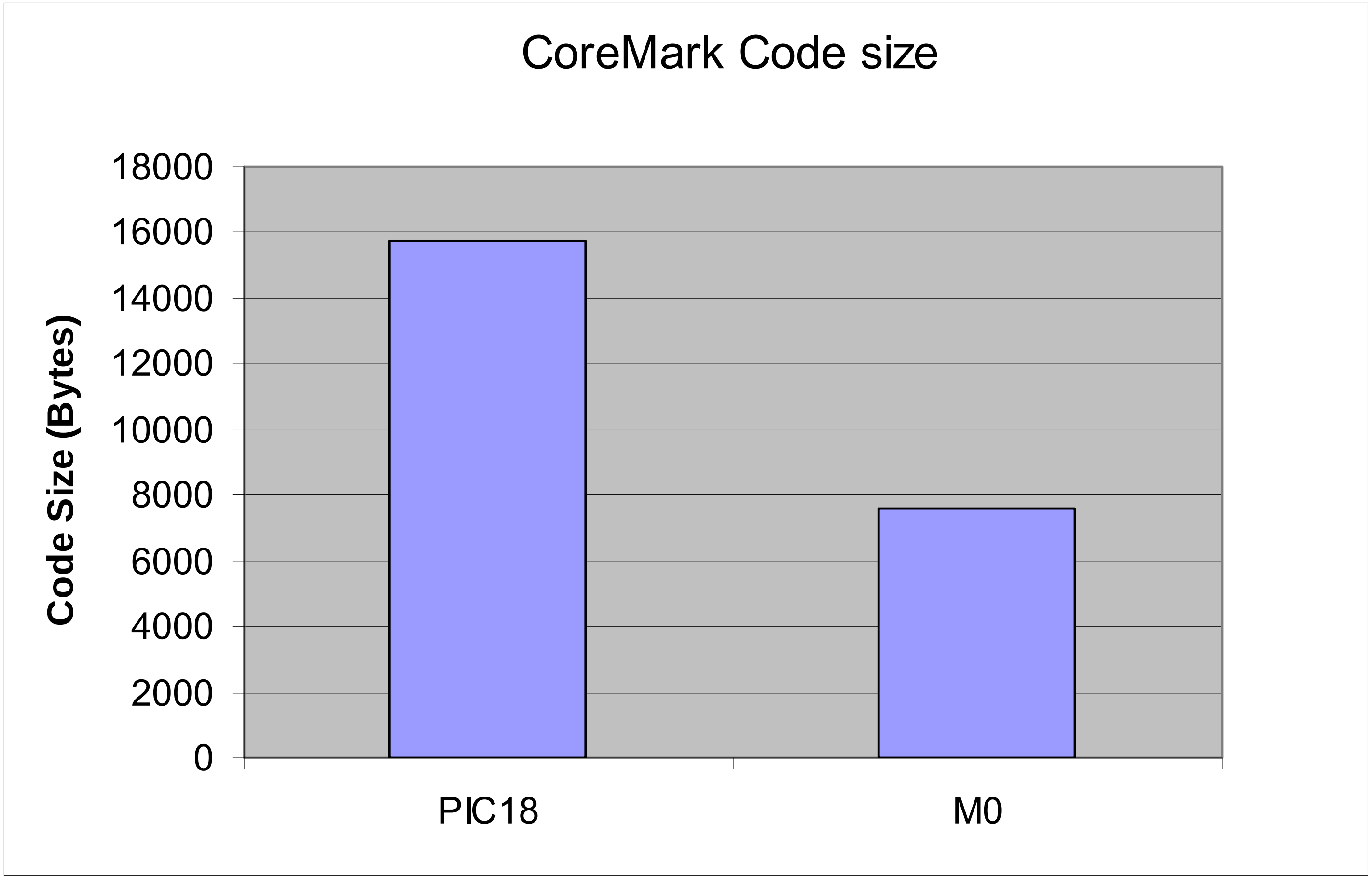
Code Size Performance (CoreMark)

- M0 code size is 53% smaller than PIC24



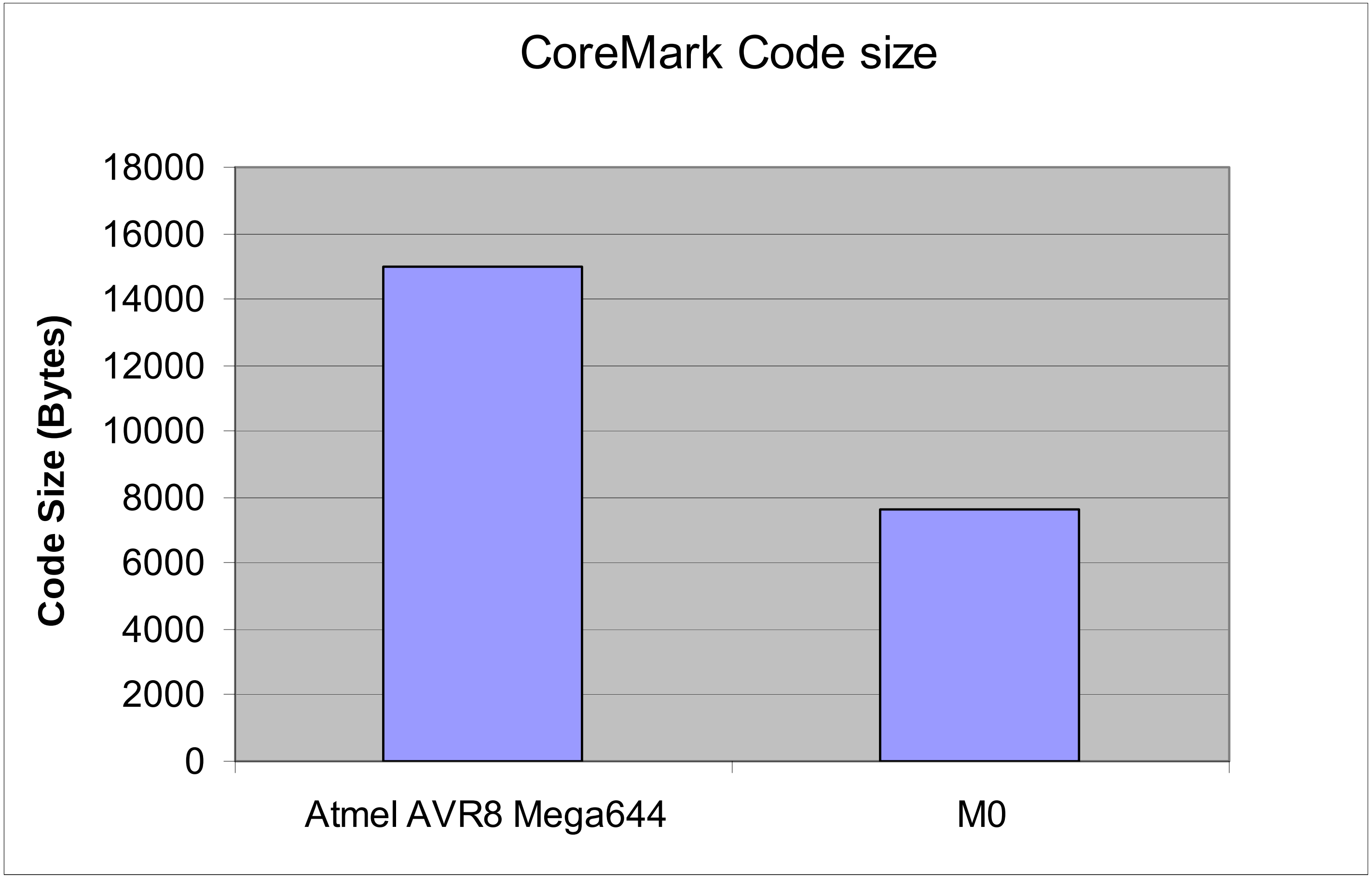
Code Size Performance (CoreMark)

