Lesson 1
Introduction

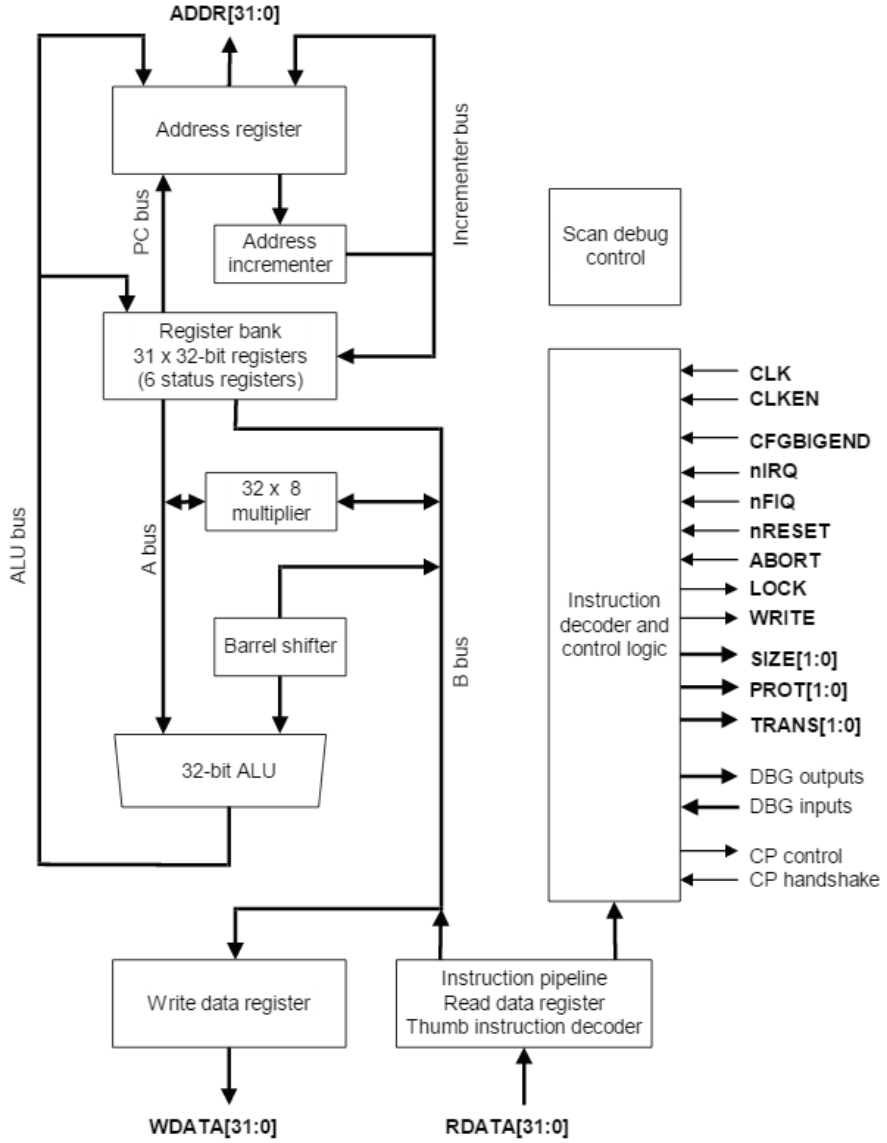
Lecturer: Harvard Tseng

ARM is a Company

ARM Holding plc (ARM)	
Founded	November 27, 1990
Headquarters	Cambridge, England, United Kingdom
Revenue	£968.3 million (2015)
Net income	£339.7 million (2015)
Website	arm.com

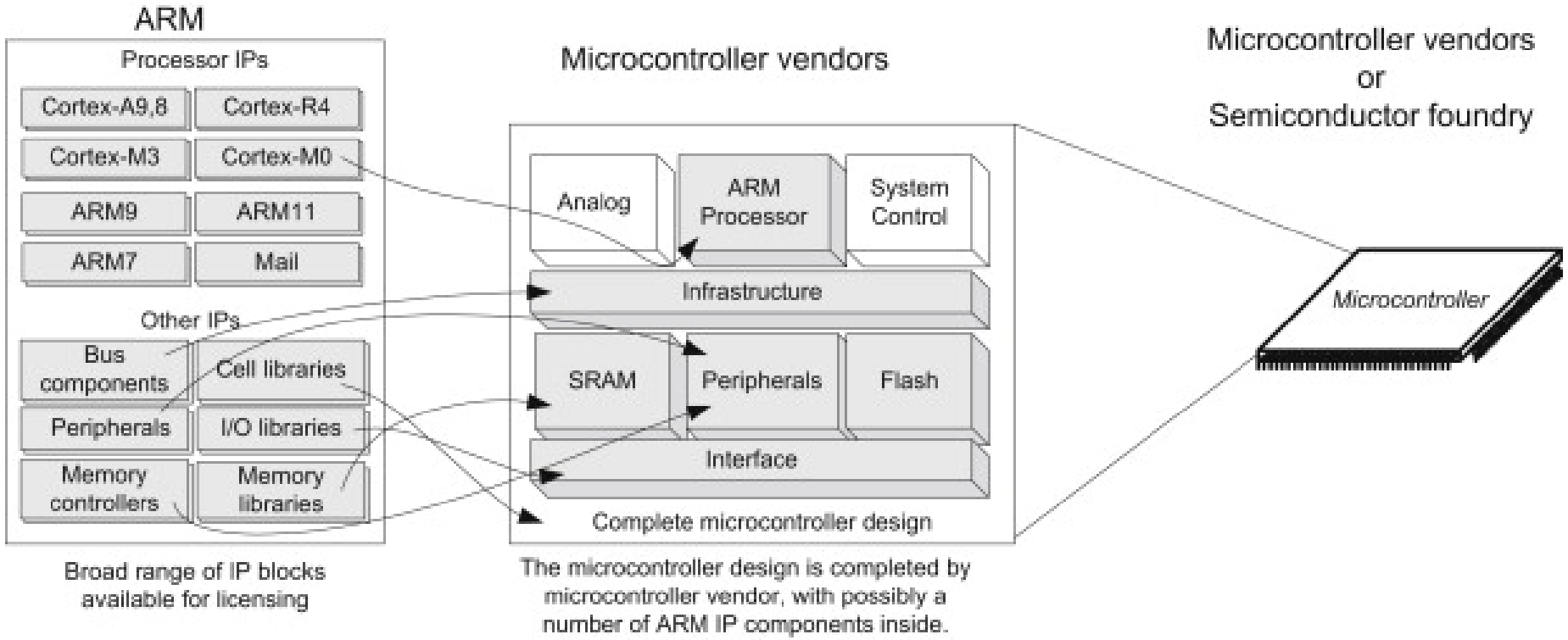
ARM is an Architecture of CPU

- Advanced RISC Machine



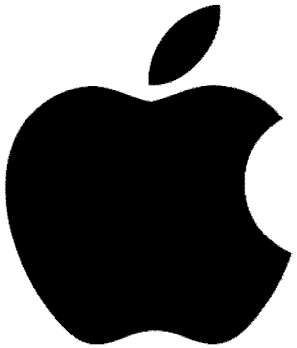
How ARM Earns Money?

- By intellectual property (IP) licensing.



Silicon Vendors

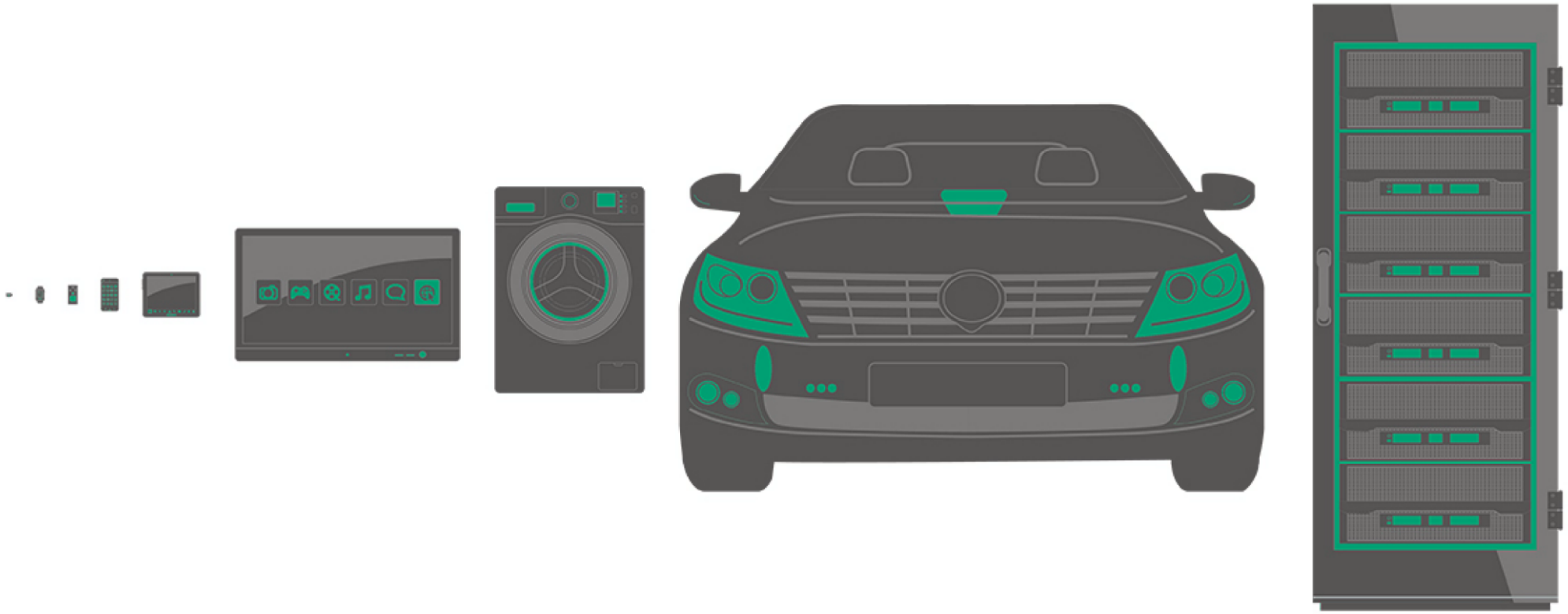
- Apple Inc.
- MediaTek Inc.
- Qualcomm Inc.
- …



Microcontroller Vendors

- STMicroelectronics (ST)
- Texas Instruments Inc. (TI)
- NXP Semiconductors N.V. (NXP)
- …

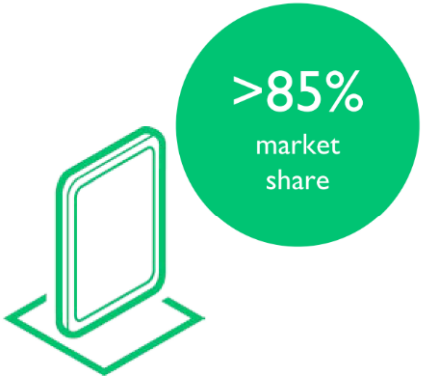
Sensors to smartphones to servers



ARM

Broad Market Penetration

smartphone

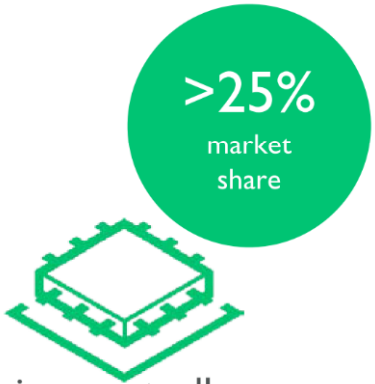


tablet

wearables

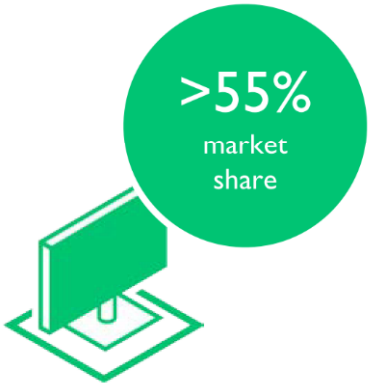
storage

automotive
infotainment



microcontrollers

wireless
connectivity

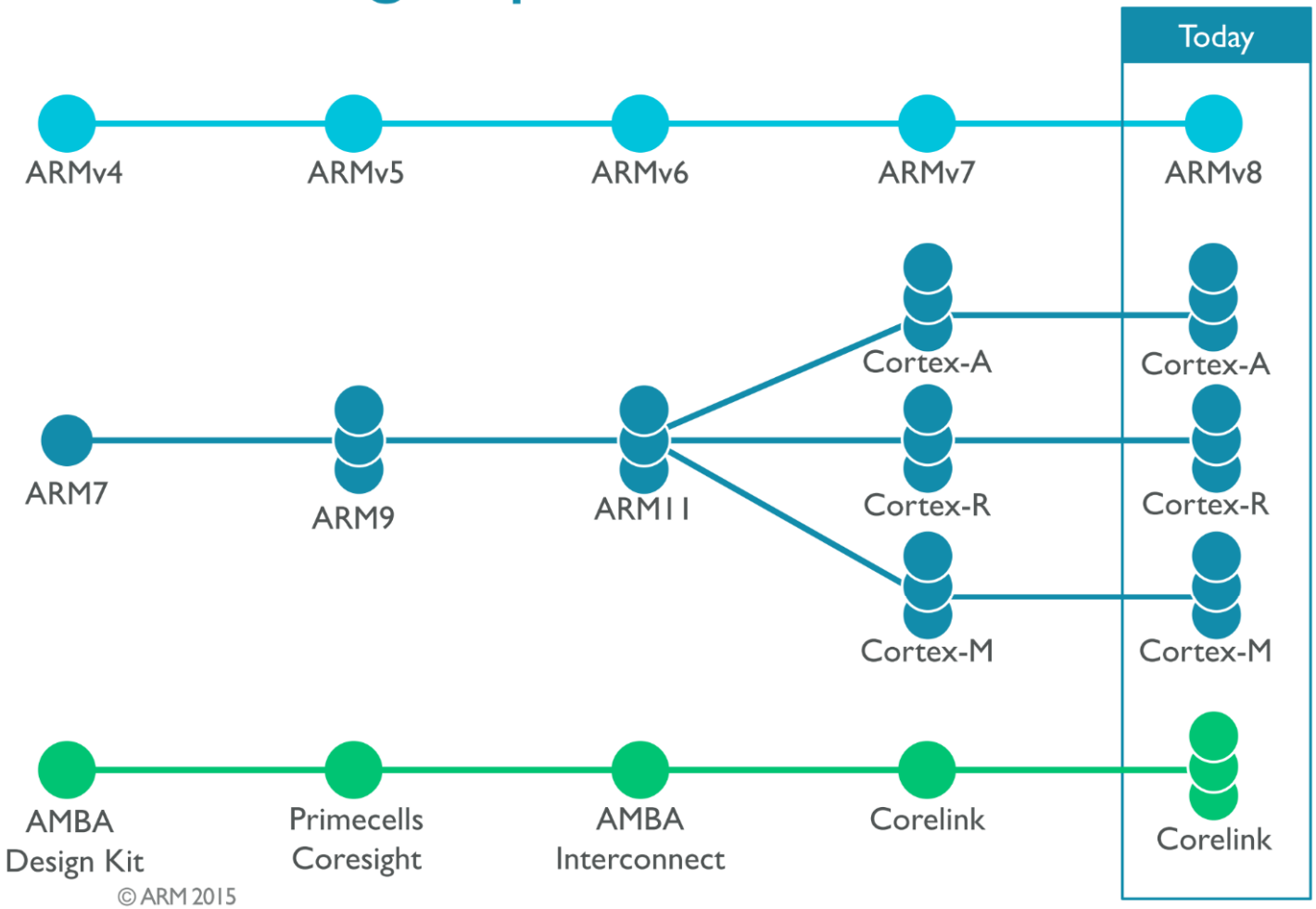


consumer
electronics

ARM

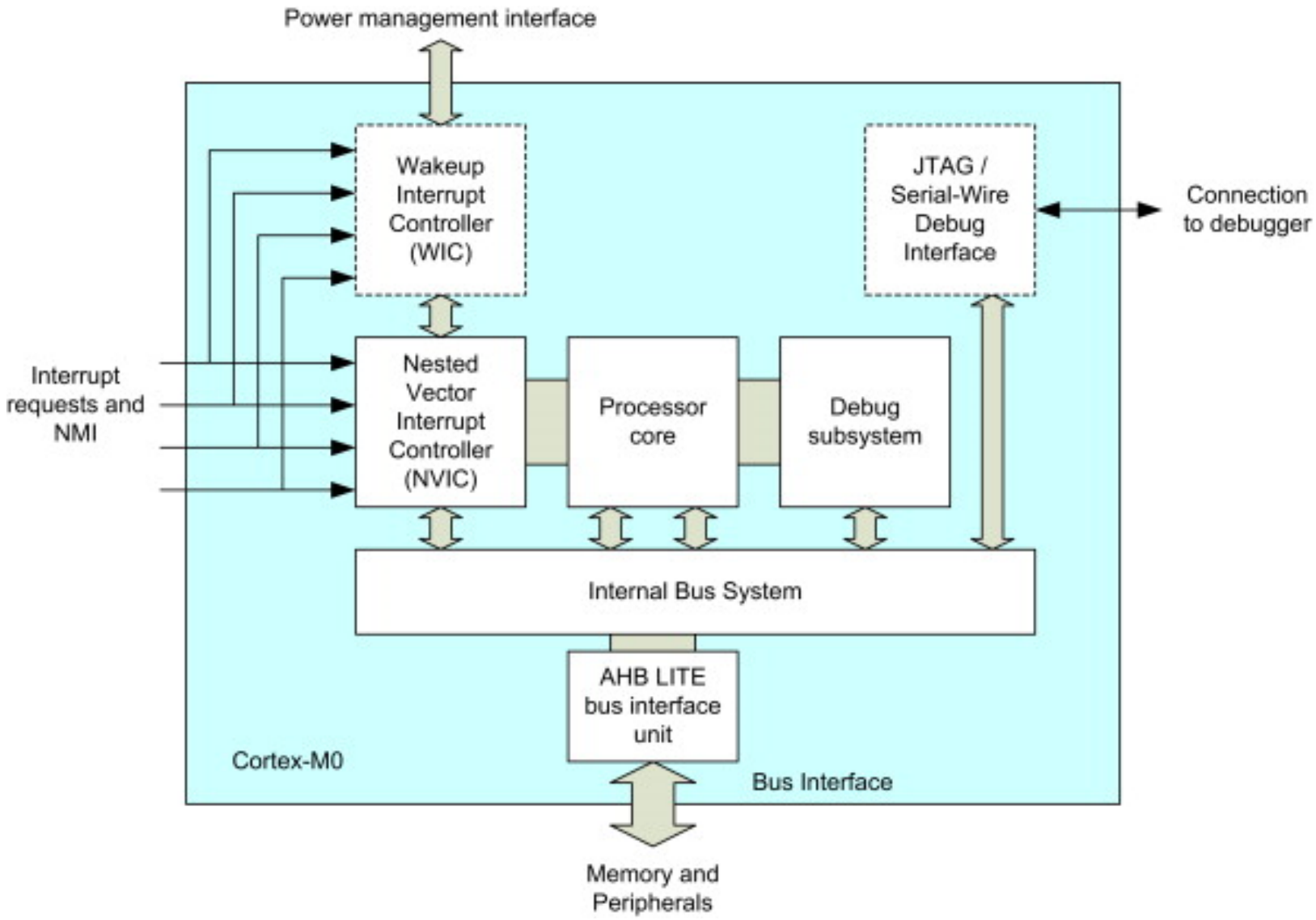
Cortex-M Series

Investing in processor innovation



ARM

Simplified Block Diagram

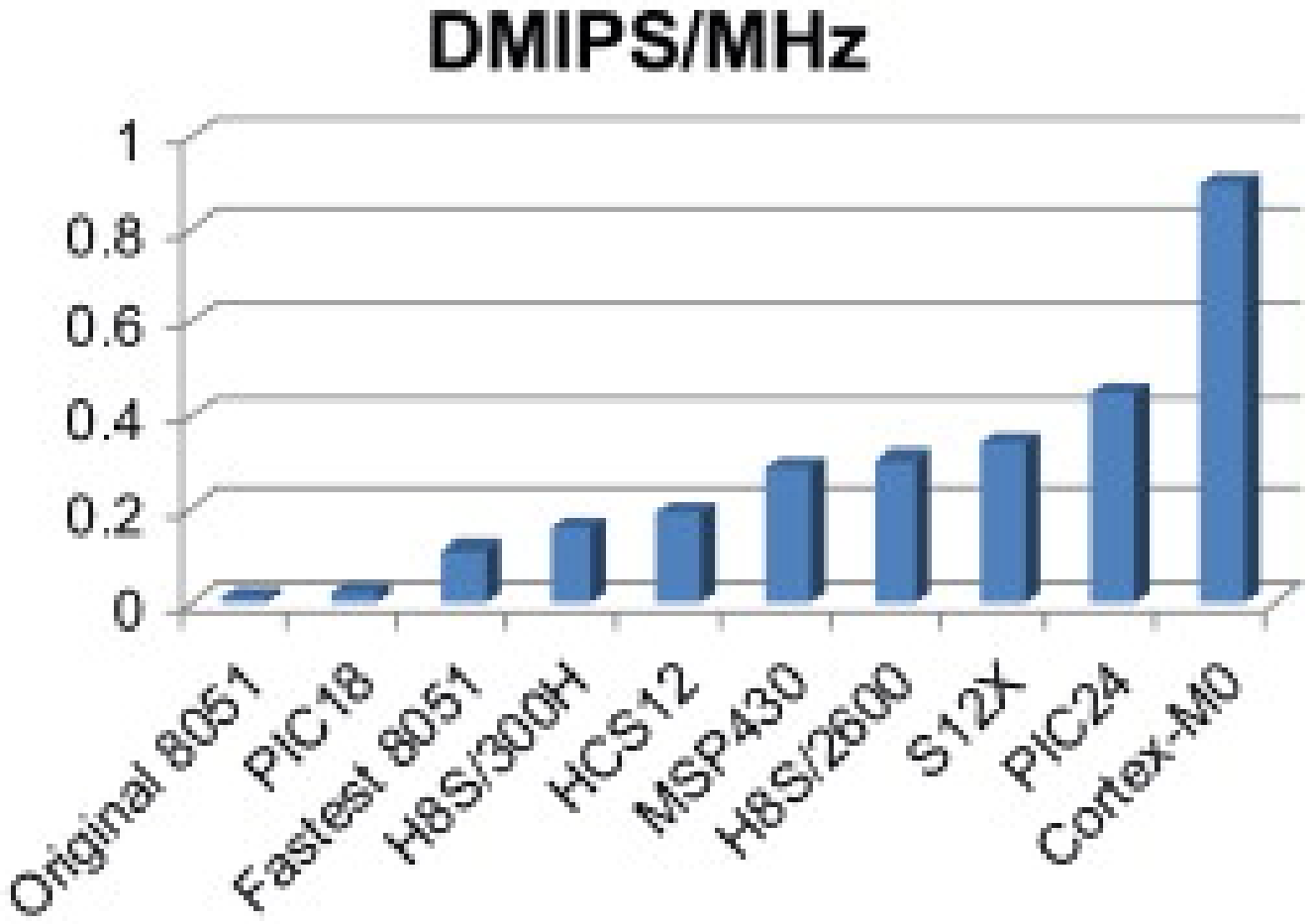


Advantages of the Cortex-M Processor

- Energy Efficiency
- Limitations in 8-Bit and 16-Bit Architectures
- Easy to Use, Software Portability
- Wide Range of Choices

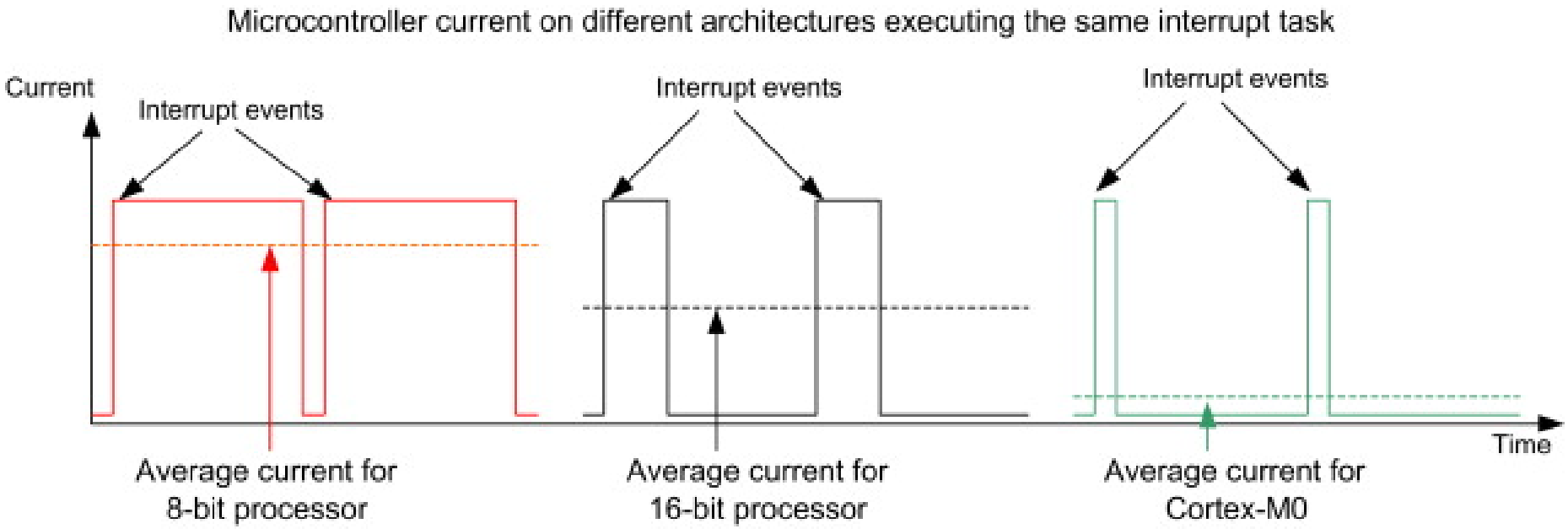
Why choose Cortex-M?