- M0 code size is 51% smaller than PIC18



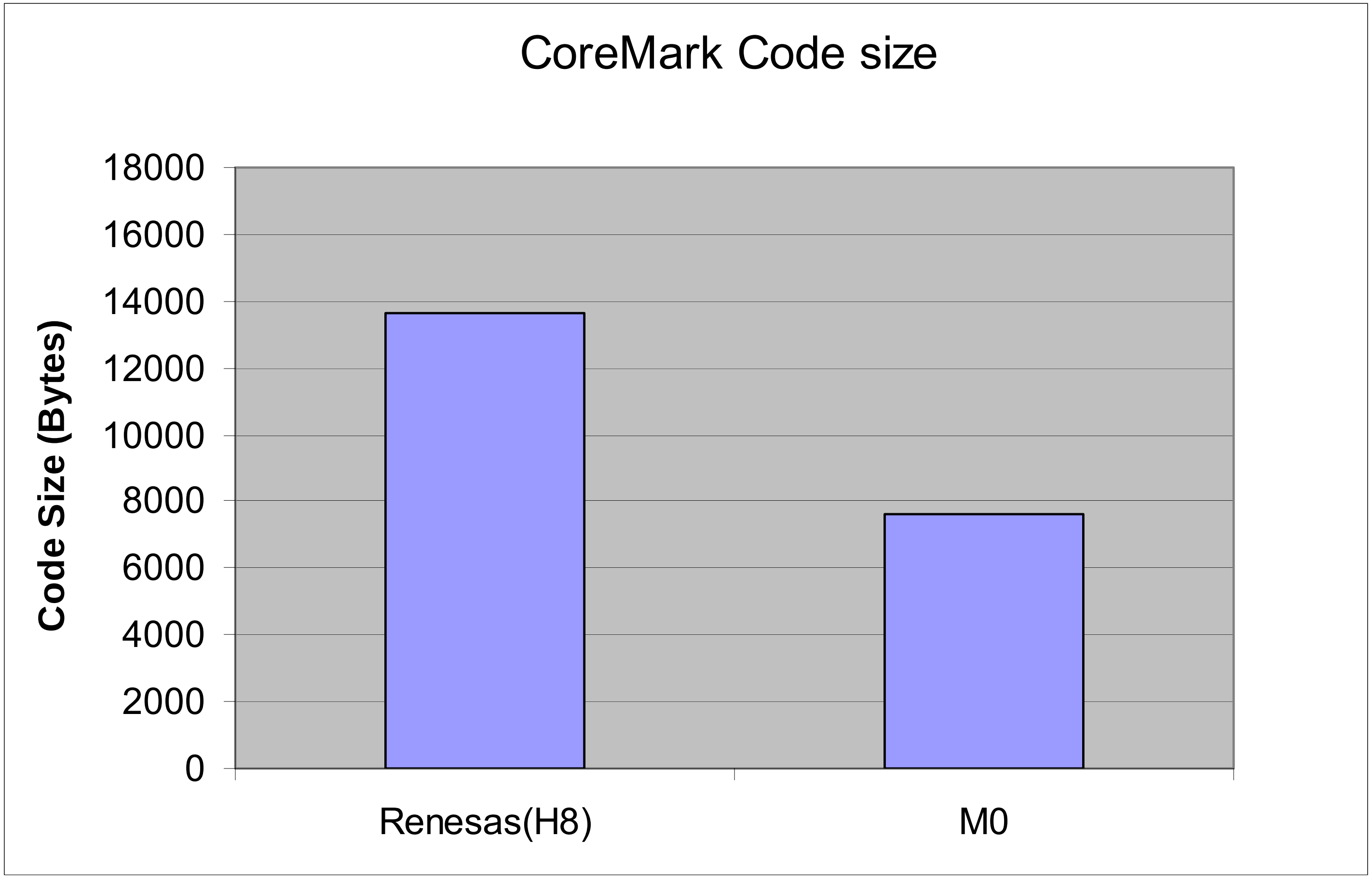
Code Size Performance (CoreMark)

- M0 code size is 49% smaller than Atmel AVR8



Code Size Performance (CoreMark)

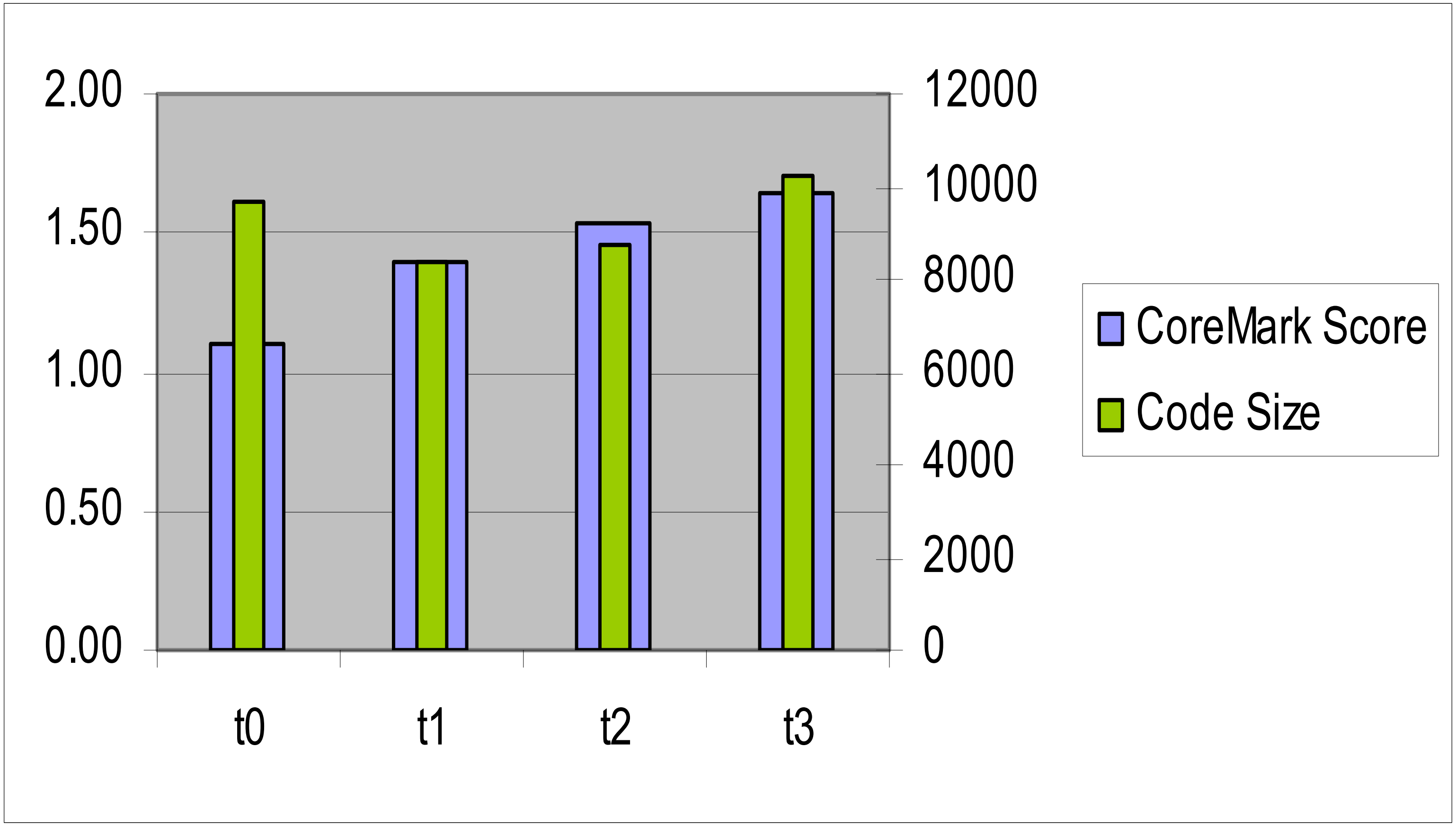
- M0 code size is 44% smaller than Renesas H8