- High efficiency

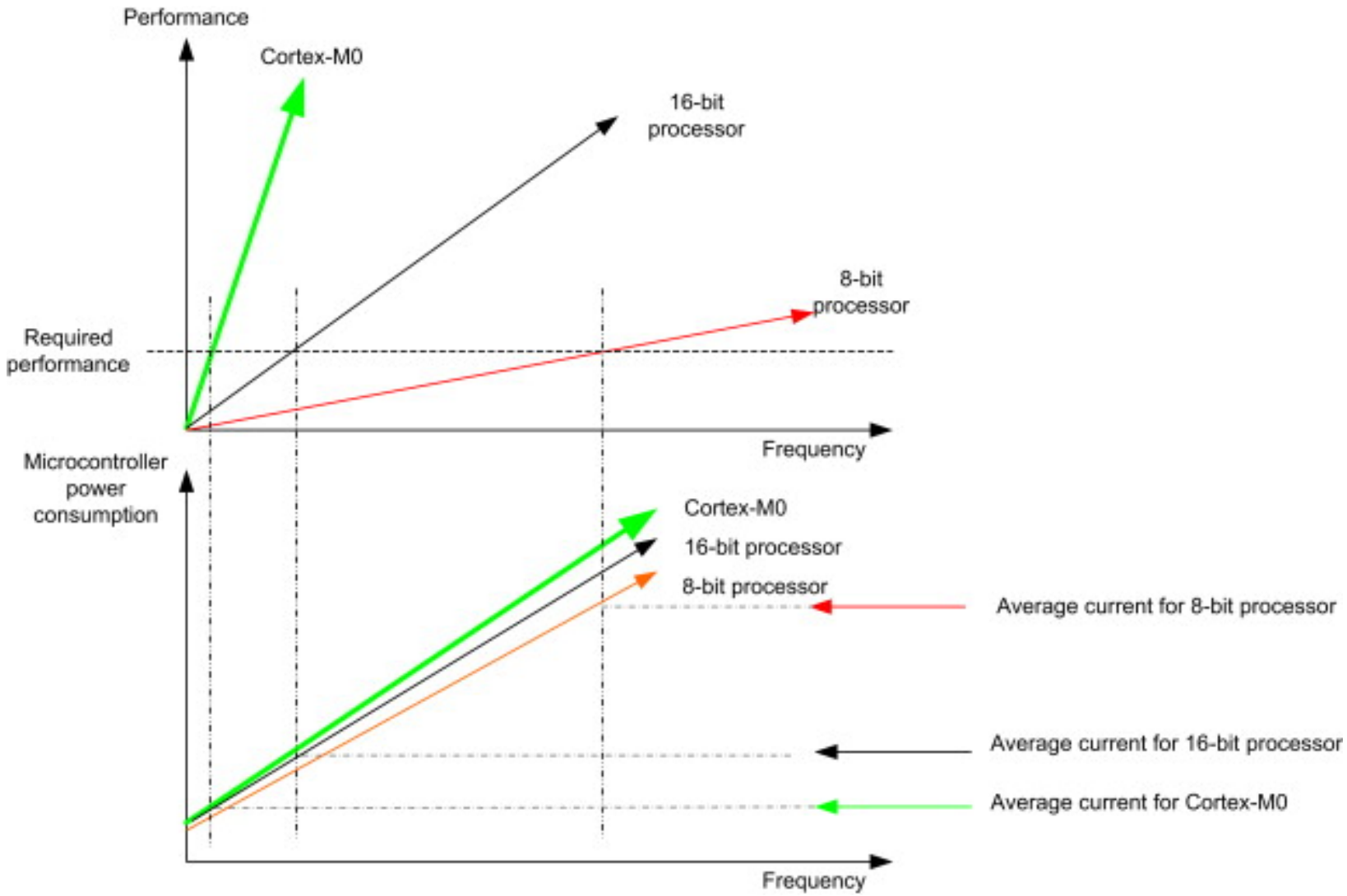


Why choose Cortex-M?

- Low-power features



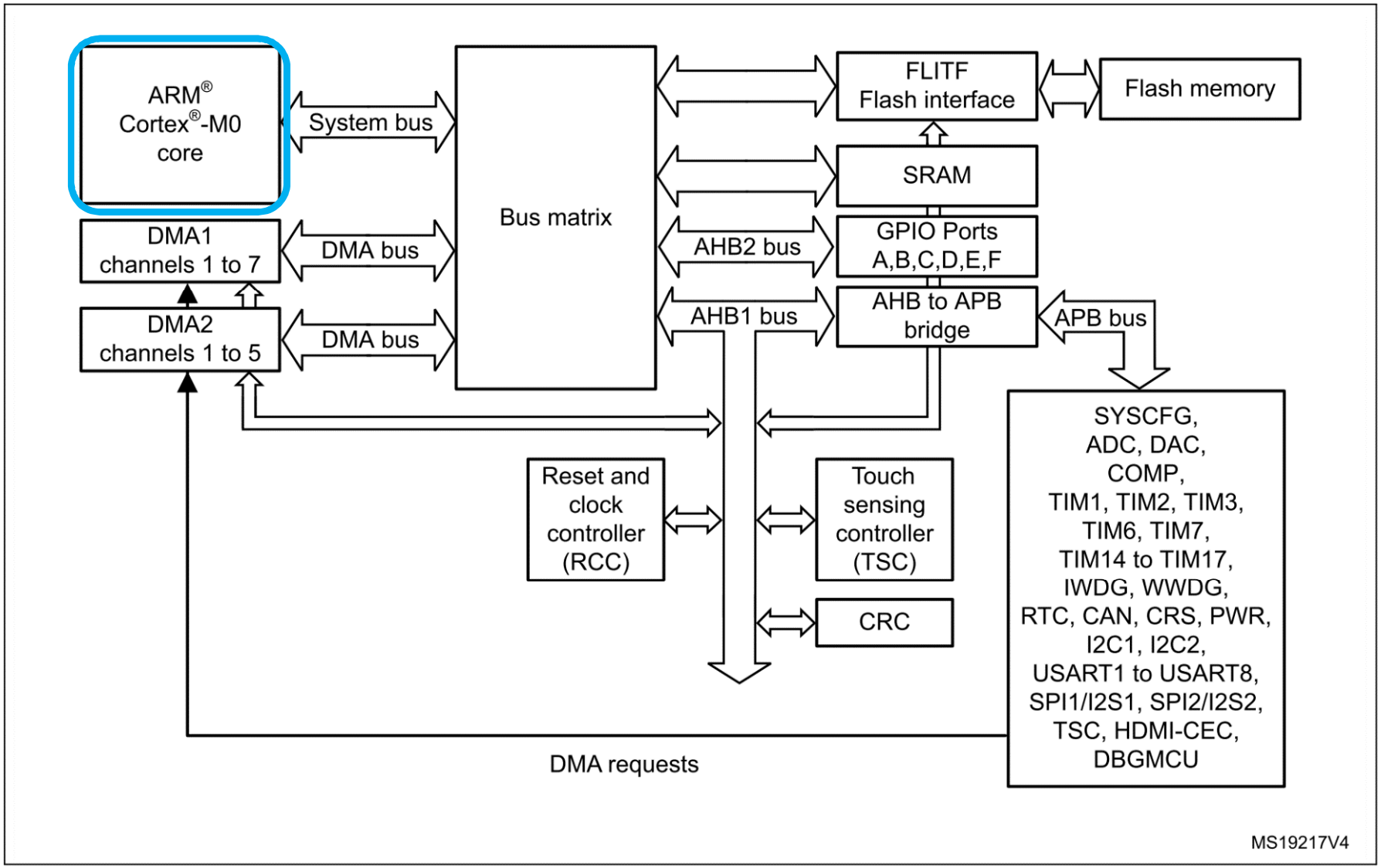
Why choose Cortex-M?