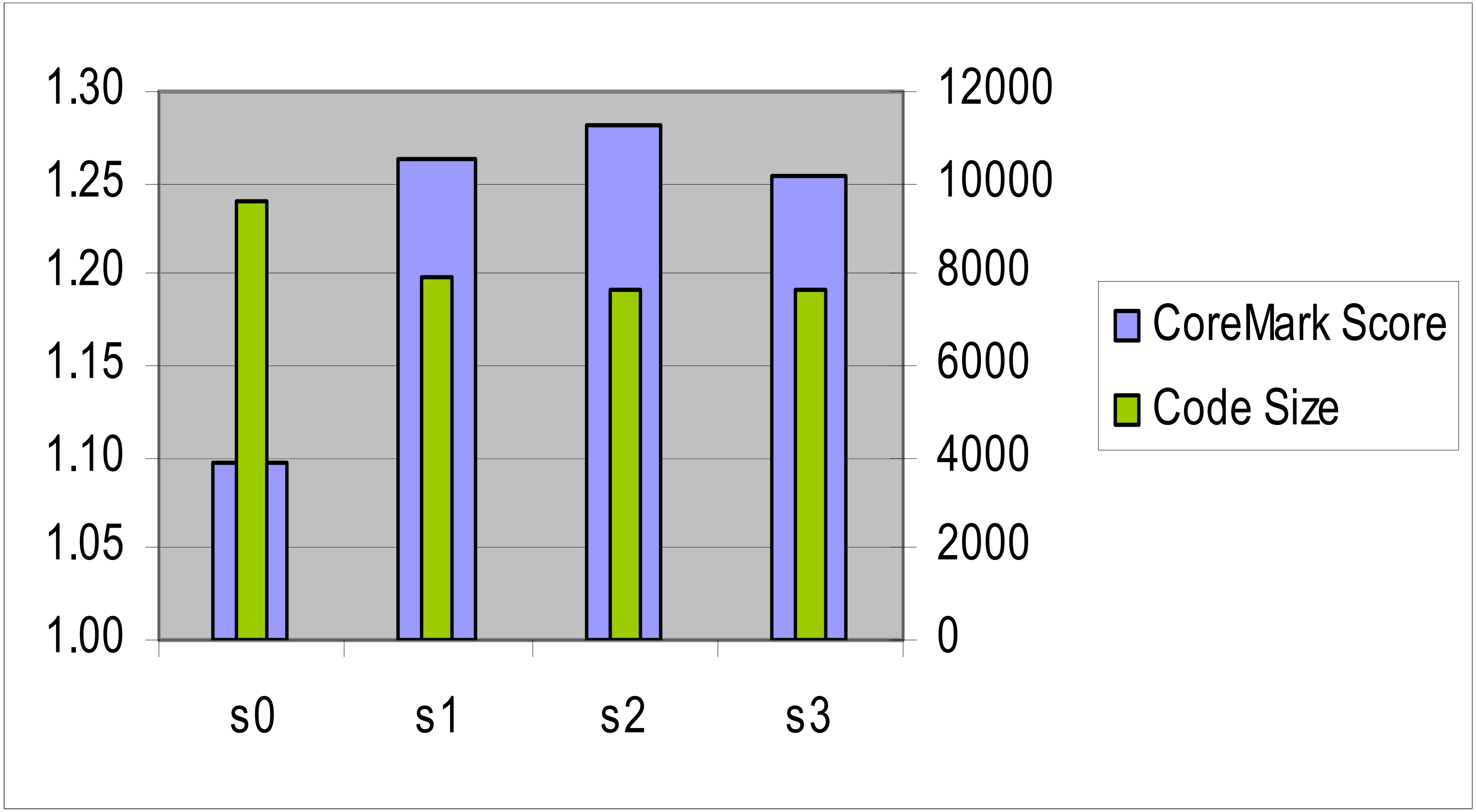
Peripheral code

Part	Init Code (Bytes)	Data rx code (Bytes)
AVR8 ATmega644	28	32
MSP430	50	28
M0 LPC11xx	68	30

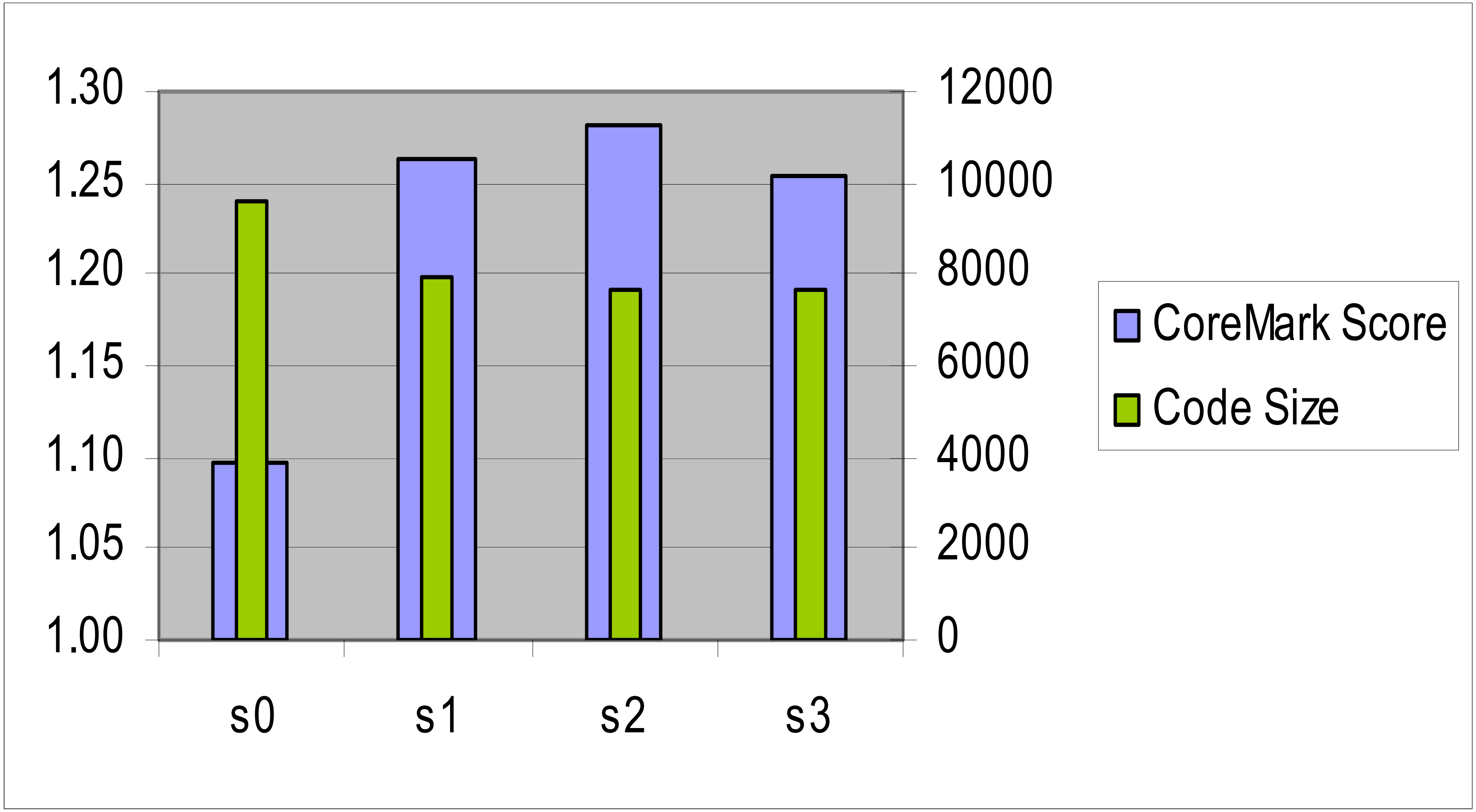
Speed Optimization effects



Size Optimization effects



Size Optimization effects



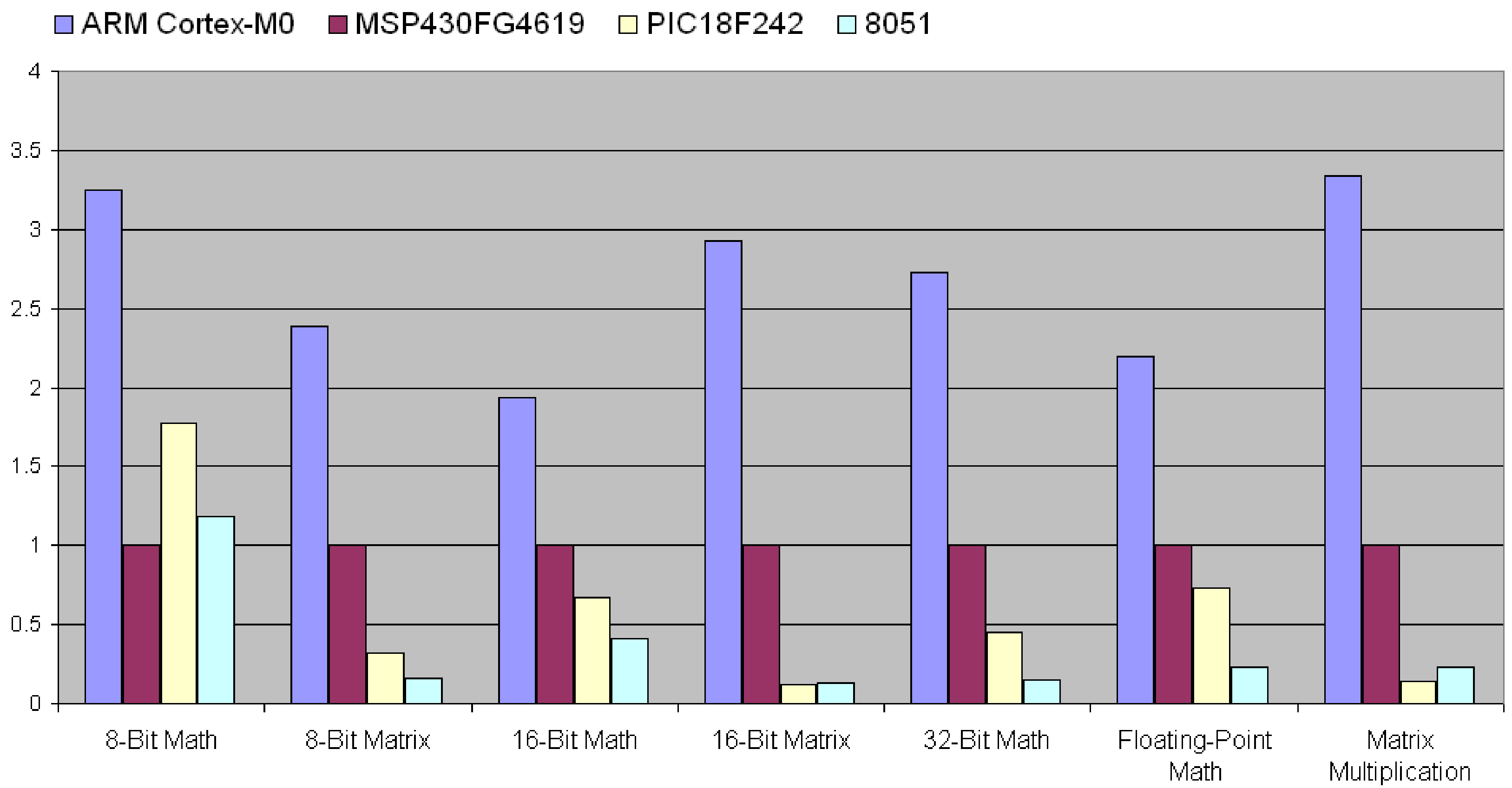
What About Libraries

- 33% reduction using optimized Libs

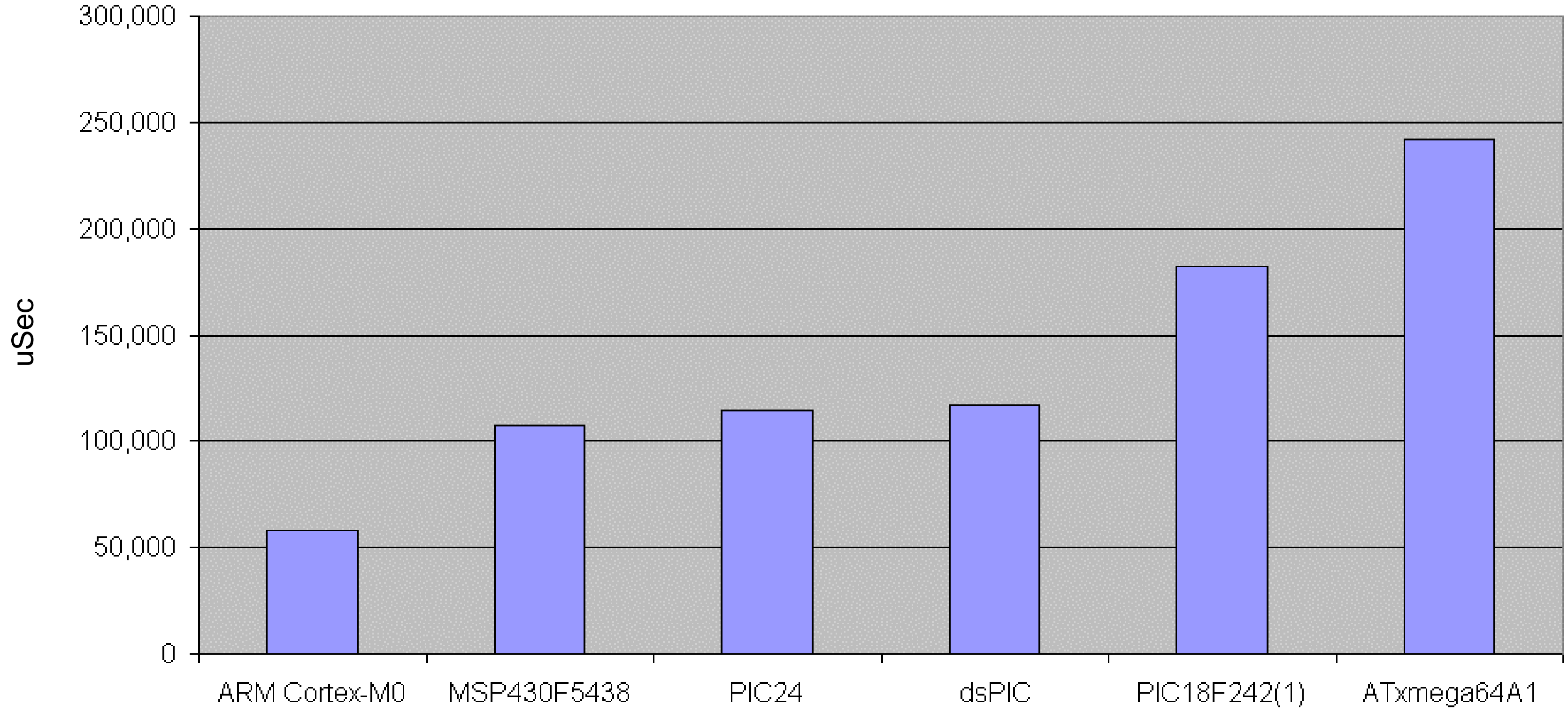
	NXP M0					
	MicroLib			Standard Lib		
Auto BM	**Compile**	**Lib**	**Total**	**Compile**	**Lib**	**Total**
a2time	4032	4552	8584	4084	9364	13448
aifftr	4636	6712	11348	4708	12668	17376
aifirf	3300	4500	7800	3356	8388	11744
aiifft	4348	6636	10984	4402	12284	16686
basefp	3348	4668	8016	3404	10460	13864
bitmnp	4776	4412	9188	4828	8328	13156
canrdr	3272	4412	7684	3328	8328	11656
idctrn	4564	6884	11448	4616	13012	17628
iirflt	4552	4540	9092	4608	8388	12996
matrix	6632	4872	11504	6684	10716	17400
pntrch	3204	4512	7716	3260	8412	11672
puwmod	3436	4500	7936	3492	8388	11880
rspeed	2728	4540	7268	2780	8328	11108
tblook	3612	4864	8476	3668	10728	14396
ttsprk	5060	4540	9600	5116	8388	13504
average (8)	3663	4496	8159	3717	8491	12208