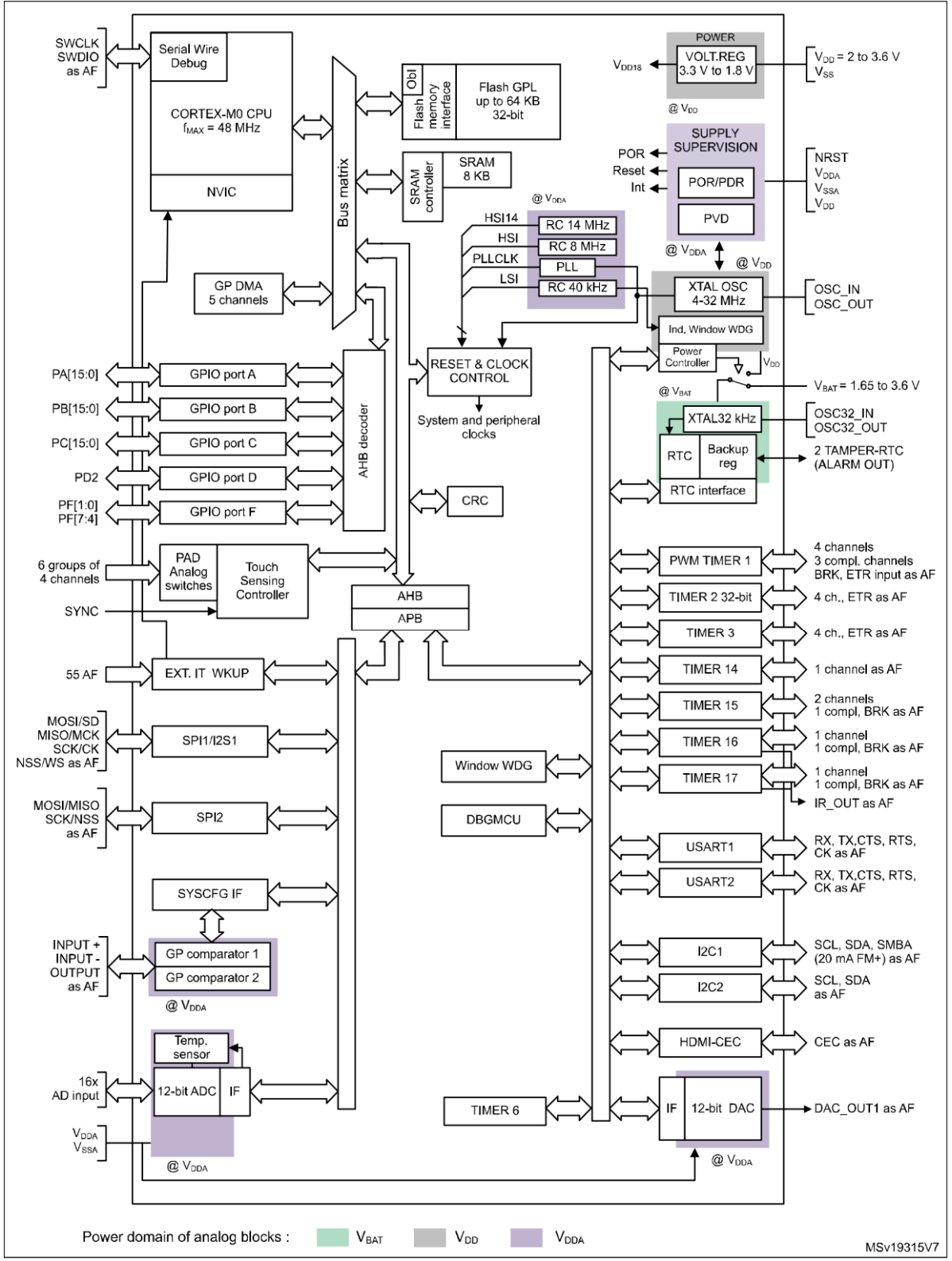


STM32F0x8

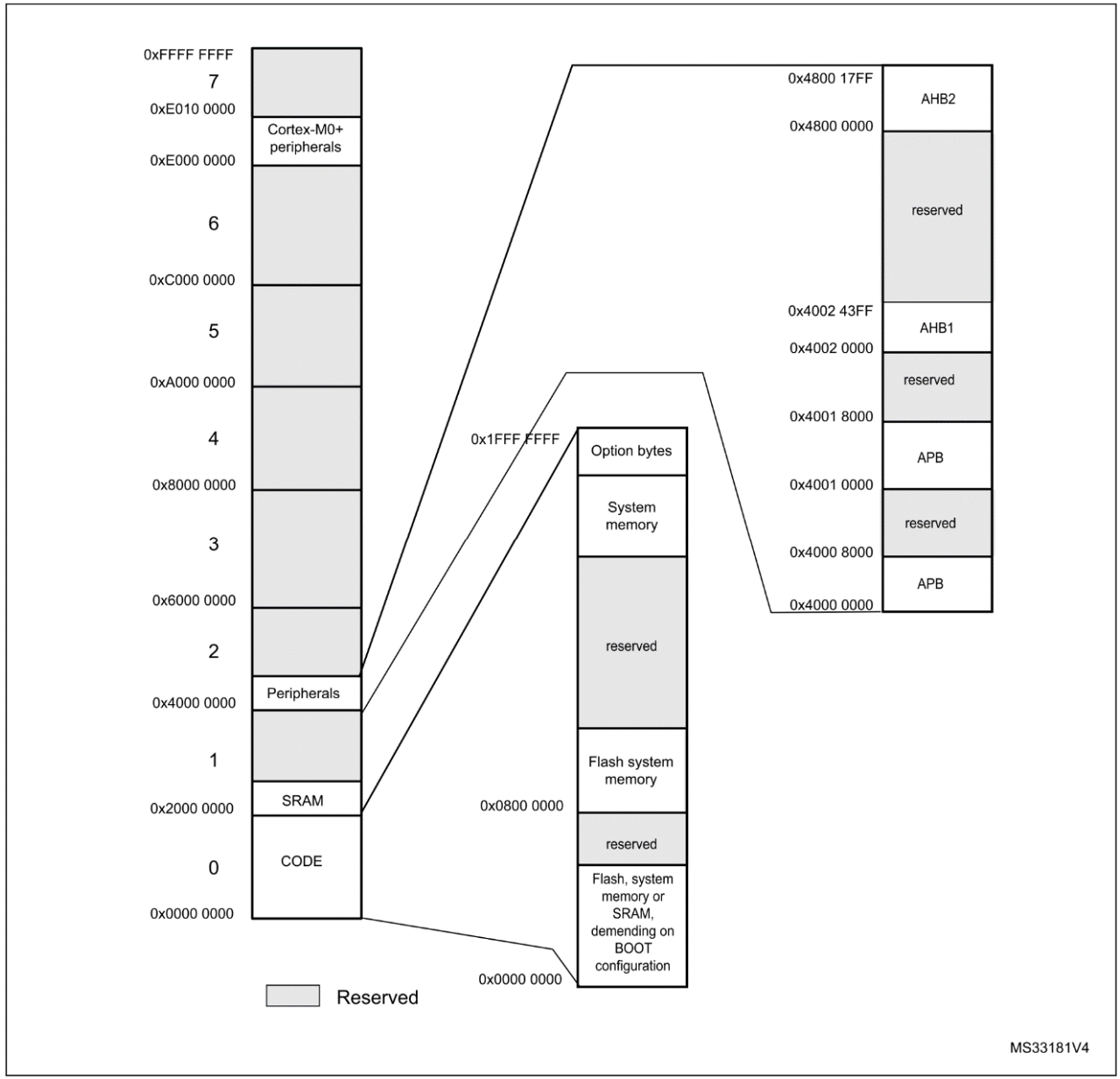
Advanced ARM®-based 32-bit MCUs

System Architecture




SWCLK
SWDIO
as AF
Serial Wire Debug
CORTEX-M0 CPU
f_{MAX} = 48 MHz
NVIC
Bus matrix
Obl
Flash memory interface
Flash GPL
up to 64 KB
32-bit
SRAM controller
SRAM
8 KB
POWER
V_{DD18}
VOLT.REG
3.3 V to 1.8 V
@ V_{DD}
V_{DD} = 2 to 3.6 V
V_{SS}
SUPPLY SUPERVISION
POR
Reset
Int
POR/PDR
PVD
@ V_{DDA}
@ V_{DD}
NRST
V_{DDA}
V_{SSA}
V_{DD}
@ V_{DDA}
HSI14
HSI
PLLCLK
LSI
RC 14 MHz
RC 8 MHz
PLL
RC 40 kHz
XTAL OSC
4-32 MHz
OSC_IN
OSC_OUT
Ind. Window WDG
Power Controller
V_{DD}
@ V_{BAT}
V_{BAT} = 1.65 to 3.6 V
XTAL32 kHz
OSC32_IN
OSC32_OUT
RTC
Backup reg
2 TAMPER-RTC
(ALARM OUT)
RTC interface
GP DMA
5 channels
RESET & CLOCK CONTROL
System and peripheral clocks
CRC
PA[15:0]
PB[15:0]
PC[15:0]
PD2
PF[1:0]
PF[7:4]
GPIO port A
GPIO port B
GPIO port C
GPIO port D
GPIO port F
AHB decoder
6 groups of 4 channels
SYNC
PAD Analog switches
Touch Sensing Controller
AHB
APB
PWM TIMER 1
4 channels
3 compl. channels
BRK, ETR input as AF
TIMER 2 32-bit
4 ch., ETR as AF
TIMER 3
4 ch., ETR as AF
TIMER 14
1 channel as AF
TIMER 15
2 channels
1 compl, BRK as AF
TIMER 16
1 channel
1 compl, BRK as AF
TIMER 17
1 channel
1 compl, BRK as AF
IR_OUT as AF
55 AF
EXT. IT WKUP
MOSI/SD
MISO/MCK
SCK/CK
NSS/WS as AF
SPI1/I2S1
Window WDG
MOSI/MISO
SCK/NSS
as AF
SPI2
DBGMCU
USART1
RX, TX,CTS, RTS, CK as AF
USART2
RX, TX,CTS, RTS, CK as AF
SYSCFG IF
INPUT +
INPUT -
OUTPUT
as AF
GP comparator 1
GP comparator 2
@ V_{DDA}
I2C1
SCL, SDA, SMBA
(20 mA FM+) as AF
I2C2
SCL, SDA
as AF
Temp. sensor
16x
AD input
12-bit ADC
IF
HDMI-CEC
CEC as AF
V_{DDA}
V_{SSA}
@ V_{DDA}
TIMER 6
IF
12-bit DAC
DAC_OUT1 as AF
@ V_{DDA}
Power domain of analog blocks :
V_{BAT}
V_{DD}
V_{DDA}
MSv19315V7

Memory Map

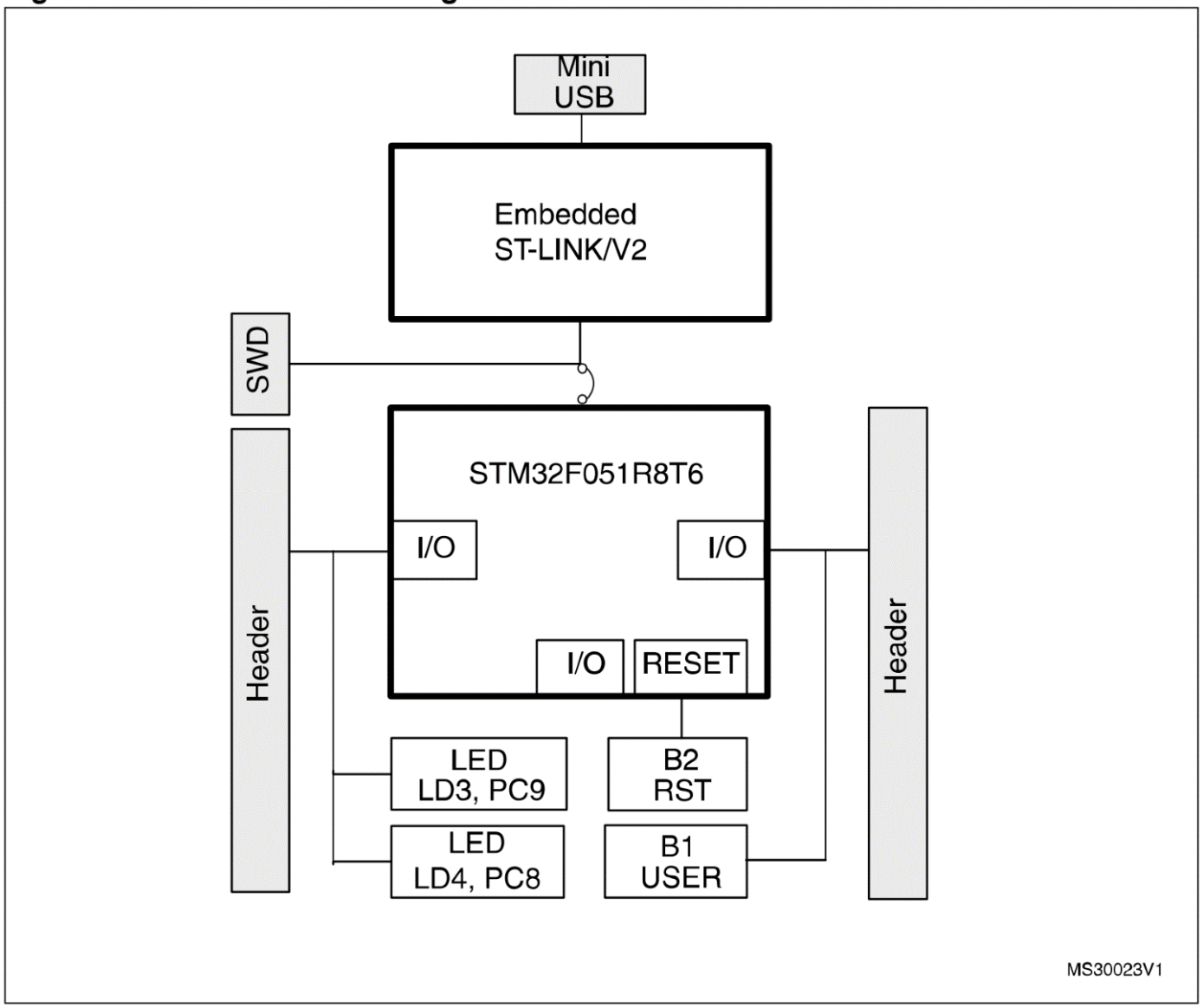

0xFFFF FFFF
7
0xE010 0000
Cortex-M0+ peripherals
0xE000 0000
6
0xC000 0000
5
0xA000 0000
4
0x8000 0000
3
0x6000 0000
2
Peripherals
0x4000 0000
1
SRAM
0x2000 0000
0
CODE
0x0000 0000
0x1FFF FFFF
Option bytes
System memory
reserved
Flash system memory
0x0800 0000
reserved
Flash, system memory or SRAM, demending on BOOT configuration
0x0000 0000
0x4800 17FF
AHB2
0x4800 0000
reserved
0x4002 43FF
AHB1
0x4002 0000
reserved
0x4001 8000
APB
0x4001 0000
reserved
0x4000 8000
APB
0x4000 0000
Reserved
MS33181V4

Reference Manual

- Some MCU vendors call it datasheet
- Download (RM0091)