Performance

Computation Performance

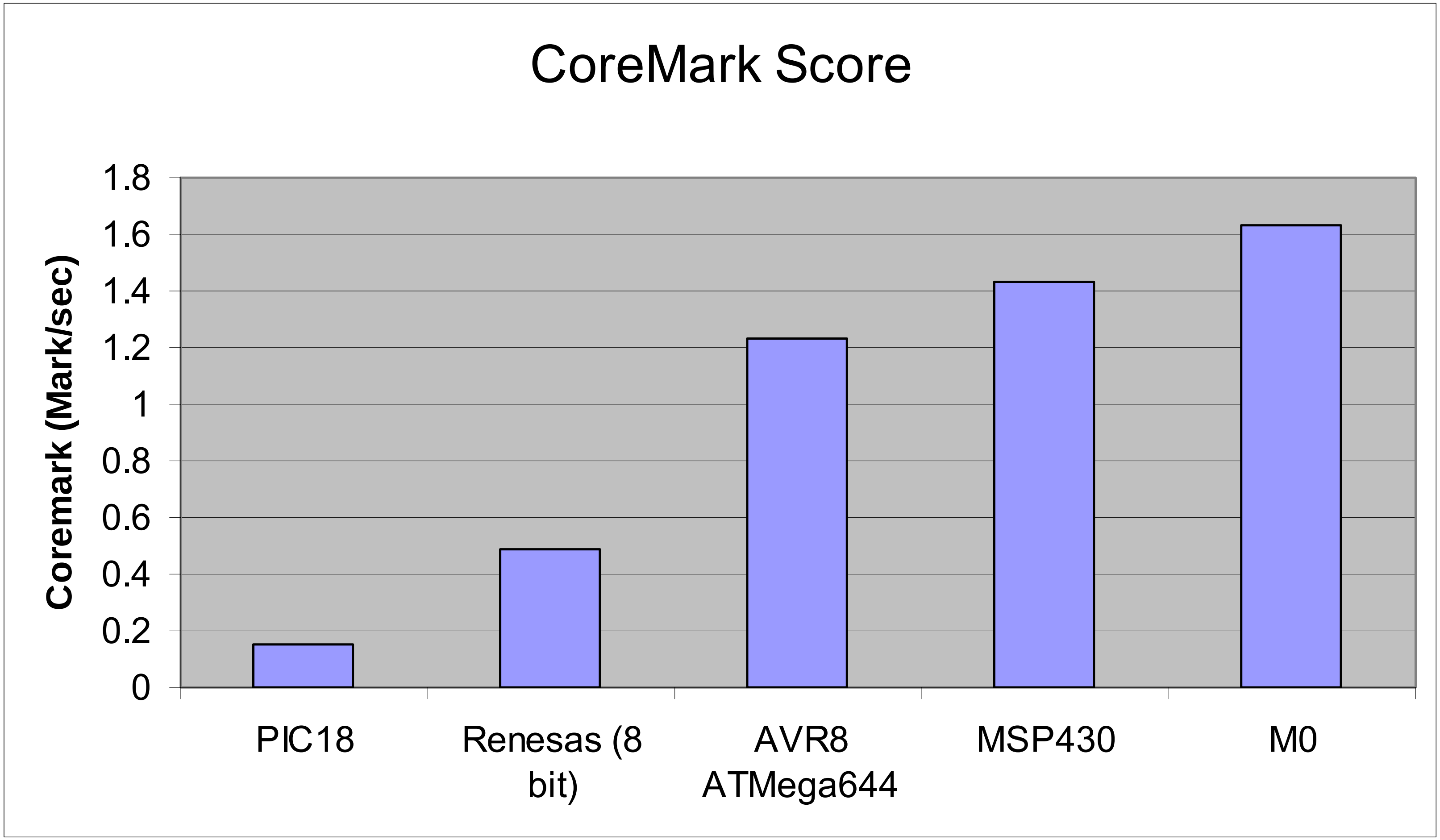


Computation Performance



16 bit FIIR filter performance at 1MHz

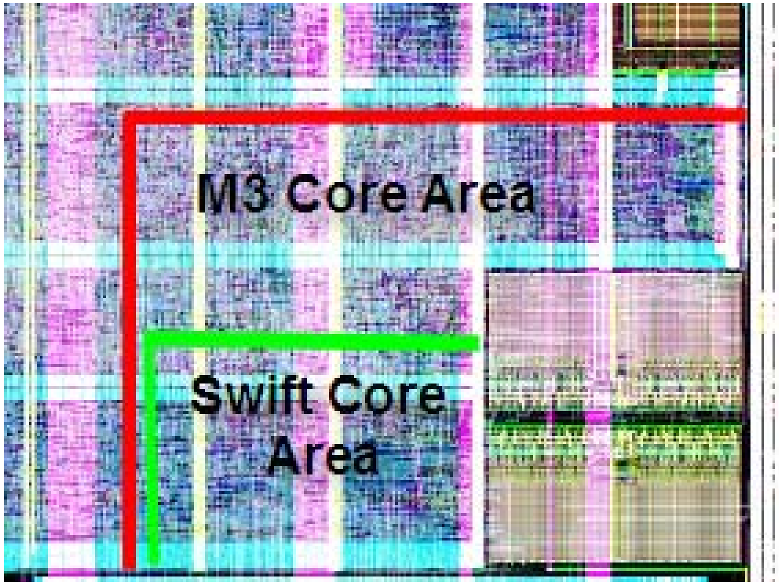
Computation Performance



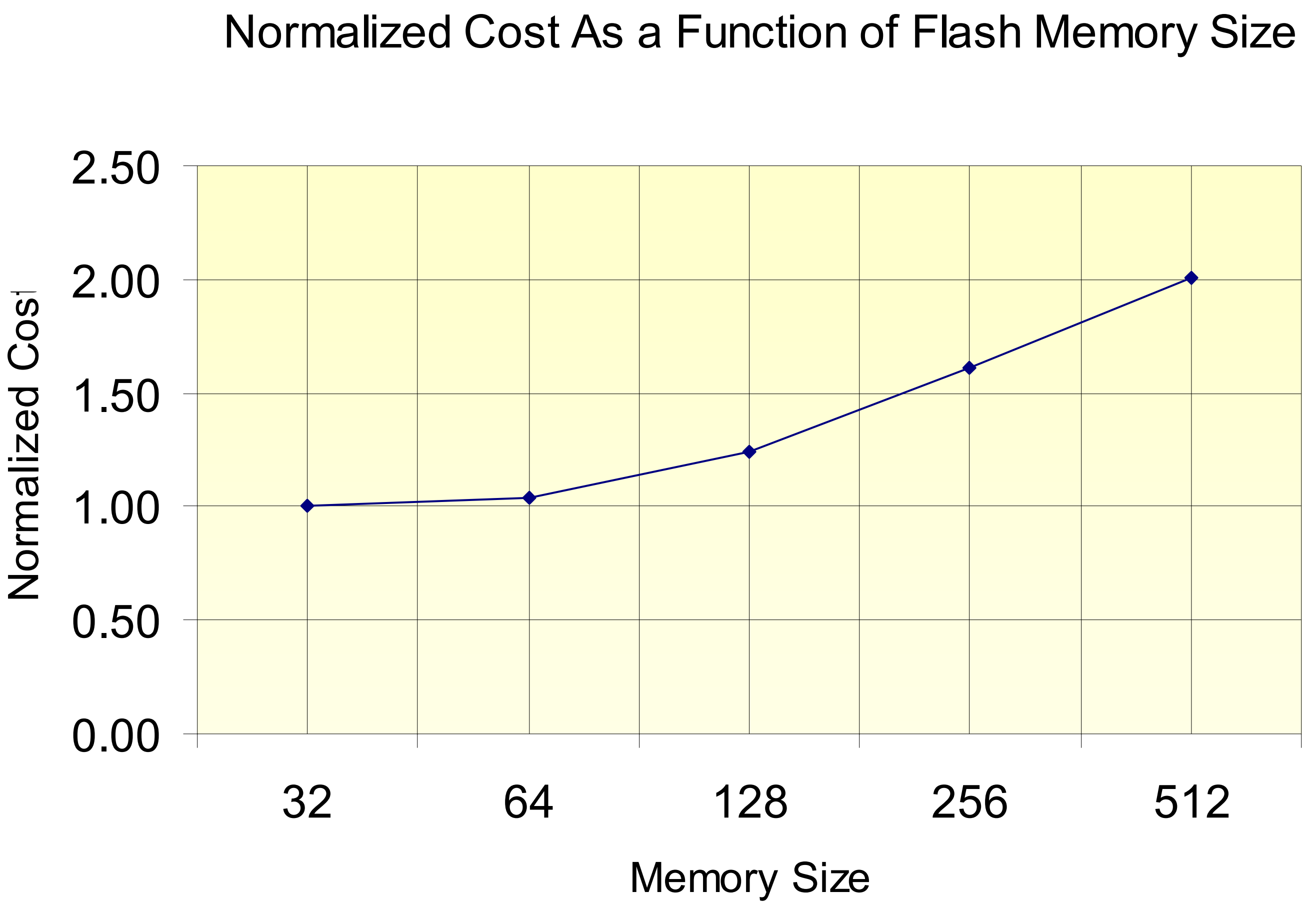
Cost

Does the core size matter?

- The M0 core is the smallest cortex core
- About 1/3 of the M3 for similar configuration
- Similar size to 8 bit cores



Core Size Matters



Tools

MCU Tool Solutions

LPCXPRESSO — *NXP's Low cost Development Tool Chain*

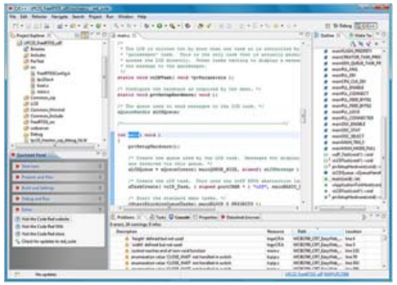
mbed — *Rapid Prototyping Online Tool*

IAR SYSTEMS, hitex, KEIL An ARM® Company — *Traditional Feature Rich Tools (third party)*

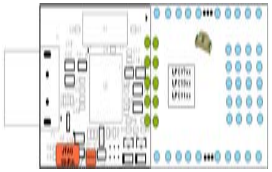
NXP's FIRST Low Cost Toolchain

LPC XPRESSO

Eclipse-based IDE



LPCXpresso Starter Board



Evaluation *Product Development*

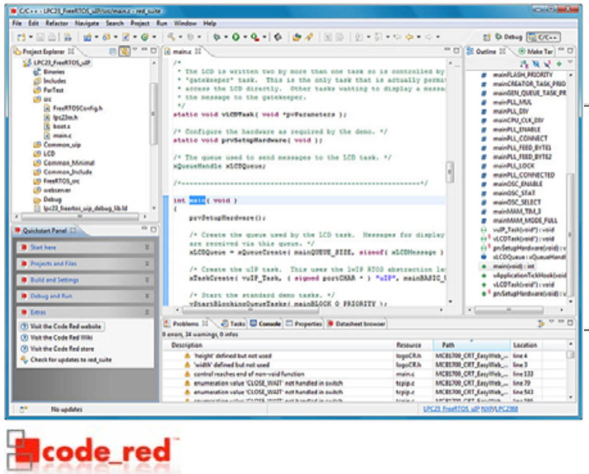
LPCXpresso

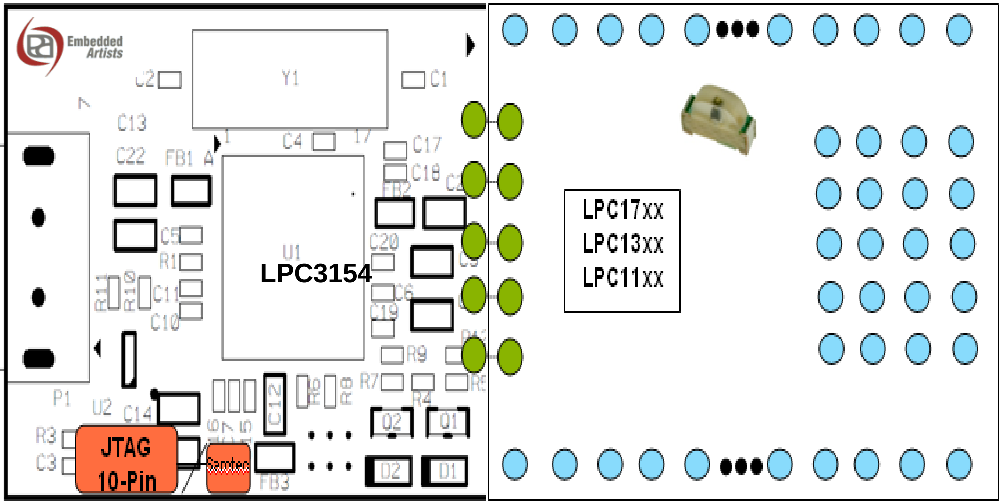
- LPCXpresso will provide end-to-end solution from evaluation all the way to product development
- Attractive upgrade options to full blown suites and development boards
- *LPCXpresso will change the perception about NXP's solution for tools*
- Key competition:
  - Microchip MPLAB
  - Atmel AVR Studio

"LPCXpresso will change the Tool Landscape for NXP"

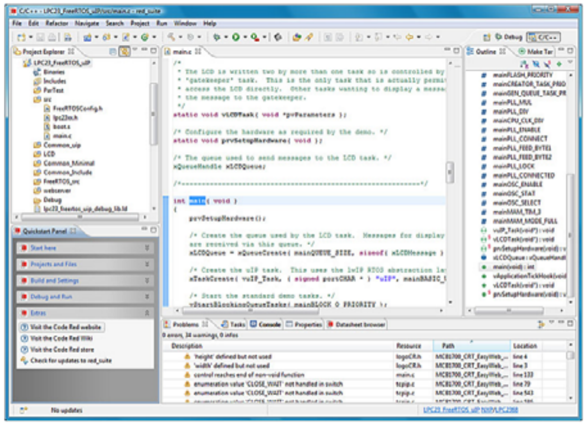
LPCXpresso Components

- NXP has created the first single perspective Eclipse IDE
- This offers the power and flexibility of Eclipse in combination with a simple and easy to learn user interface
- Supports all NXP products (currently up to 128k)
- LPC3154 HS USB download and debug engine
- LPC134x Target board