Development Board

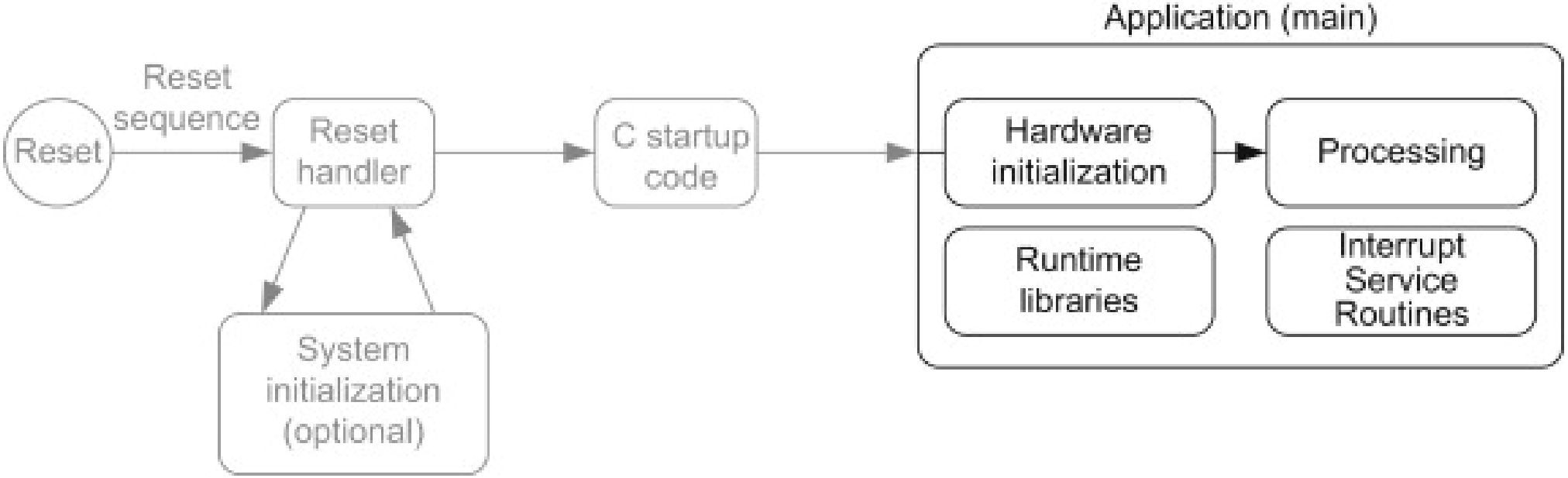
Figure 2. Hardware block diagram



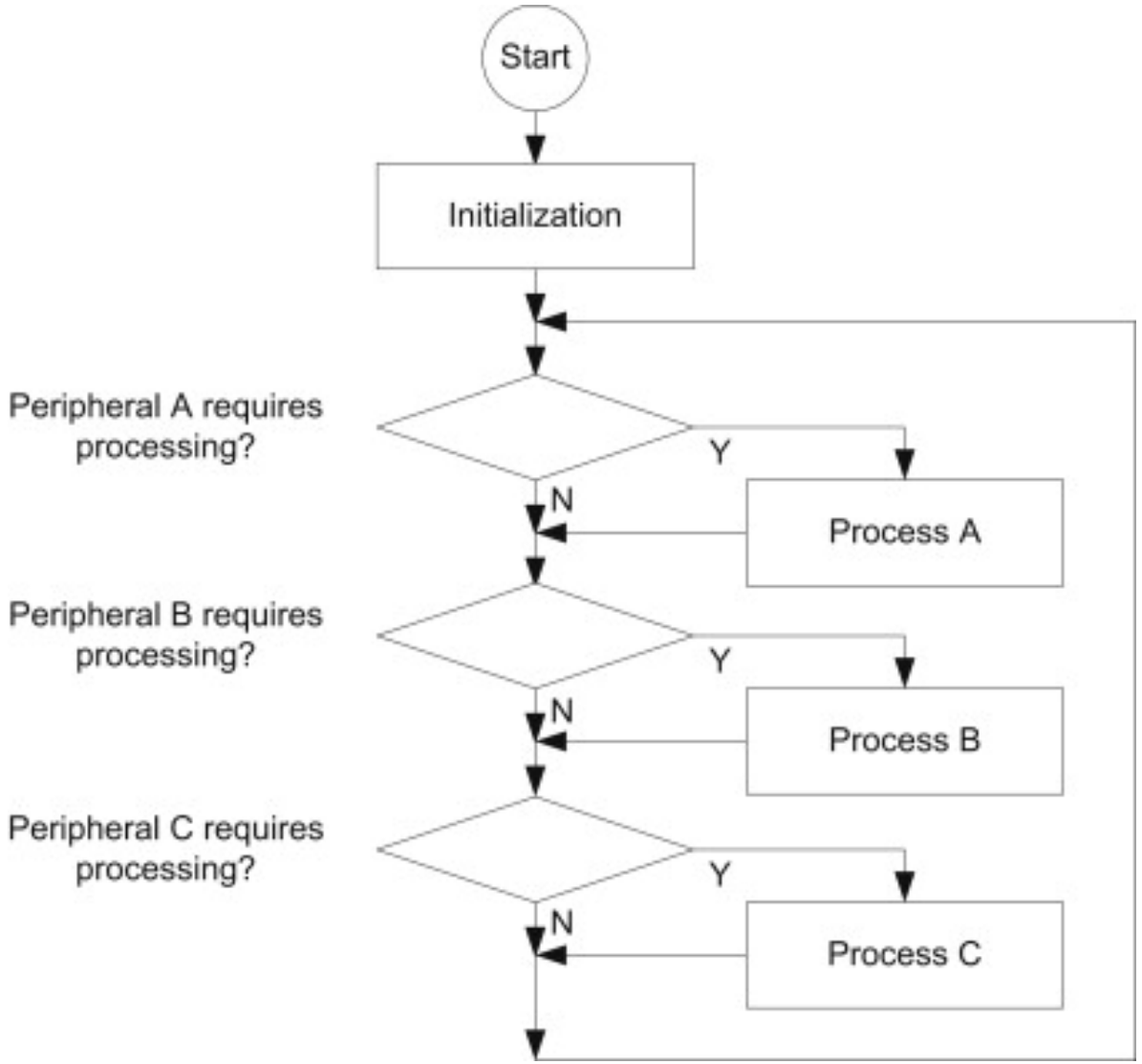
Cortex-M0 Programming

When a Microcontroller Starts

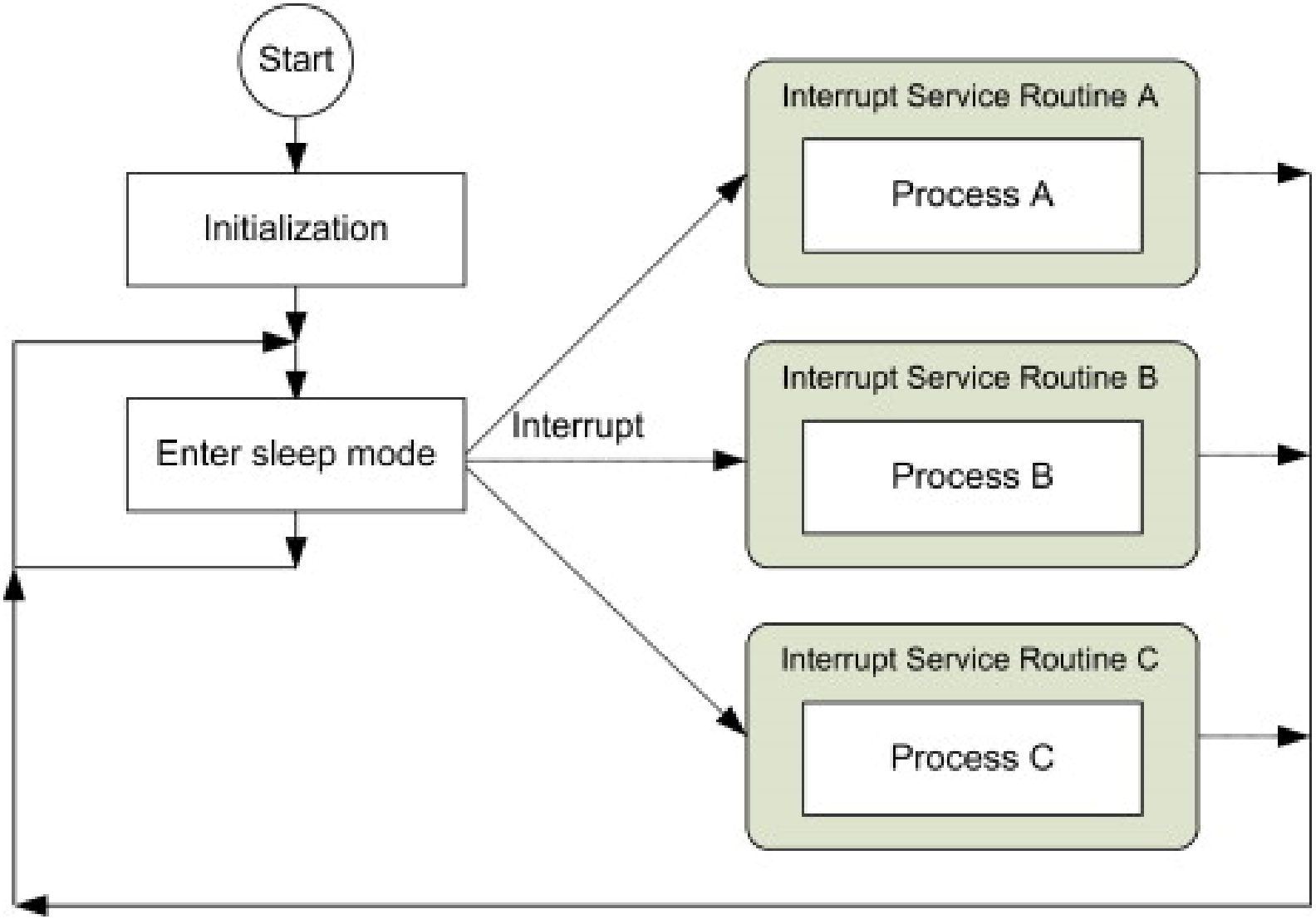
startup_stm32f0xx.s system_stm32f0xx.c



Polling


Start
Initialization
Peripheral A requires processing?
Y
N
Process A
Peripheral B requires processing?
Y
N
Process B
Peripheral C requires processing?
Y
N
Process C

Interrupt Driven



IDE

Integrated Development Environment

Development Toolchain

- IAR EWARM
  - 30-day evaluation edition
  - 32-Kbyte Limited QuickStart edition (16-Kbyte limitation for Cortex M0)
- Keil MDK-ARM
  - MDK-Lite (32-Kbyte code size limitation)
- ARMmbed
  - Free Online Development Tools
- GCC-based IDE

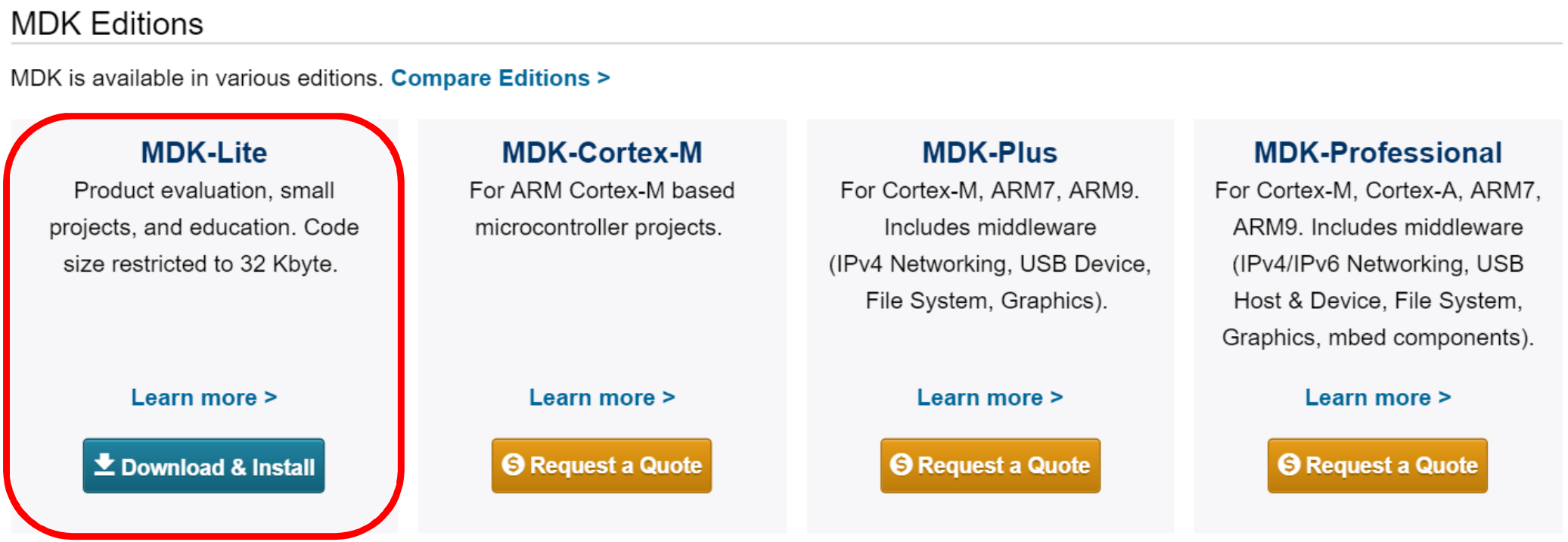
What do IDE Contain?

- C/C++ compilers
- Debuggers

Keil MDK Version 5 Development Kit

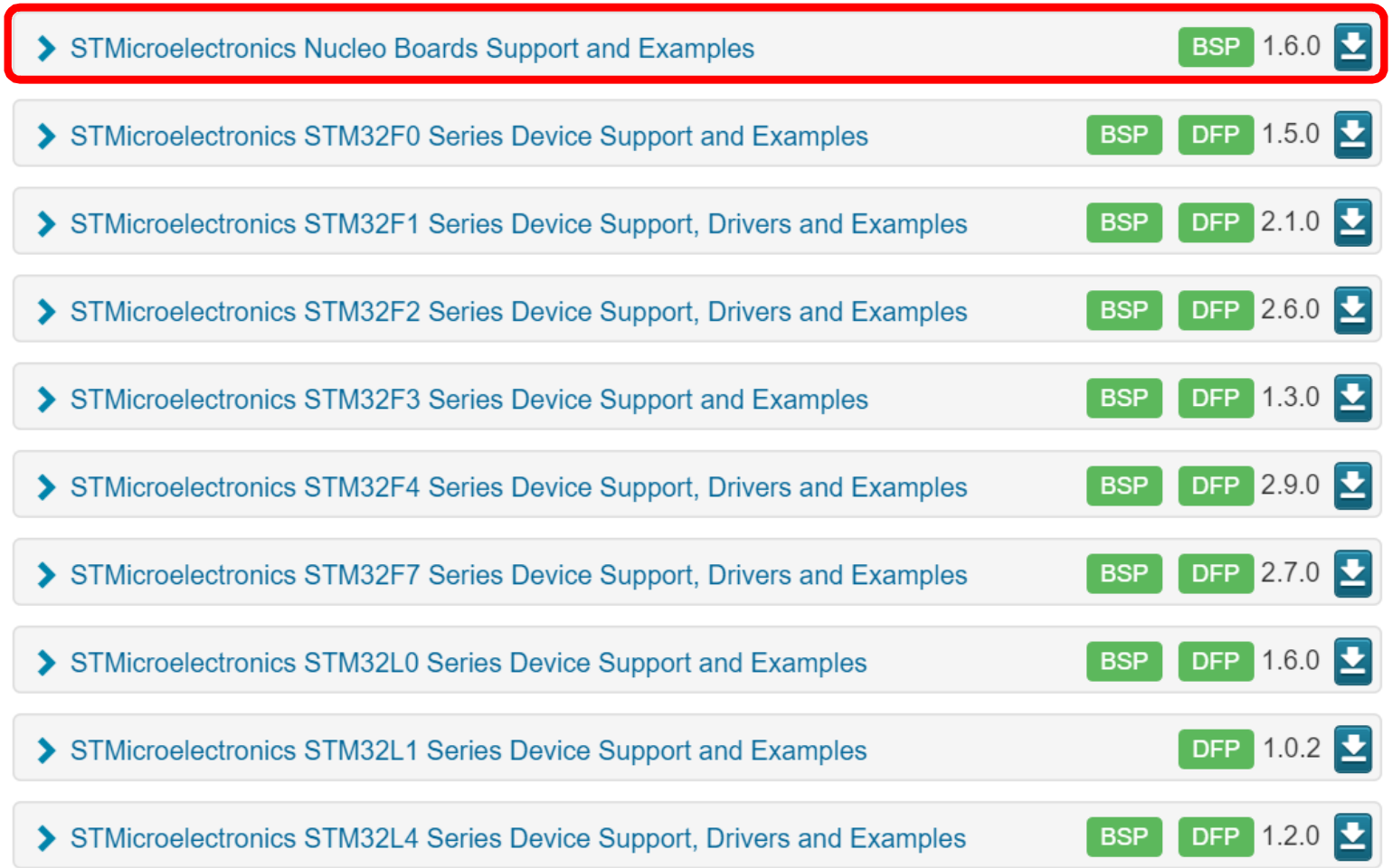
Step1. Install IDE

- Download from Kile.

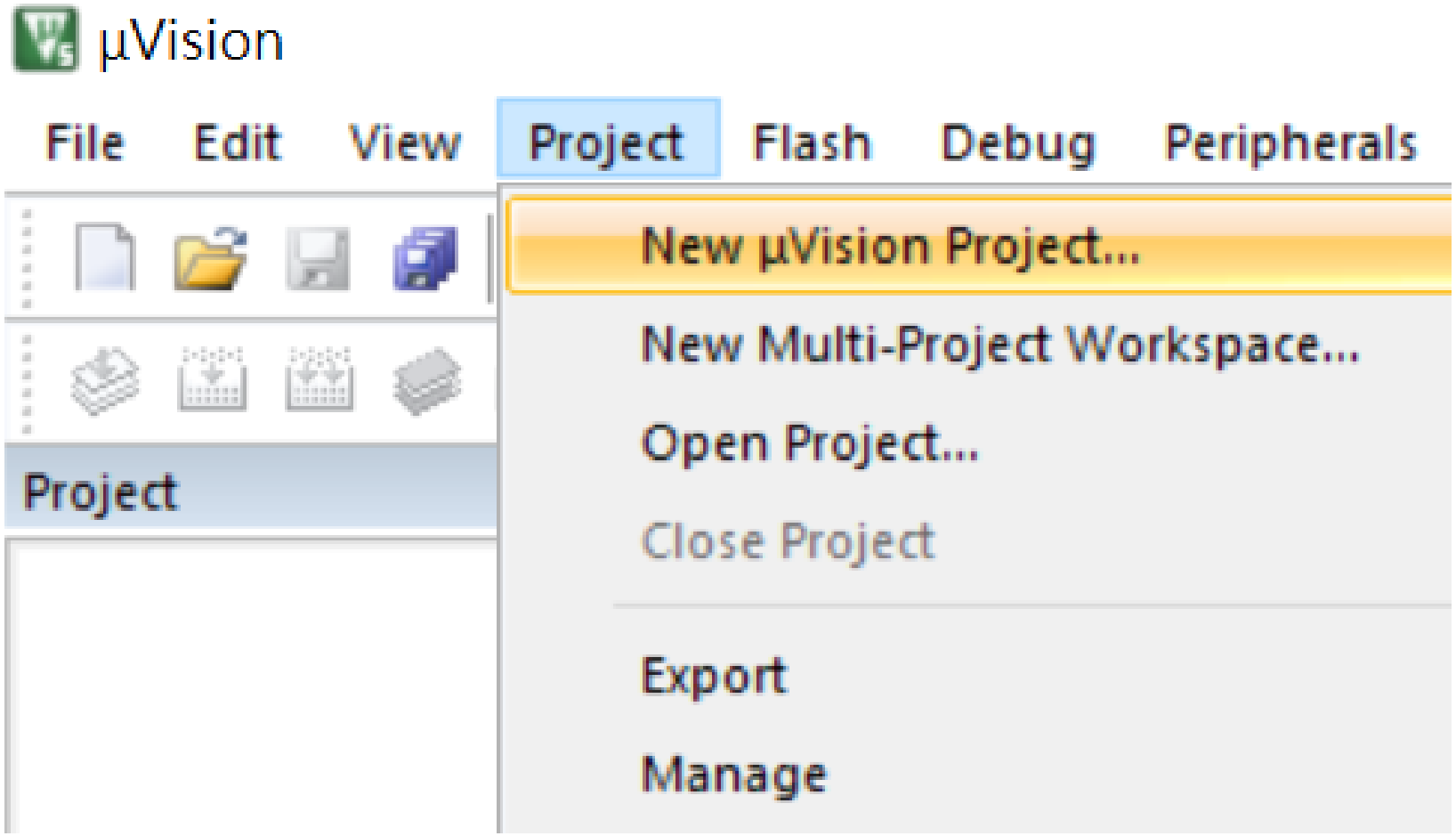


Step2. Install Software Packs

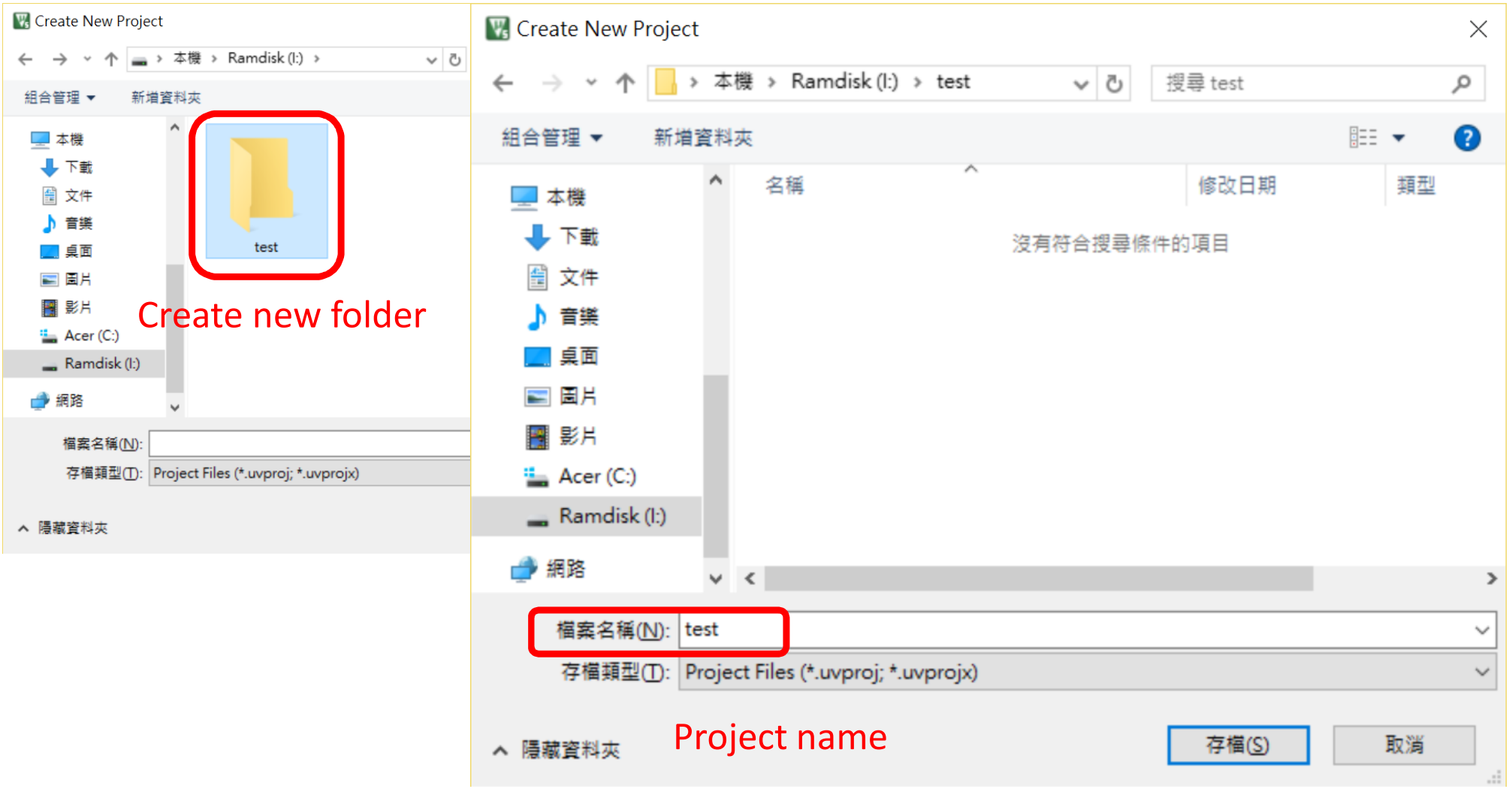
- Download from Kile.