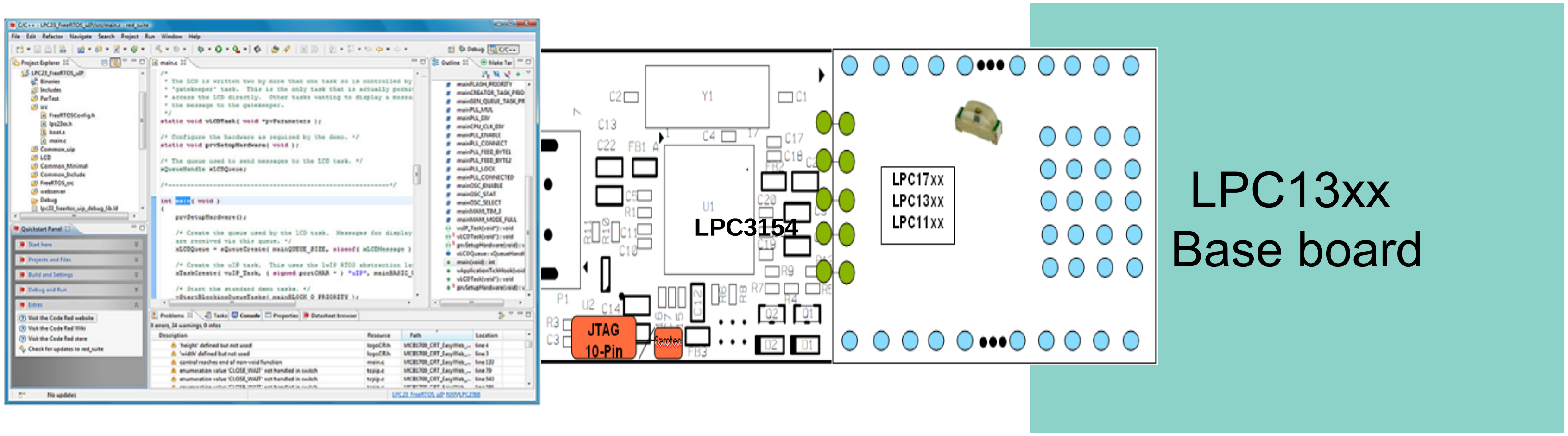


Evaluation



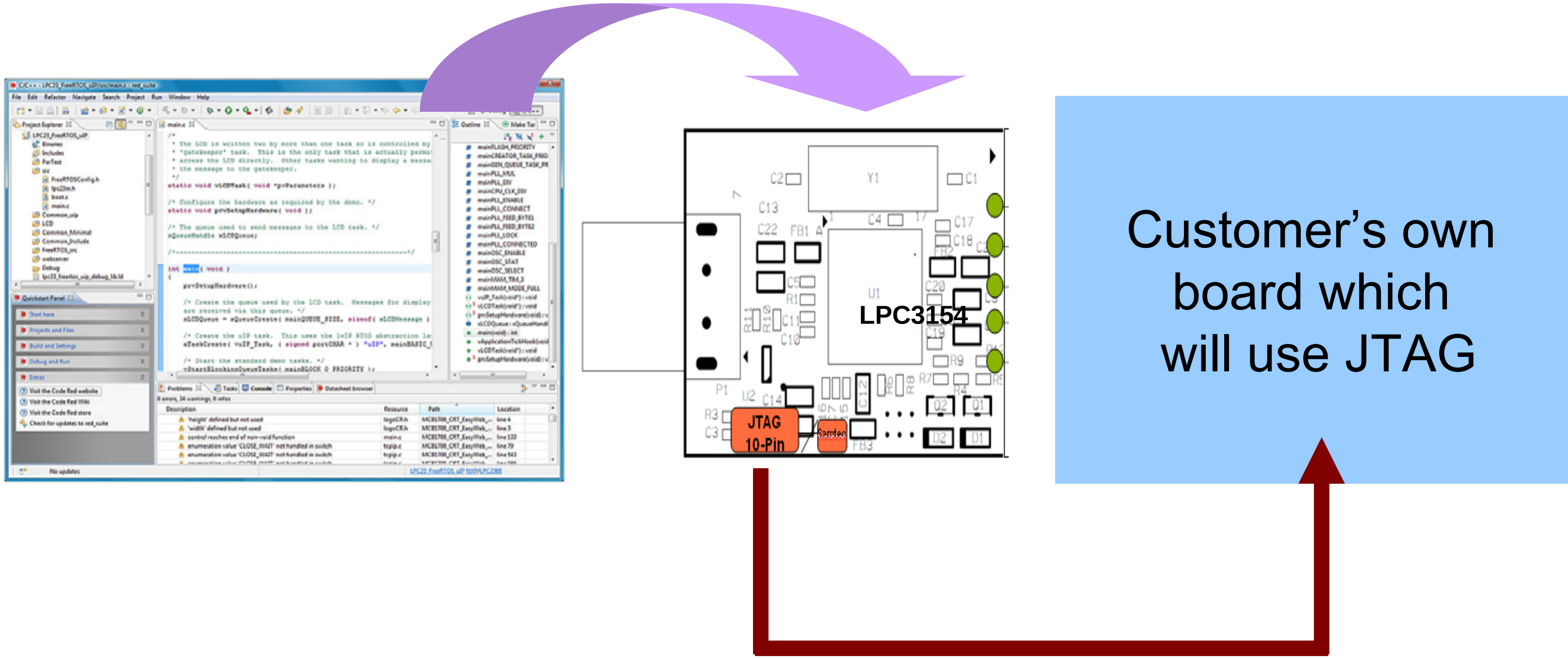
- The target board is very simple with one LED and a layout option for USB
- Traces between the two boards can be cut, to allow SWD connection to any customer target. (Eval target can be reconnected by jumpers)

Exploration



- Customers can upgrade to full version of Red Suite (Discount coupon)
- Customers can buy an add-on EA base board that connects a wide range of resources to the I/O and peripherals of the LPC13xx. Customers can also upgrade to other EA boards (Discount coupon)

Development

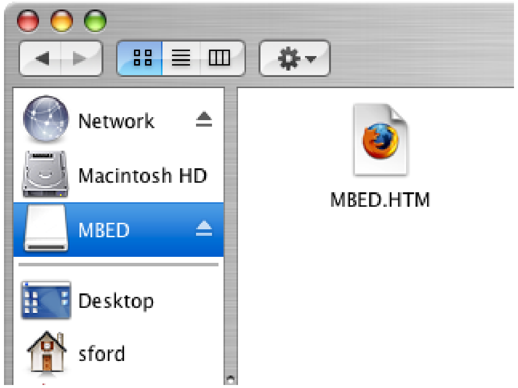


- Traces can be cut and the LPC13xx target board will out of the picture
- Customers can then use the JTAG connection to download code into their own application board using the same existing IDE and JTAG connector
- Note: Customers can directly jump to this stage and use LPCXpresso for their complete application development without ever having to upgrade

mbed LPC1768 Value Proposition

- New users start creating applications in 60 seconds
- Rapid Prototyping with LPC1700 series MCUs
  - Immediate connectivity to peripherals and modules for prototyping LPC1700-based system designs
  - Providing developers with the freedom to be more innovative & productive
- mbed C/C++ Libraries provide API-driven approach to coding
  - High-level interfaces to peripherals enables rock-solid, compact code
  - Built on Cortex Microcontroller Software Interface Standard (CMSIS)
- Download compiled binary by saving to the mbed hardware
  - Just like saving to a USB Flash Drive
- Tools are online - there is nothing to configure, install or update, and everything works on Windows, Mac or Linux
- Hardware in a 40-pin 0.1" pitch DIP form-factor
  - Ideal for solderless breadboard, stripboard and through-hole PCBs

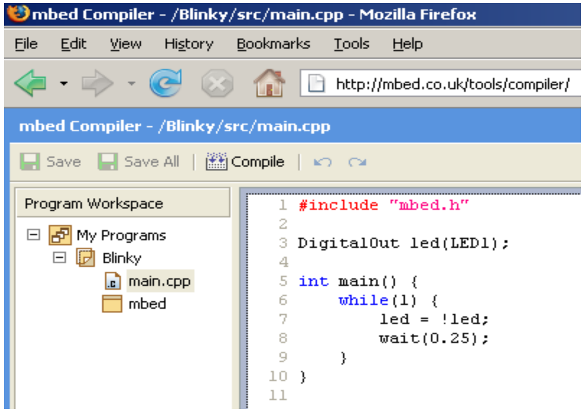
First Experience – Hassle-Free Evaluation



Remove board from the box

Plug it in…

Up pops a USB Disk linking to website



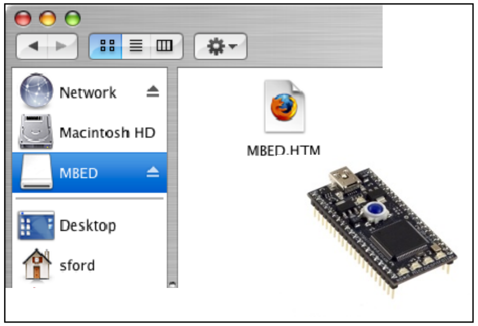
Compile a program online

Save to the board and you're up and running

No Installation!