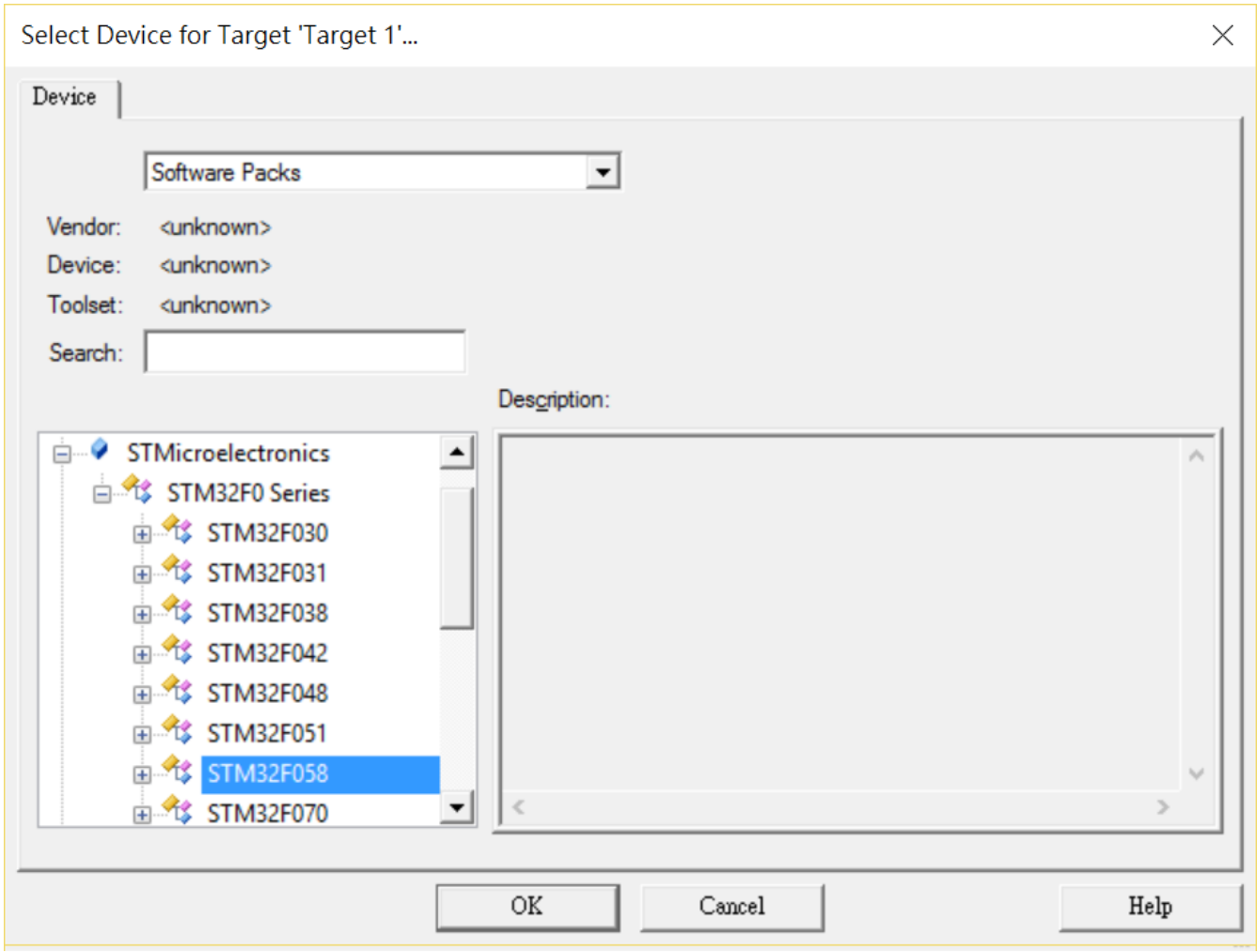
Step3. Create a New Project



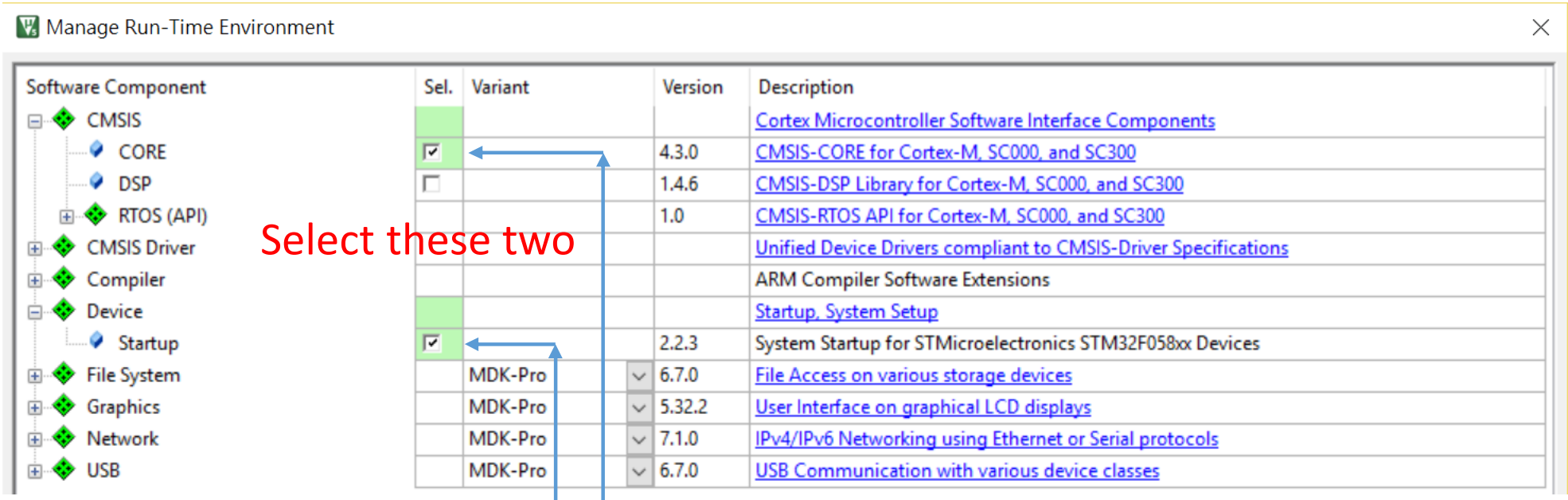
Step3. Create a New Project



Step4. Select Device



Step5. Add Software Component



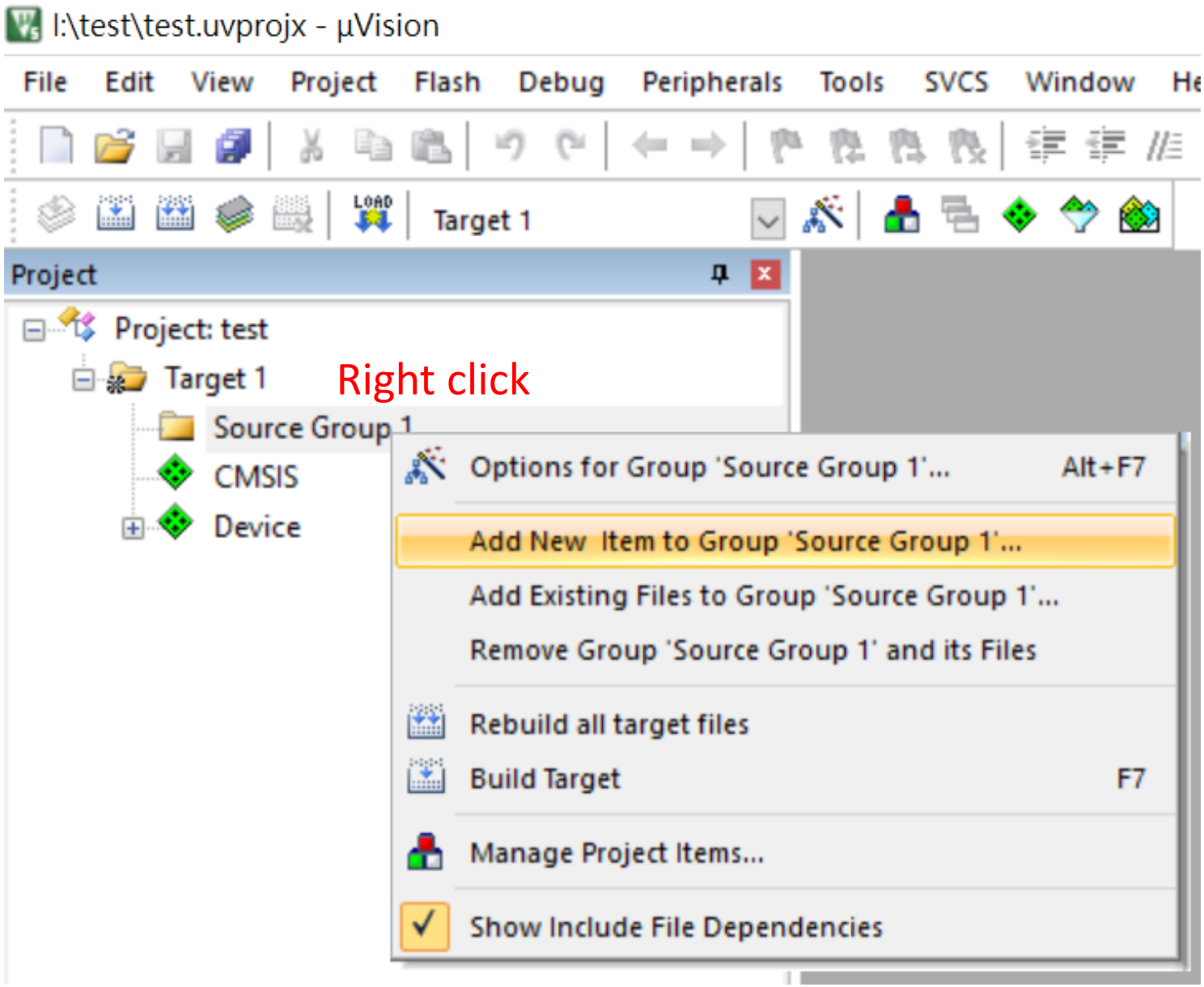
Select these two

core_cmx.h

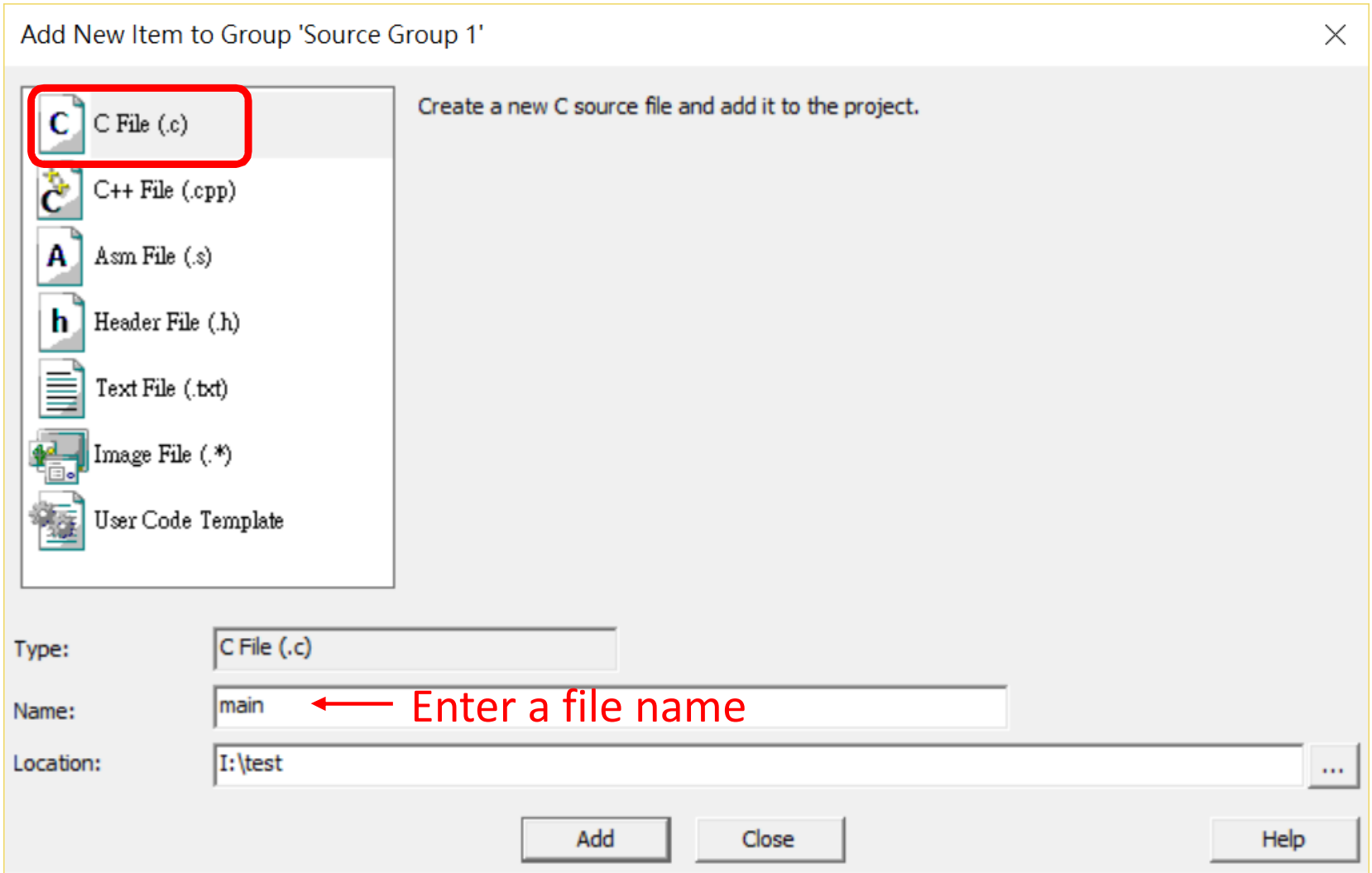
startup_<device>.s

system_<device>.c

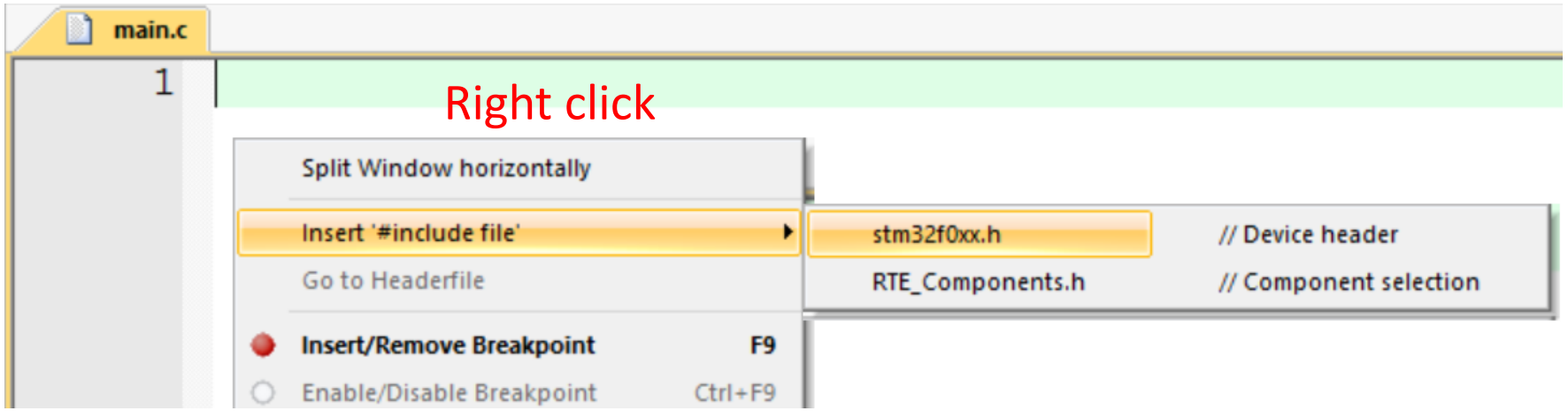
Step6. Add main.c

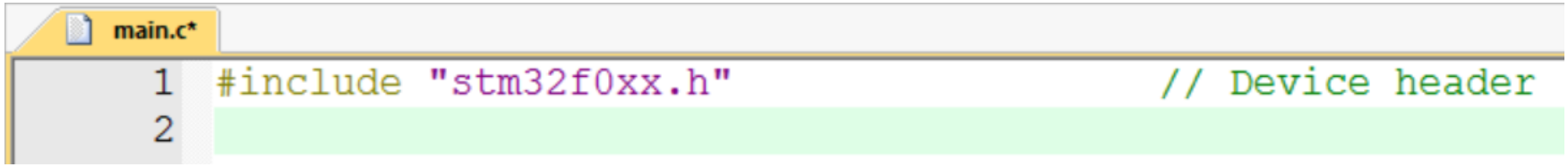


Step6. Add main.c

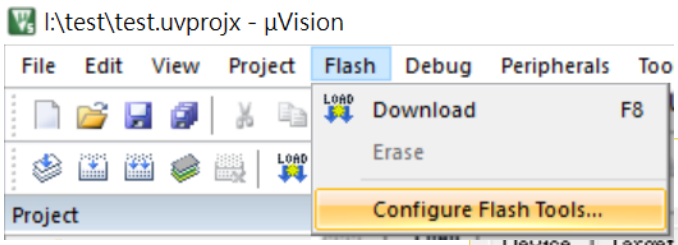


Step7. Include Device Header





Step8. Configure Flash



C/C++

Target 'Target 1'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

Preprocessor Symbols

Define:

Undefine:

Language / Code Generation

- [ ] Execute-only Code
- Optimization: Level 0 (-O0)
- [ ] Optimize for Time
- [ ] Split Load and Store Multiple
- [x] One ELF Section per Function
- [ ] Strict ANSI C
- [ ] Enum Container always int
- [ ] Plain Char is Signed
- [ ] Read-Only Position Independent
- [ ] Read-Write Position Independent

Warnings: All Warnings

- [ ] Thumb Mode
- [ ] No Auto Includes
- [x] C99 Mode

Click C99 Mode

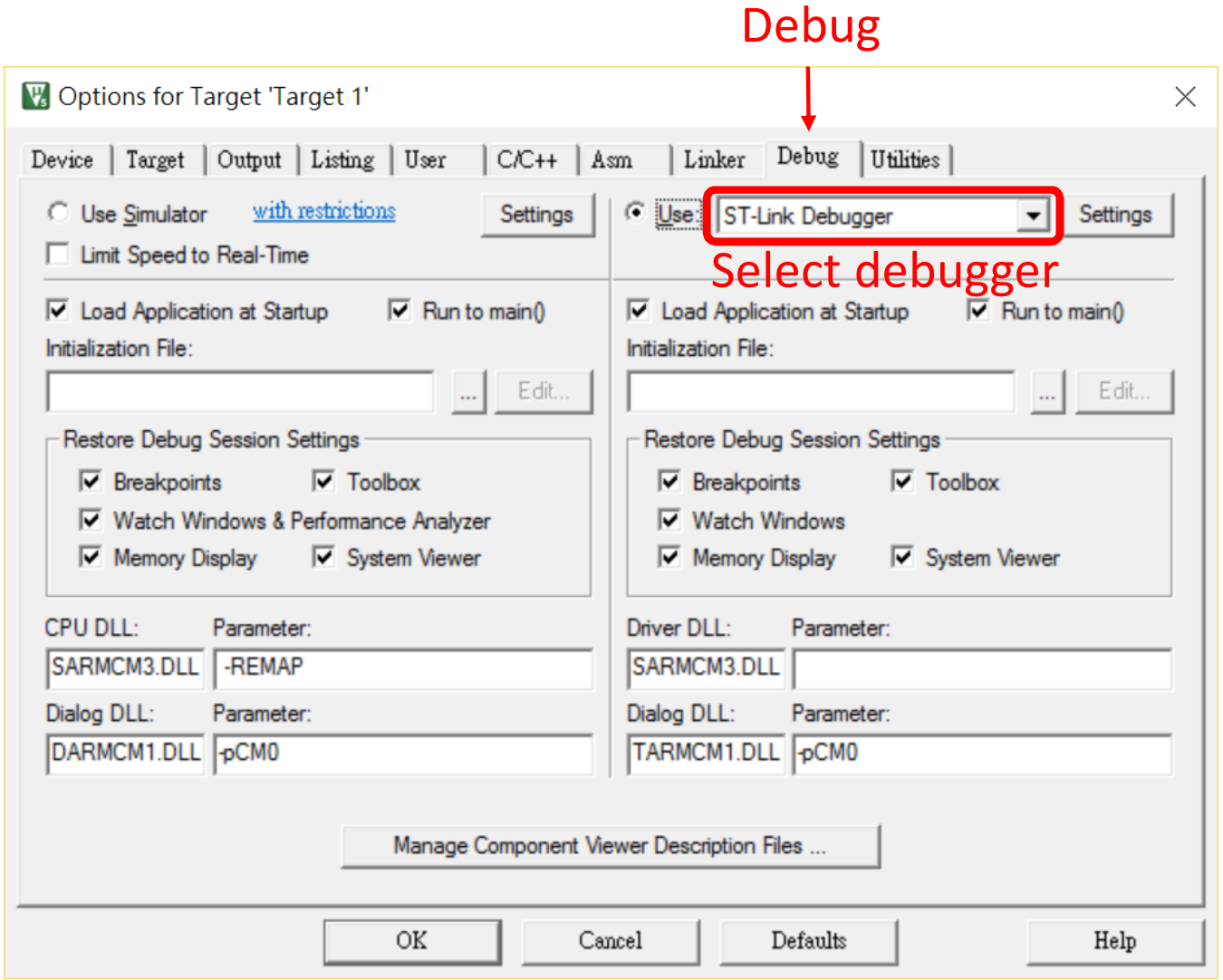