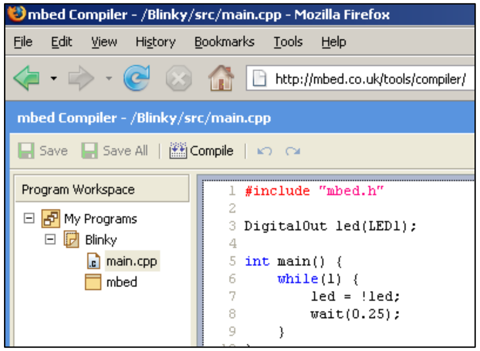
"Hello World!" in 60 seconds

mbed Technology



USB Drag ‘n’ Drop Programming Interface

- Nothing to Install: Program by saving binaries
- Works on Windows, Linux, Mac, without drivers
- Links through to mbed.org website



Online Compiler

- Nothing to Install: Browser-based IDE
- Best in class RealView Compiler in the back end
- No code size or production limitations

```
#include “mbed.h”
Serial terminal(9,10);
AnalogIn temp(19);
int main() {
 if(temp > 0.8)
   terminal.printf(“Hot!”);
}
```

High-level Peripheral Abstraction Libraries

- Instantly understandable APIs
- Object-oriented hardware/software abstraction
- Enables experimentation without knowing MCU details

Example Beta Projects - Videos

- Rocket Launch
  - http://www.youtube.com/watch?v=zyY451Rb-50&feature=PlayList&p=000FD2855BEA7E90&index=11
- Billy Bass
  - http://www.youtube.com/watch?v=Y6kECR7T4LY
- Voltmeter
  - http://www.youtube.com/watch?v=y_7WxhdLLVU&feature=PlayList&p=000FD2855BEA7E90&index=8
- Knight Rider
  - http://www.youtube.com/watch?v=tmfkLJY-1hc&feature=PlayList&p=000FD2855BEA7E90&index=4
- Bluetooth Big Trak
  - http://www.youtube.com/watch?v=RhC9AbJ_bu8&feature=PlayList&p=000FD2855BEA7E90&index=3
- Scratch Pong
  - http://www.youtube.com/watch?v=aUtYRguMX9g&feature=PlayList&p=000FD2855BEA7E90&index=5

More information

- Available from NXP Distributors and eTools
- Boards cost $99

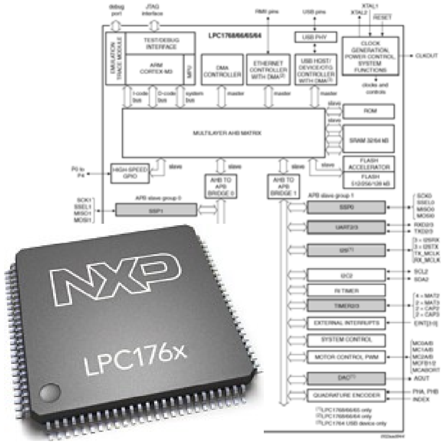
Learn More:

- http://www.standardics.nxp.com/support/development.hardware/mbed.lpc176x/
- http://mbed.org
- Featured Articles:
  - Circuit Cellar
  - Elektor

mbed

Rapid Prototyping for Microcontrollers

What's happening in Microcontrollers?



- Microcontrollers are getting **<u>cheap</u>**
  - 32-bit ARM Cortex-M3 Microcontrollers @ $1
- Microcontrollers are getting **<u>powerful</u>**
  - Lots of processing, memory, I/O in one package
- Microcontrollers are getting **<u>interactive</u>**
  - Internet connectivity, new sensors and actuators
- **Creates new opportunities for microcontrollers**

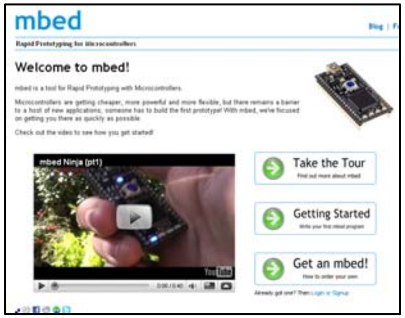
Rapid Prototyping

- **Rapid Prototyping helps industries create new products**
  - Control, communication and interaction increasingly define products
  - Development cycles for microelectronics have not kept pace

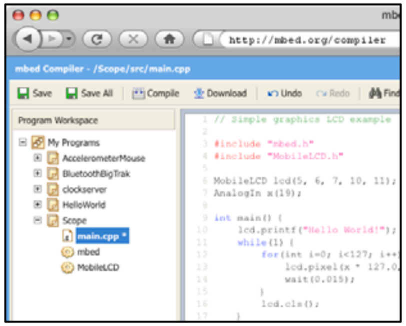
3D Moulding | *3D Printing* | *2D/3D Design* | *Web Frameworks*

mbed

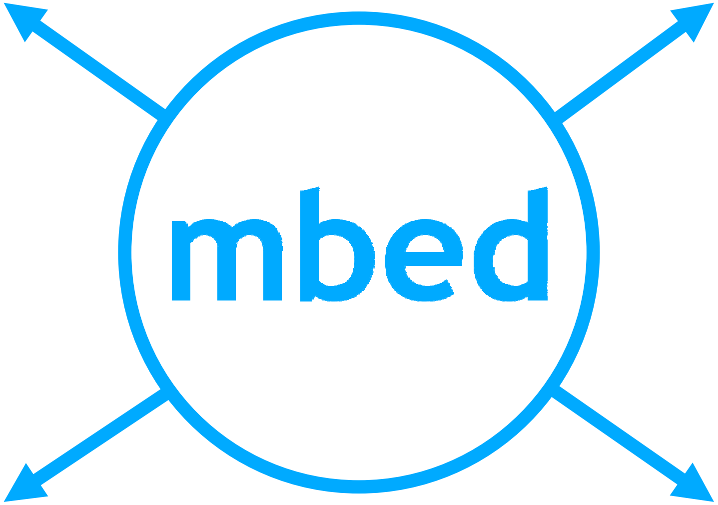
- **Getting Started and Rapid Prototyping with ARM MCUs**
  - Complete Targeted Hardware, Software and Web 2.0 Platform



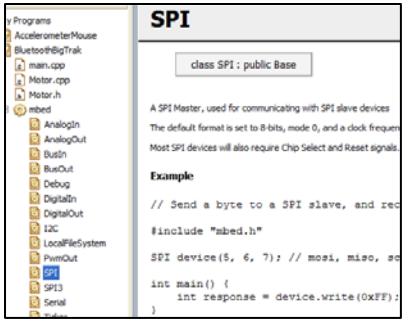
Dedicated Developer Web Platform



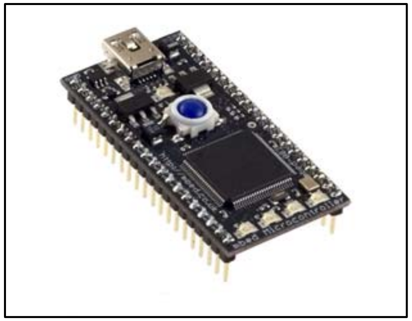
Lightweight Online Compiler



Rapid Prototyping for Microcontrollers



High-level Peripheral APIs



LPC Cortex-M MCU in a Prototyping Form-Factor

mbed Audience

mbed's focus on Rapid Prototyping has a broad appeal

- Designers new to embedded applications
  - **Enables new designs where electronics is not the focus**
- Experienced embedded engineers
  - **Enables fast proof-of-concepts to reduce risk and push boundaries**
- Marketing, distributors and application engineers
  - **A consistent platform enables effective and efficient demonstration, support and evaluation of MCUs**

Conclusion

- LPC1100 Family Based on the Cortex-M0 core
  - There are many users of 8 and 16 bit microcontrollers that are reluctant to use 32 bit architectures citing either overkill or complexity.
  - The M0 is an architecture that makes this argument irrelevant.
  - The LPC ARM Cortex-M0 family provides a microcontroller that is very low power, has better real-time performance than microcontrollers of lower bit width and provides a bridge to the full spectrum of the LPC families.