Include Paths

Misc Controls

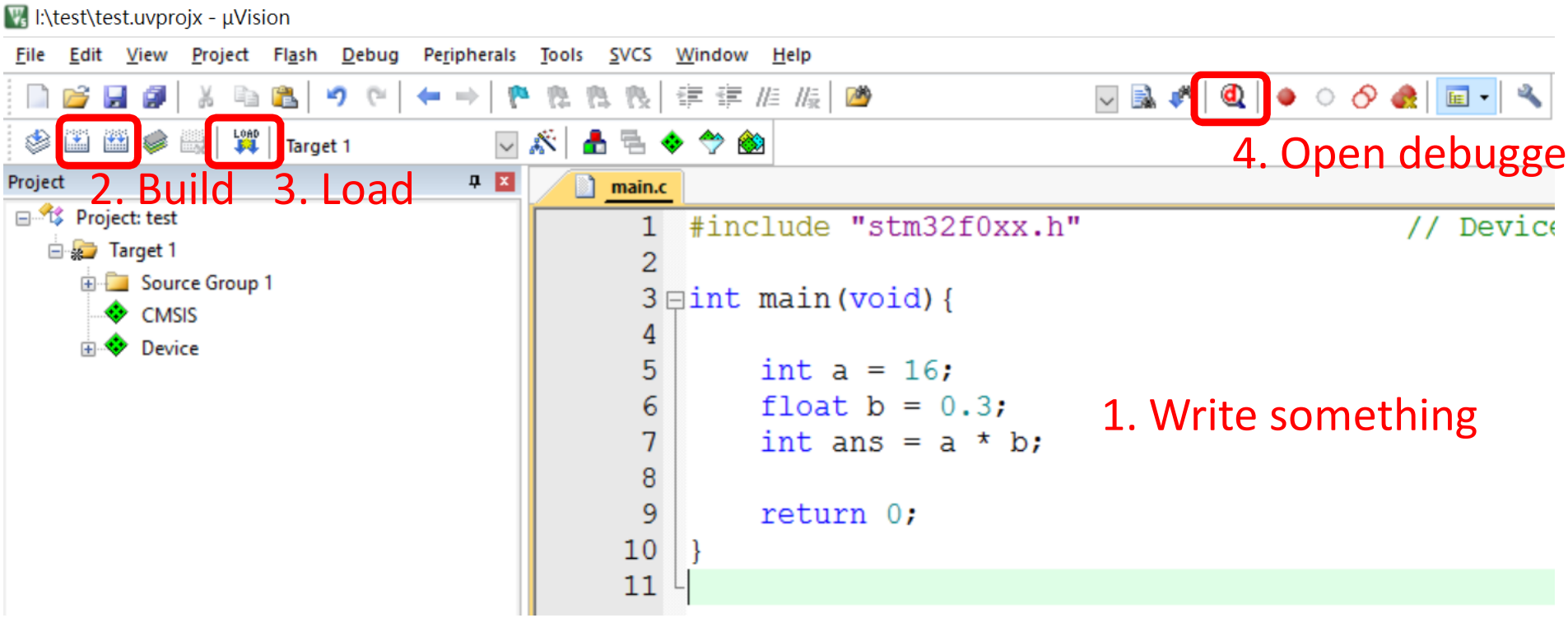
Compiler control string: --c99 -c --cpu Cortex-M0 -D__EVAL -g -O0 --apcs=interwork --split_sections -I I:\test\RTE

OK | Cancel | Defaults | Help

Step8. Configure Flash

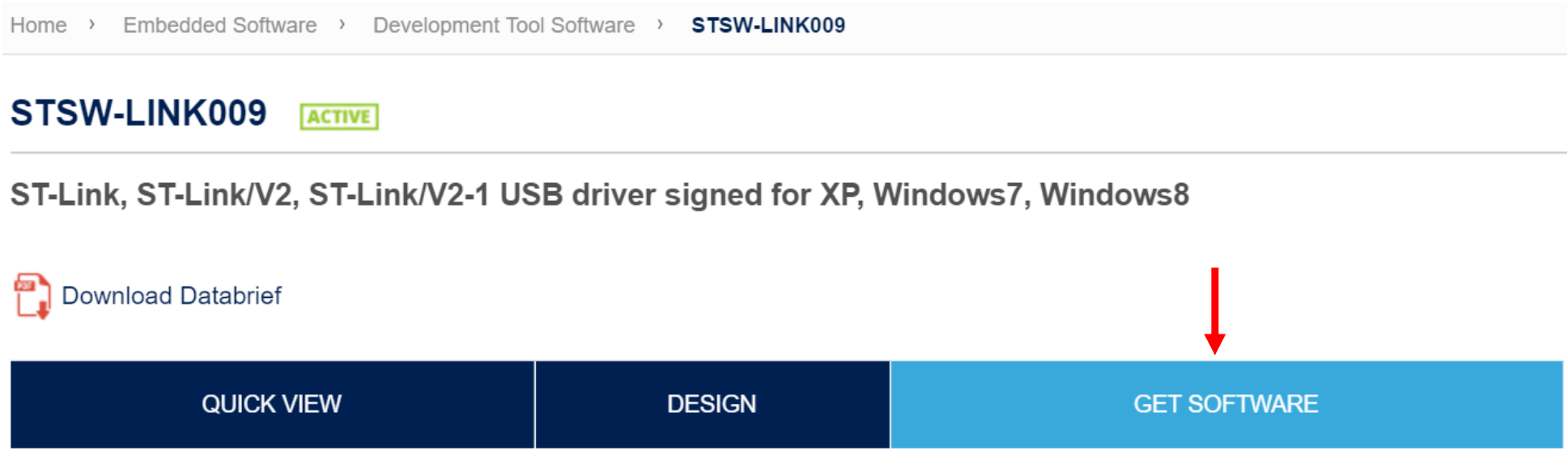


Step9. Test


I:\test\test.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
2. Build
3. Load
4. Open debugger
Project
Project: test
Target 1
Source Group 1
CMSIS
Device
main.c
1 #include "stm32f0xx.h" // Device
2
3 int main(void){
4
5 int a = 16;
6 float b = 0.3;
7 int ans = a * b;
8
9 return 0;
10 }
11
1. Write something

Install Debugger USB Driver

- If your board debugger doesn't work, download from ST.



Exercise

Do it back home.

Basic C Operator

main.c

```c
#include "stm32f0xx.h"

int a = 3, b = 6, ans;
int main(void){
    while(1){
        ans = a + b;
        ans = a - b;
        ans = a * b;
        ans = a / b;
        ans = a % b;
        ans = a | b;
        ans = a & b;
        ans = a ^ b;
        ans = ~ans;
    }
    return 0;
}
```

Use Debugger

- Review answer step by step.
- Monitor the variable you want.
- Set break points.